DETECTION OF RECOLORED IMAGES USING DEEP DISCRIMINATIVE MODEL

Training The Model

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random
import os
print(os.listdir("dataset"))
```

```
Using TensorFlow backend.
```

```
['originalimg', 'recolorimg', 'source', 'target', 'testingset', 'trainingse
t']
```

In [2]:

```
FAST_RUN = False
IMAGE_WIDTH=128
IMAGE_HEIGHT=128
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3 # RGB color
```

- Use the dataset created by RecImDet.ipnyb for training the model

In [3]:

```
filenames = os.listdir("dataset/trainingset")
categories = []
for filename in filenames:
    category = filename.split('.')[0]
    if category == 'pic':
        categories.append(1)
    else:
        categories.append(0)

df = pd.DataFrame({
    'filename': filenames,
    'category': categories
})
```

In [4]:

```
1 df.head()
```

Out[4]:

	filename	category
0	img.0.jpg	0
1	img.1.jpg	0
2	img.10.jpg	0
3	img.11.jpg	0
4	img.12.jpg	0

In [5]:

```
1 df.tail()
```

Out[5]:

	filename	category
95	pic.5.jpg	1
96	pic.6.jpg	1
97	pic.7.jpg	1
98	pic.8.jpg	1
99	pic.9.jpg	1

In [6]:

```
1 df['category']=df['category'].astype(str)
```

In [7]:

```
1 df.dtypes
```

Out[7]:

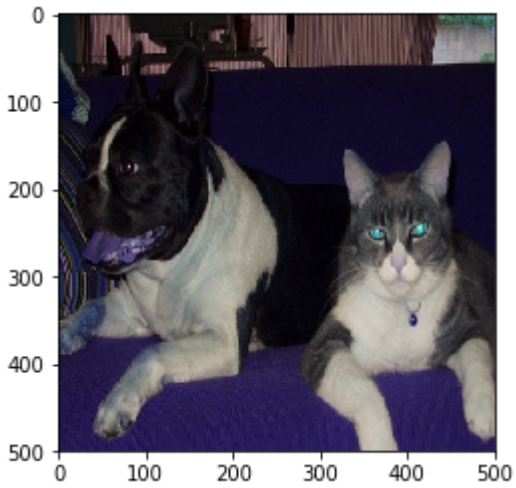
```
filename    object
category    object
dtype: object
```

In [8]:

```
sample = random.choice(filenames)
image = load_img("dataset/trainingset/"+sample)
plt.imshow(image)
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x1f9a38c9400>
```

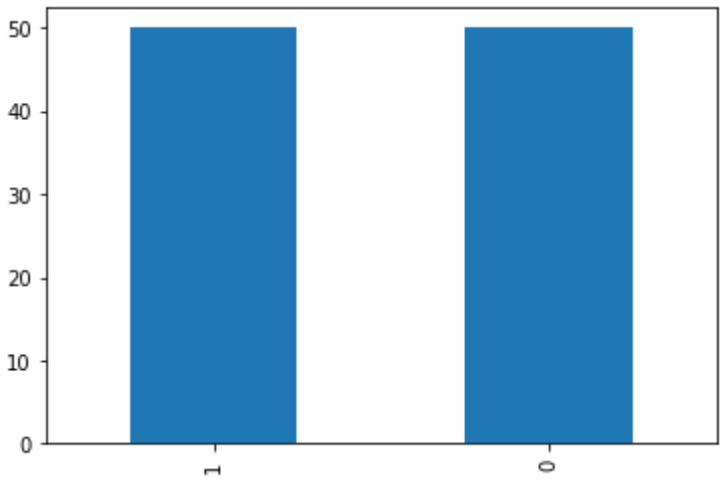


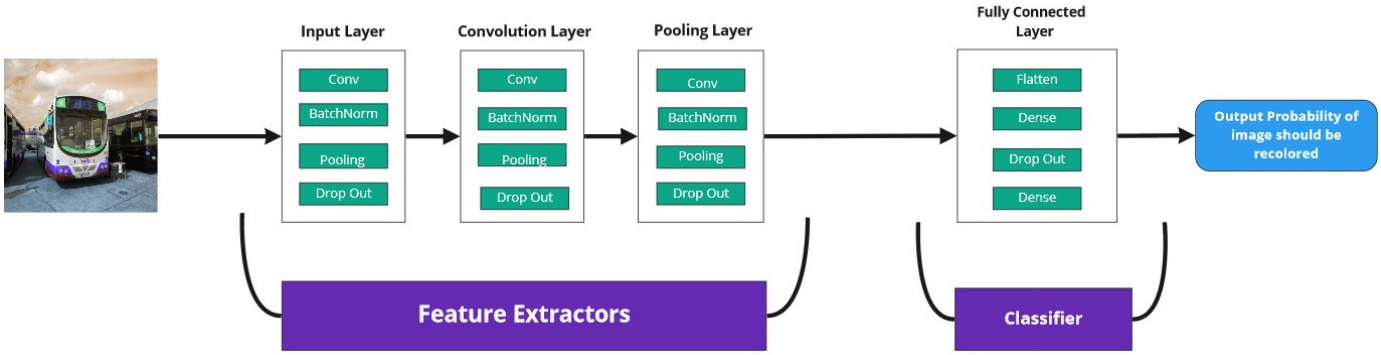
In [9]:

```
df['category'].value_counts().plot.bar()
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f9a3924c88>
```





- **Input Layer**: It represent input image data. It will reshape image into single diminsion array. Example your image is 64x64 = 4096, it will convert to (4096,1) array.
- **Convolution Layer**: This layer will extract features from image.
- **Pooling Layer**: This layerreduce the spatial volume of input image after convolution.
- **Fully Connected Layer**: It connect the network from a layer to another layer
- **Output Layer**: It is the predicted values layer.

In [10]:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, Bat

model = Sequential()

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

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

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

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

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

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 64)      1792
_________________________________________________________________
batch_normalization_1 (Batch (None, 126, 126, 64)      256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 64)        36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 61, 61, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 64)        36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 64)        256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_3 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12544)             0
```

```
_________________________________________________________________
dense_1 (Dense)              (None, 64)                802880
_________________________________________________________________
batch_normalization_4 (Batch (None, 64)                256
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 879,617
Trainable params: 879,105
Non-trainable params: 512
_________________________________________________________________
```

Callbacks

In [11]:

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

> **Early Stop**
>
> - To prevent over fitting we will stop the learning after 10 epochs and val_loss value not decreased

In [12]:

```
earlystop = EarlyStopping(monitor='val_loss',patience=10)
```

> **Learning Rate Reduction**
>
> - We will reduce the learning rate when then accuracy not increase for 2 steps

In [13]:

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss',
                                            patience=2,
                                            verbose=1,
                                            factor=0.05,
                                            min_lr=0.00001)
```

In [14]:

```
callbacks = [earlystop,learning_rate_reduction]
```

In [15]:

```
train_df, validate_df = train_test_split(df, test_size=0.20, random_state=42)
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
```

In [16]:

```
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
batch_size=4
```

In [17]:

```
print(train_df.shape)
print(validate_df.shape)
```

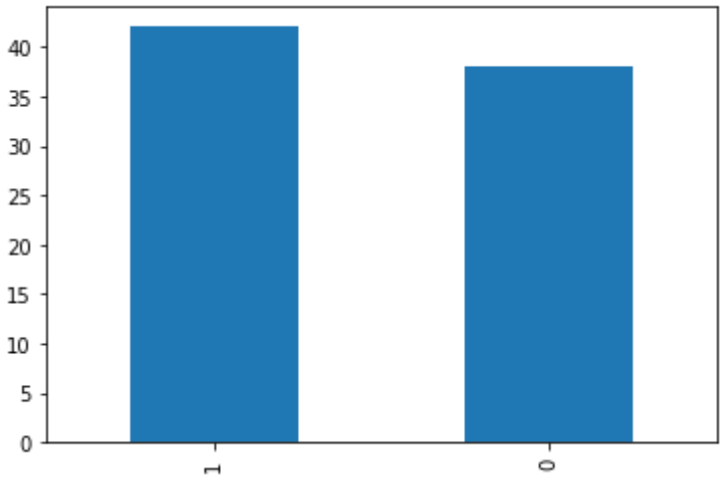
```
(80, 2)
(20, 2)
```

In [18]:

```
train_df['category'].value_counts().plot.bar()
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f9a4b60438>
```



In [19]:

```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)
```

In [20]:

```
train_generator = train_datagen.flow_from_dataframe(
    train_df,
    "dataset/trainingset",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='binary',
    batch_size=batch_size
)
```

```
Found 80 validated image filenames belonging to 2 classes.
```

In [21]:

```
validation_datagen = ImageDataGenerator(rescale=1./255)
validation_generator = validation_datagen.flow_from_dataframe(
    validate_df,
    "dataset/trainingset",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='binary',
    batch_size=batch_size
)
```

```
Found 20 validated image filenames belonging to 2 classes.
```

Fit the model

In [22]:

```
epochs=3 if FAST_RUN else 20
history = model.fit_generator(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=total_validate//batch_size,
    steps_per_epoch=total_train//batch_size,

)
```

```
Epoch 1/20
20/20 [==============================] - 13s 668ms/step - loss: 0.9005 - acc
uracy: 0.5250 - val_loss: 0.7533 - val_accuracy: 0.6000
Epoch 2/20
20/20 [==============================] - 9s 429ms/step - loss: 1.0249 - accu
racy: 0.4625 - val_loss: 0.7626 - val_accuracy: 0.6000
Epoch 3/20
20/20 [==============================] - 9s 440ms/step - loss: 0.8107 - accu
racy: 0.5000 - val_loss: 0.6533 - val_accuracy: 0.6000
Epoch 4/20
20/20 [==============================] - 10s 487ms/step - loss: 0.6818 - acc
uracy: 0.5875 - val_loss: 1.2919 - val_accuracy: 0.6000
Epoch 5/20
20/20 [==============================] - 9s 437ms/step - loss: 0.7050 - accu
racy: 0.6125 - val_loss: 1.6054 - val_accuracy: 0.6000
Epoch 6/20
20/20 [==============================] - 9s 436ms/step - loss: 0.8252 - accu
racy: 0.5500 - val_loss: 0.6178 - val_accuracy: 0.6000
Epoch 7/20
20/20 [==============================] - 9s 434ms/step - loss: 0.8256 - accu
racy: 0.5875 - val_loss: 1.2969 - val_accuracy: 0.6000
Epoch 8/20
20/20 [==============================] - 9s 434ms/step - loss: 0.8648 - accu
racy: 0.5500 - val_loss: 1.1442 - val_accuracy: 0.6000
Epoch 9/20
20/20 [==============================] - 9s 433ms/step - loss: 0.8990 - accu
racy: 0.4875 - val_loss: 0.9171 - val_accuracy: 0.6000
Epoch 10/20
20/20 [==============================] - 9s 447ms/step - loss: 0.7280 - accu
racy: 0.5750 - val_loss: 1.1564 - val_accuracy: 0.6000
Epoch 11/20
20/20 [==============================] - 9s 439ms/step - loss: 0.8013 - accu
racy: 0.5250 - val_loss: 1.7100 - val_accuracy: 0.6000
Epoch 12/20
20/20 [==============================] - 9s 448ms/step - loss: 0.9404 - accu
racy: 0.4000 - val_loss: 0.8377 - val_accuracy: 0.4500
Epoch 13/20
20/20 [==============================] - 9s 438ms/step - loss: 0.9270 - accu
racy: 0.5375 - val_loss: 0.6884 - val_accuracy: 0.4500
Epoch 14/20
20/20 [==============================] - 9s 438ms/step - loss: 0.7346 - accu
racy: 0.6375 - val_loss: 0.3362 - val_accuracy: 0.4000
Epoch 15/20
20/20 [==============================] - 9s 431ms/step - loss: 0.8160 - accu
racy: 0.5375 - val_loss: 0.8542 - val_accuracy: 0.5000
Epoch 16/20
20/20 [==============================] - 9s 433ms/step - loss: 0.8658 - accu
racy: 0.5875 - val_loss: 1.6295 - val_accuracy: 0.5000
Epoch 17/20
```

```
20/20 [==============================] - 9s 436ms/step - loss: 0.7633 - accu
racy: 0.6125 - val_loss: 0.7884 - val_accuracy: 0.4000
Epoch 18/20
20/20 [==============================] - 9s 446ms/step - loss: 0.7673 - accu
racy: 0.6125 - val_loss: 0.9463 - val_accuracy: 0.3500
Epoch 19/20
20/20 [==============================] - 9s 434ms/step - loss: 0.8648 - accu
racy: 0.4625 - val_loss: 0.6554 - val_accuracy: 0.3500
Epoch 20/20
20/20 [==============================] - 9s 433ms/step - loss: 0.7152 - accu
racy: 0.5875 - val_loss: 1.0763 - val_accuracy: 0.3500
```

Save the model

In [23]:

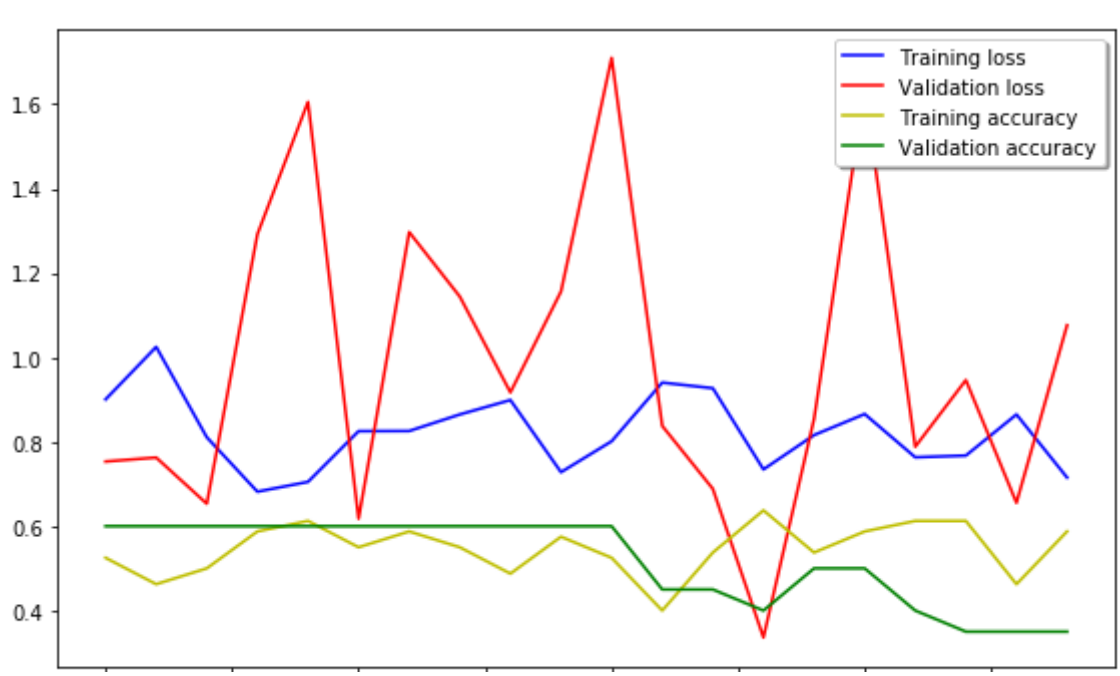
```
model.save_weights("RecImgDecNet.h5")
```

Visualize Training

In [24]:

```
fig, (ax1) = plt.subplots(1, 1, figsize=(8, 5))
ax1.plot(history.history['loss'], color='b', label="Training loss")
ax1.plot(history.history['val_loss'], color='r', label="Validation loss")
ax1.plot(history.history['accuracy'], color='y', label="Training accuracy")
ax1.plot(history.history['val_accuracy'], color='g',label="Validation accuracy")


#ax1.set_xticks(np.arange(1, epochs, 1))
#ax1.set_yticks(np.arange(0, 1, 0.1))
legend = plt.legend(loc='best', shadow=True)
plt.tight_layout()
plt.show()
```



Testing The Model

Preparing Test Data

In [25]:

```
test_filenames = os.listdir("dataset/testingset")
test_df = pd.DataFrame({
    'filename': test_filenames
})
nb_samples = test_df.shape[0]
```

In [26]:

```
print(nb_samples)
```

```
32
```

Creating Test Generator

In [27]:

```
test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(
    test_df,
    "dataset/testingset",
    x_col='filename',
    y_col=None,
    class_mode=None,
    target_size=IMAGE_SIZE,
    batch_size=batch_size,
    shuffle=False
)
```

```
Found 32 validated image filenames.
```

Predict

- For categoral classication the prediction will come with probability of each category.

In [28]:

```
predict = model.predict_generator(test_generator, steps=np.ceil(nb_samples/batch_size))
```

In [29]:

```
print(predict)
```

```
[[0.59206027]
 [0.41791338]
 [0.6814715 ]
 [0.32960314]
 [0.60395145]
 [0.23712257]
 [0.5060772 ]
 [0.36897856]
 [0.33071783]
 [0.22974512]
 [0.2097553 ]
 [0.6014137 ]
 [0.21073207]
 [0.31700346]
 [0.39265856]
 [0.3189384 ]
 [0.24531238]
 [0.28414643]
 [0.11258423]
 [0.18251933]
 [0.43914756]
 [0.44274154]
 [0.4459205 ]
 [0.3170194 ]
 [0.61339784]
 [0.47437096]
 [0.6779071 ]
 [0.57538396]
 [0.4094928 ]
 [0.5831324 ]
 [0.54973006]
 [0.30086952]]
```

In [30]:

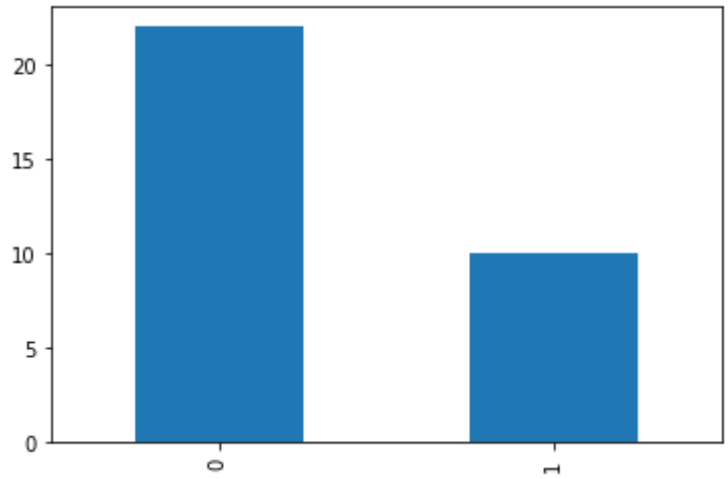
```
threshold = 0.5
test_df['probability'] = predict
test_df['category'] = np.where(test_df['probability'] > threshold, 1,0)
```

In [31]:

```
1 test_df['category'].value_counts().plot.bar()
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f9a67fc908>
```



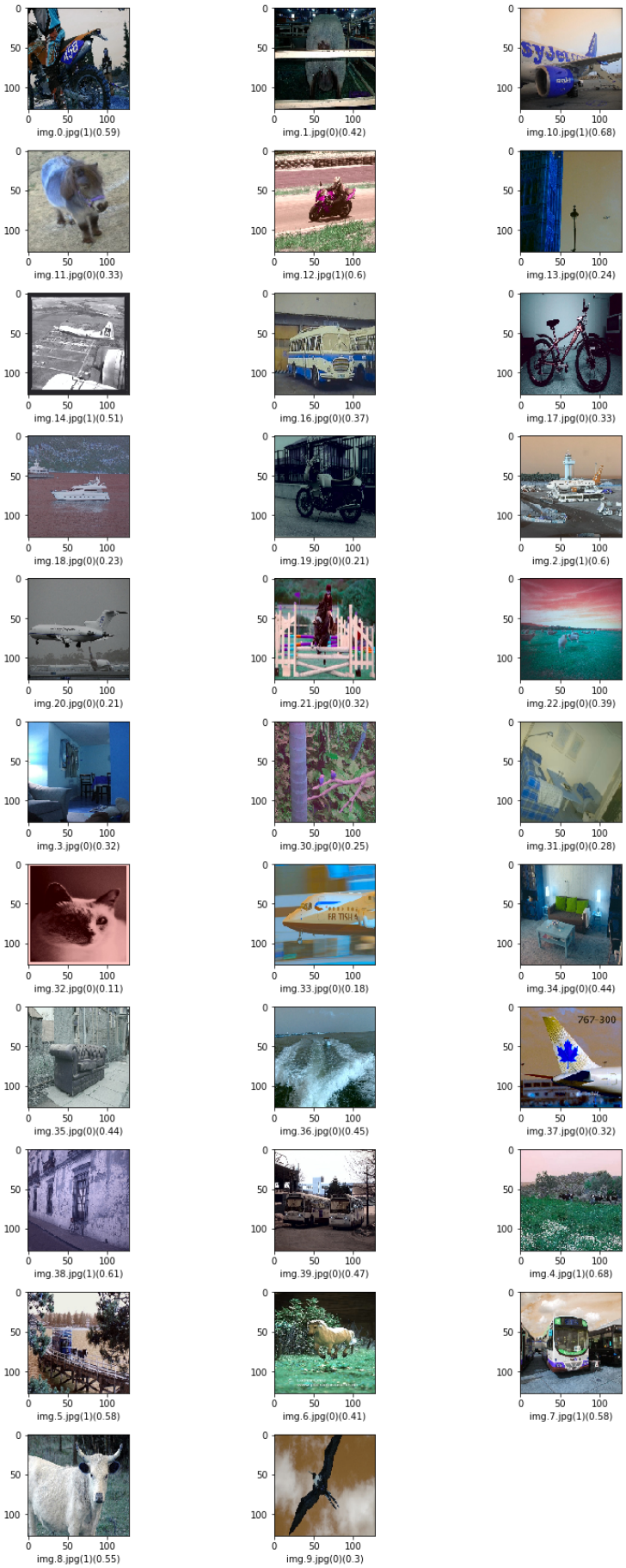
Visualize the Test Results

In [32]:

```
sample_test = test_df #.head(5)
sample_test.head()
plt.figure(figsize=(12, 24))
for index, row in sample_test.iterrows():
    filename = row['filename']
    category = row['category']
    probability = row['probability']
    img = load_img("dataset/testingset/"+filename, target_size=IMAGE_SIZE)
    plt.subplot(11, 3, index+1)
    plt.imshow(img)
    plt.xlabel(filename + '(' + "{}".format(category) + ')' '(' + "{}".format(round(pro
plt.figtext(1,1,'Category(0):RECOLORED,Category(1):ORIGINAL',fontsize='large')
plt.tight_layout()
plt.show()

```

Category(0):RECOLORED,Category(1):ORIGINAL



Submission of Test Results to a CSV

In [33]:

```
submission_df = test_df.copy()
submission_df['id'] = submission_df['filename'].str.split('.').str[0]
submission_df['label'] = submission_df['category']
submission_df.drop(['filename', 'category'], axis=1, inplace=True)
submission_df.to_csv('submission.csv', index=False)
```

In [34]:

```
print('\n \n Category(0):RECOLORED,Category(1):ORIGINAL \n \n')
print(test_df)
```

```
 Category(0):RECOLORED,Category(1):ORIGINAL


      filename  probability  category
0    img.0.jpg     0.592060         1
1    img.1.jpg     0.417913         0
2   img.10.jpg     0.681472         1
3   img.11.jpg     0.329603         0
4   img.12.jpg     0.603951         1
5   img.13.jpg     0.237123         0
6   img.14.jpg     0.506077         1
7   img.16.jpg     0.368979         0
8   img.17.jpg     0.330718         0
9   img.18.jpg     0.229745         0
10  img.19.jpg     0.209755         0
11   img.2.jpg     0.601414         1
12  img.20.jpg     0.210732         0
13  img.21.jpg     0.317003         0
14  img.22.jpg     0.392659         0
15   img.3.jpg     0.318938         0
16  img.30.jpg     0.245312         0
17  img.31.jpg     0.284146         0
18  img.32.jpg     0.112584         0
19  img.33.jpg     0.182519         0
20  img.34.jpg     0.439148         0
21  img.35.jpg     0.442742         0
22  img.36.jpg     0.445920         0
23  img.37.jpg     0.317019         0
24  img.38.jpg     0.613398         1
25  img.39.jpg     0.474371         0
26   img.4.jpg     0.677907         1
27   img.5.jpg     0.575384         1
28   img.6.jpg     0.409493         0
29   img.7.jpg     0.583132         1
30   img.8.jpg     0.549730         1
31   img.9.jpg     0.300870         0
```