Classification

In [21]:
```
import pickle
import time
```

In [22]:
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

In [23]:
```
x = pickle.load(open('x.pkl','rb'))
y = pickle.load(open('y.pkl','rb'))
```

In [24]:
```
x
```

Out[24]:
```
array([[[[149, 179, 218],
         [138, 182, 235],
         [141, 190, 252],
         ...,
         [172, 198, 245],
         [179, 206, 253],
         [197, 209, 241]],

        [[152, 181, 220],
         [143, 187, 240],
         [130, 179, 241],
         ...,
         [176, 200, 241],
         [184, 206, 248],
         [201, 214, 246]],

        [[158, 188, 227],
         [137, 181, 234],
         [127, 176, 239],
         ...,
         [190, 208, 242],
         [196, 212, 246],
         [203, 215, 247]],

        ...,

        [[149, 179, 227],
         [131, 171, 234],
         [129, 169, 234],
         ...,
         [130, 164, 234],
         [130, 164, 234],
         [116, 170, 236]],

        [[154, 184, 231],
         [136, 176, 239],
         [134, 174, 240],
         ...,
         [125, 159, 229],
         [125, 159, 229],
         [124, 176, 236]],

        [[148, 179, 226],
         [142, 182, 245],
         [137, 177, 243],
         ...,
         [135, 169, 239],
         [138, 172, 242],
         [126, 173, 226]]],
```

```
        [123, 138, 184]],

       [[251, 253, 254],
        [241, 246, 248],
        [224, 227, 233],
        ...,
        [106, 119, 163],
        [108, 120, 164],
        [106, 123, 166]],

       ...,

       [[ 49,  52,  57],
        [ 53,  56,  65],
        [ 60,  64,  79],
        ...,
        [104, 116, 173],
        [119, 134, 191],
        [143, 159, 204]],

       [[ 76,  78, 103],
        [ 89,  94, 116],
        [ 96, 100, 129],
        ...,
        [ 90,  98, 148],
        [100, 111, 161],
        [117, 126, 173]],

       [[113, 113, 149],
        [117, 118, 156],
        [116, 121, 160],
        ...,
        [ 70,  77, 116],
        [ 77,  84, 126],
        [ 86,  93, 142]]]], dtype=uint8)
```

In [25]:

```
x = x/255

x
```

Out[25]:

```
array([[[[0.58431373, 0.70196078, 0.85490196],
         [0.54117647, 0.71372549, 0.92156863],
         [0.55294118, 0.74509804, 0.98823529],
         ...,
         [0.6745098 , 0.77647059, 0.96078431],
         [0.70196078, 0.80784314, 0.99215686],
         [0.77254902, 0.81960784, 0.94509804]],

        [[0.59607843, 0.70980392, 0.8627451 ],
         [0.56078431, 0.73333333, 0.94117647],
         [0.50980392, 0.70196078, 0.94509804],
         ...,
         [0.69019608, 0.78431373, 0.94509804],
         [0.72156863, 0.80784314, 0.97254902],
         [0.78823529, 0.83921569, 0.96470588]],

        [[0.61960784, 0.7372549 , 0.89019608],
         [0.5372549 , 0.70980392, 0.91764706],
         [0.49803922, 0.69019608, 0.9372549 ],
         ...,
         [0.74509804, 0.81568627, 0.94901961],
         [0.76862745, 0.83137255, 0.96470588],
         [0.79607843, 0.84313725, 0.96862745]],

        ...,

        [[0.58431373, 0.70196078, 0.89019608],
         [0.51372549, 0.67058824, 0.91764706],
         [0.50588235, 0.6627451 , 0.91764706],
         ...,
         [0.50980392, 0.64313725, 0.91764706],
         [0.50980392, 0.64313725, 0.91764706],
         [0.45490196, 0.66666667, 0.9254902 ]],
```

```
      [[[1.        , 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        ...,
        [0.49803922, 0.5372549 , 0.72156863],
        [0.50588235, 0.54509804, 0.73333333],
        [0.52941176, 0.55686275, 0.74901961]],

       [[1.        , 0.99607843, 0.99215686],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        ...,
        [0.47058824, 0.50980392, 0.69411765],
        [0.4745098 , 0.51372549, 0.69803922],
        [0.48235294, 0.54117647, 0.72156863]],

       [[0.98431373, 0.99215686, 0.99607843],
        [0.94509804, 0.96470588, 0.97254902],
        [0.87843137, 0.89019608, 0.91372549],
        ...,
        [0.41568627, 0.46666667, 0.63921569],
        [0.42352941, 0.47058824, 0.64313725],
        [0.41568627, 0.48235294, 0.65098039]],

       ...,

       [[0.19215686, 0.20392157, 0.22352941],
        [0.20784314, 0.21960784, 0.25490196],
        [0.23529412, 0.25098039, 0.30980392],
        ...,
        [0.40784314, 0.45490196, 0.67843137],
        [0.46666667, 0.5254902 , 0.74901961],
        [0.56078431, 0.62352941, 0.8       ]],

       [[0.29803922, 0.30588235, 0.40392157],
        [0.34901961, 0.36862745, 0.45490196],
        [0.37647059, 0.39215686, 0.50588235],
        ...,
        [0.35294118, 0.38431373, 0.58039216],
        [0.39215686, 0.43529412, 0.63137255],
        [0.45882353, 0.49411765, 0.67843137]],

       [[0.44313725, 0.44313725, 0.58431373],
        [0.45882353, 0.4627451 , 0.61176471],
        [0.45490196, 0.4745098 , 0.62745098],
        ...,
        [0.2745098 , 0.30196078, 0.45490196],
        [0.30196078, 0.32941176, 0.49411765],
        [0.3372549 , 0.36470588, 0.55686275]]]])
```

In [26]:
```
y
```

Out[26]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0])
```

In [27]:
```
x.shape
```

Out[27]:
```
(131, 150, 150, 3)
```

In [28]:
```
model = Sequential()
model.add(Conv2D(64, (3,3), activation = 'relu' ))
model.add(MaxPooling2D((2,2)))
```

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

In [29]:
```
model.add(Flatten())
```

In [30]:
```
model.add(Dense(128,  input_shape = x.shape[1:],  activation = 'relu'))
```

In [31]:
```
model.add(Dense(2,  activation = 'softmax'))
```

In [32]:
```
model.compile(optimizer = 'adam',  loss =
'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

In [33]:
```
history = model.fit(x,y,epochs = 10,  validation_split = 0.2,batch_size
= 10,  callbacks =[tensorboard])  #batch_size = 32
```

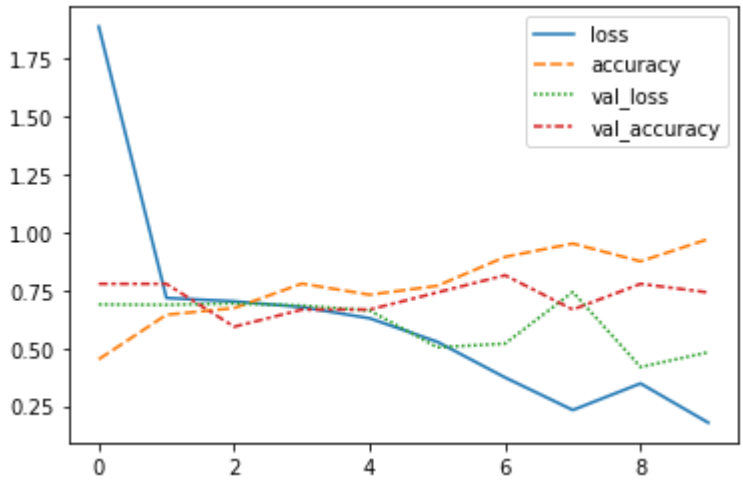
```
Epoch 1/10
 2/11 [====>.........................] - ETA: 10s - loss: 3.7444 - accuracy: 0.4000WA
RNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch
time (batch time: 0.6001s vs `on_train_batch_end` time: 1.6963s). Check your callback
s.
11/11 [==============================] - 8s 758ms/step - loss: 1.8891 - accuracy: 0.4
519 - val_loss: 0.6889 - val_accuracy: 0.7778
Epoch 2/10
11/11 [==============================] - 7s 600ms/step - loss: 0.7167 - accuracy: 0.6
442 - val_loss: 0.6872 - val_accuracy: 0.7778
Epoch 3/10
11/11 [==============================] - 6s 527ms/step - loss: 0.7023 - accuracy: 0.6
731 - val_loss: 0.6931 - val_accuracy: 0.5926
Epoch 4/10
11/11 [==============================] - 6s 554ms/step - loss: 0.6774 - accuracy: 0.7
788 - val_loss: 0.6838 - val_accuracy: 0.6667
Epoch 5/10
11/11 [==============================] - 6s 525ms/step - loss: 0.6290 - accuracy: 0.7
308 - val_loss: 0.6626 - val_accuracy: 0.6667
Epoch 6/10
11/11 [==============================] - 6s 550ms/step - loss: 0.5275 - accuracy: 0.7
692 - val_loss: 0.5035 - val_accuracy: 0.7407
Epoch 7/10
11/11 [==============================] - 6s 546ms/step - loss: 0.3734 - accuracy: 0.8
942 - val_loss: 0.5197 - val_accuracy: 0.8148
Epoch 8/10
11/11 [==============================] - 7s 623ms/step - loss: 0.2328 - accuracy: 0.9
519 - val_loss: 0.7425 - val_accuracy: 0.6667
Epoch 9/10
11/11 [==============================] - 6s 589ms/step - loss: 0.3476 - accuracy: 0.8
750 - val_loss: 0.4185 - val_accuracy: 0.7778
Epoch 10/10
11/11 [==============================] - 6s 546ms/step - loss: 0.1782 - accuracy: 0.9
712 - val_loss: 0.4823 - val_accuracy: 0.7407
```

In [34]:
```
history.history.keys()
```

Out[34]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [35]:
```
import seaborn as sns
sns.lineplot(data = history.history)
```

Out[35]:
```
<AxesSubplot:>
```


loss
accuracy
val_loss
val_accuracy
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
2
4
6
8