In [9]:

```
path="images/"
```

In [10]:

```
import os
folders = os.listdir(path)
print(os.listdir(path))
```

```
['cats', 'dogs', 'horses', 'humans']
```

In [11]:

```
for f in folders:
    print(f+" "+str(len(os.listdir(path+f))))
```

```
cats 202
dogs 202
horses 202
humans 202
```

In [12]:

```
from keras.preprocessing import image
import matplotlib.pyplot as plt
import shutil
```

In [13]:

```
sample_path = path+"cats/cat.1.jpg"
img = image.load_img(sample_path) # Of Type PIL
x = image.img_to_array(img)/255.0 # Of Type Numpy
plt.imshow(x)
plt.axis("off")
plt.show()
```

In [14]:

```
from keras.layers import *
from keras.models import Sequential
from keras.datasets import mnist
from keras.utils import to_categorical
```

In [15]:

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation='relu',input_shape=(150,150,3)))
model.add(MaxPool2D((2,2)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPool2D((2,2)))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPool2D((2,2)))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPool2D((2,2)))
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dense(4,activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       73856     
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 128)       0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 15, 15, 128)       147584    
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 128)         0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 6272)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                401472    
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 260       
=================================================================
Total params: 642,564
Trainable params: 642,564
Non-trainable params: 0
_________________________________________________________________
```

In [16]:

```
# Compiling Model
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
adam = optimizers.adam(lr=1e-4)
model.compile(optimizer=adam,loss="categorical_crossentropy",metrics=["accuracy"])
```

In [17]:

```
# Making Validation Folder and moving some pics from original folders to validation fo
if not os.path.isdir("val_images"):
    os.mkdir("val_images")
classes = ["dogs","cats","horses","humans"]
```

In [18]:

```
for c in classes:
    p = os.path.join("val_images",c)
    print(p)
    print(type(p))
    if not os.path.exists(p):
        os.mkdir(p)

val_split = 0.1
for folder in os.listdir("images"):
    path = "images/"+folder
    images = os.listdir(path)

    split_size = int(val_split*len(images))
    files_to_move = images[:split_size]
    print(len(files_to_move))

    for img_f in files_to_move:
        src = os.path.join(path,img_f)
        dest = os.path.join("val_images/"+folder,img_f)
        shutil.move(src,dest)
        print(src)
        print(dest)
```

```
val_images\dogs
<class 'str'>
val_images\cats
<class 'str'>
val_images\horses
<class 'str'>
val_images\humans
<class 'str'>
20
images/cats\cat.1.jpg
val_images/cats\cat.1.jpg
images/cats\cat.10.jpg
val_images/cats\cat.10.jpg
images/cats\cat.100.jpg
val_images/cats\cat.100.jpg
images/cats\cat.101.jpg
val_images/cats\cat.101.jpg
images/cats\cat.102.jpg
val_images/cats\cat.102.jpg
images/cats\cat.103.jpg
val_images/cats\cat.103.jpg
images/cats\cat.104.jpg
val_images/cats\cat.104.jpg
images/cats\cat.105.jpg
```

```
val_images/cats\cat.105.jpg
images/cats\cat.106.jpg
val_images/cats\cat.106.jpg
images/cats\cat.107.jpg
val_images/cats\cat.107.jpg
images/cats\cat.108.jpg
val_images/cats\cat.108.jpg
images/cats\cat.109.jpg
val_images/cats\cat.109.jpg
images/cats\cat.11.jpg
val_images/cats\cat.11.jpg
images/cats\cat.110.jpg
val_images/cats\cat.110.jpg
images/cats\cat.111.jpg
val_images/cats\cat.111.jpg
images/cats\cat.112.jpg
val_images/cats\cat.112.jpg
images/cats\cat.113.jpg
val_images/cats\cat.113.jpg
images/cats\cat.114.jpg
val_images/cats\cat.114.jpg
images/cats\cat.115.jpg
val_images/cats\cat.115.jpg
images/cats\cat.116.jpg
val_images/cats\cat.116.jpg
20
images/dogs\dog.1.jpg
val_images/dogs\dog.1.jpg
images/dogs\dog.10.jpg
val_images/dogs\dog.10.jpg
images/dogs\dog.100.jpg
val_images/dogs\dog.100.jpg
images/dogs\dog.101.jpg
val_images/dogs\dog.101.jpg
images/dogs\dog.102.jpg
val_images/dogs\dog.102.jpg
images/dogs\dog.103.jpg
val_images/dogs\dog.103.jpg
images/dogs\dog.104.jpg
val_images/dogs\dog.104.jpg
images/dogs\dog.105.jpg
val_images/dogs\dog.105.jpg
images/dogs\dog.106.jpg
val_images/dogs\dog.106.jpg
images/dogs\dog.107.jpg
val_images/dogs\dog.107.jpg
images/dogs\dog.108.jpg
val_images/dogs\dog.108.jpg
images/dogs\dog.109.jpg
val_images/dogs\dog.109.jpg
images/dogs\dog.11.jpg
val_images/dogs\dog.11.jpg
images/dogs\dog.110.jpg
val_images/dogs\dog.110.jpg
images/dogs\dog.111.jpg
val_images/dogs\dog.111.jpg
images/dogs\dog.112.jpg
val_images/dogs\dog.112.jpg
images/dogs\dog.113.jpg
val_images/dogs\dog.113.jpg
images/dogs\dog.114.jpg
```

```
val_images/dogs\dog.114.jpg
images/dogs\dog.115.jpg
val_images/dogs\dog.115.jpg
images/dogs\dog.116.jpg
val_images/dogs\dog.116.jpg
20
images/horses\horse-1.jpg
val_images/horses\horse-1.jpg
images/horses\horse-10.jpg
val_images/horses\horse-10.jpg
images/horses\horse-100.jpg
val_images/horses\horse-100.jpg
images/horses\horse-101.jpg
val_images/horses\horse-101.jpg
images/horses\horse-102.jpg
val_images/horses\horse-102.jpg
images/horses\horse-103.jpg
val_images/horses\horse-103.jpg
images/horses\horse-104.jpg
val_images/horses\horse-104.jpg
images/horses\horse-105.jpg
val_images/horses\horse-105.jpg
images/horses\horse-106.jpg
val_images/horses\horse-106.jpg
images/horses\horse-107.jpg
val_images/horses\horse-107.jpg
images/horses\horse-108.jpg
val_images/horses\horse-108.jpg
images/horses\horse-109.jpg
val_images/horses\horse-109.jpg
images/horses\horse-11.jpg
val_images/horses\horse-11.jpg
images/horses\horse-110.jpg
val_images/horses\horse-110.jpg
images/horses\horse-111.jpg
val_images/horses\horse-111.jpg
images/horses\horse-112.jpg
val_images/horses\horse-112.jpg
images/horses\horse-113.jpg
val_images/horses\horse-113.jpg
images/horses\horse-114.jpg
val_images/horses\horse-114.jpg
images/horses\horse-115.jpg
val_images/horses\horse-115.jpg
images/horses\horse-116.jpg
val_images/horses\horse-116.jpg
20
images/humans\rider-1.jpg
val_images/humans\rider-1.jpg
images/humans\rider-10.jpg
val_images/humans\rider-10.jpg
images/humans\rider-100.jpg
val_images/humans\rider-100.jpg
images/humans\rider-101.jpg
val_images/humans\rider-101.jpg
images/humans\rider-102.jpg
val_images/humans\rider-102.jpg
images/humans\rider-103.jpg
val_images/humans\rider-103.jpg
images/humans\rider-104.jpg
val_images/humans\rider-104.jpg
```

```
images/humans\rider-105.jpg
val_images/humans\rider-105.jpg
images/humans\rider-106.jpg
val_images/humans\rider-106.jpg
images/humans\rider-107.jpg
val_images/humans\rider-107.jpg
images/humans\rider-108.jpg
val_images/humans\rider-108.jpg
images/humans\rider-109.jpg
val_images/humans\rider-109.jpg
images/humans\rider-11.jpg
val_images/humans\rider-11.jpg
images/humans\rider-110.jpg
val_images/humans\rider-110.jpg
images/humans\rider-111.jpg
val_images/humans\rider-111.jpg
images/humans\rider-112.jpg
val_images/humans\rider-112.jpg
images/humans\rider-113.jpg
val_images/humans\rider-113.jpg
images/humans\rider-114.jpg
val_images/humans\rider-114.jpg
images/humans\rider-115.jpg
val_images/humans\rider-115.jpg
images/humans\rider-116.jpg
val_images/humans\rider-116.jpg
```

In [19]:

```
print("Training Data")
for f in folders:
    print(f+" "+str(len(os.listdir("images/"+f))))

print("\nValidation Data")
for f in folders:
    print(f+" "+str(len(os.listdir("val_images/"+f))))
```

```
Training Data
cats 182
dogs 182
horses 182
humans 182

Validation Data
cats 20
dogs 20
horses 20
humans 20
```

In [20]:

```
# model.fit(...) useful when data is small and it can fit inside the memory

# Fitting model using Generators
train_gen = ImageDataGenerator(rescale=1.0/255.0)
val_gen = ImageDataGenerator(rescale=1.0/255.0)


train_generator = train_gen.flow_from_directory(
    "images/",
    target_size=(150,150),
    batch_size=32,
    class_mode='categorical'
)

val_generator = val_gen.flow_from_directory(
    "val_images/",
    target_size=(150,150),
    batch_size=32,
    class_mode='categorical'
)

# train_generator.next() gives tuple with x and y values
x,y = train_generator.next()
print(x.shape)
print(type(x))
print(y.shape)
print(type(y))
```

```
Found 728 images belonging to 4 classes.
Found 80 images belonging to 4 classes.
(32, 150, 150, 3)
<class 'numpy.ndarray'>
(32, 4)
<class 'numpy.ndarray'>
```

In [21]:

```
train_generator.labels
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3])
```

```
In [22]:

train_generator.next()
```

```
          ...,
          [0.9960785 , 0.9960785 , 0.9960785 ],
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ]],

         ...,

         [[0.49803925, 0.4666667 , 0.27450982],
          [0.41960788, 0.3921569 , 0.18039216],
          [0.3921569 , 0.36862746, 0.14117648],
          ...,
          [0.24705884, 0.25882354, 0.09019608],
          [0.34509805, 0.35686275, 0.14901961],
          [0.40784317, 0.41176474, 0.18823531]],

         [[0.43921572, 0.427451  , 0.21176472],
          [0.42352945, 0.41176474, 0.20392159],
          [0.43921572, 0.427451  , 0.21960786],
          ...,
          [0.37647063, 0.3921569 , 0.19215688],
          [0.40784317, 0.41176474, 0.1764706 ],
          [0.6       , 0.6       , 0.34901962]],

         [[0.5529412 , 0.54901963, 0.3254902 ],
          [0.5254902 , 0.52156866, 0.30980393]
```

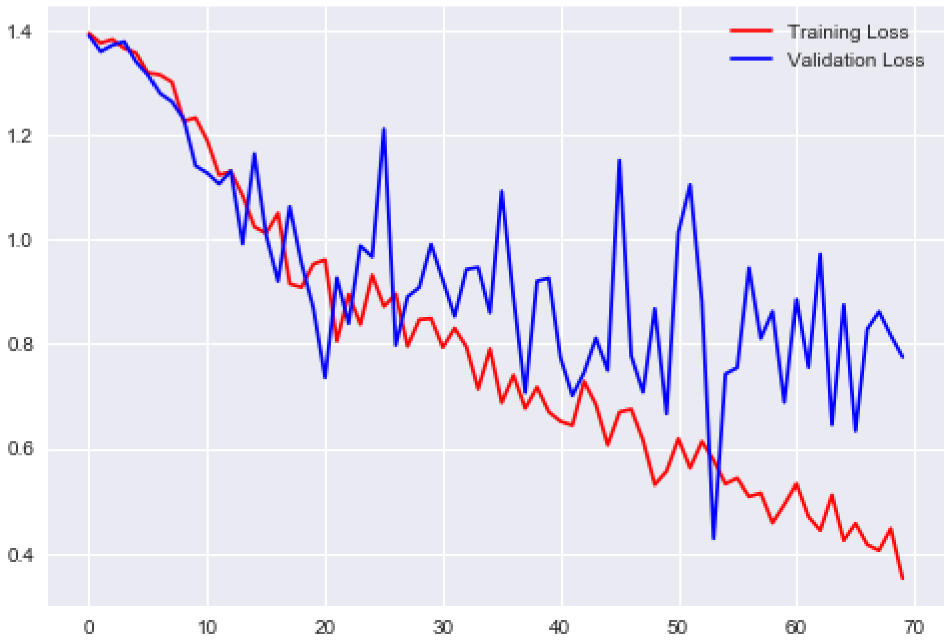
In [23]:

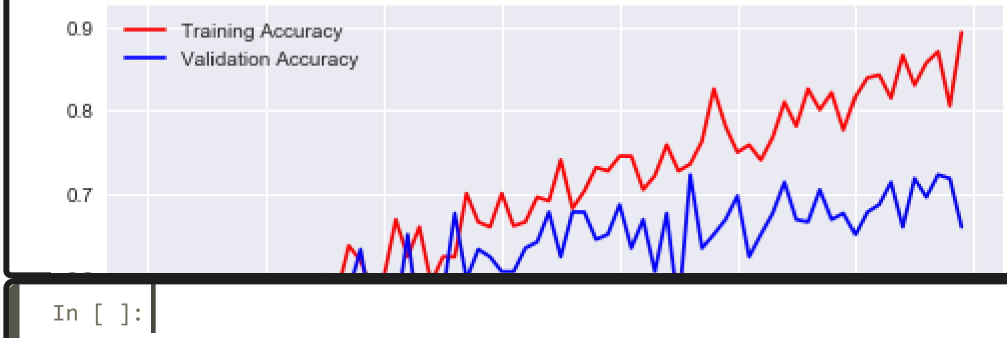
```
hist = model.fit_generator(
    train_generator,
    epochs = 70,
    steps_per_epoch = 7,
    validation_data = val_generator,
    validation_steps=4
)
```

```
7/7 [==============================] - 8s 1s/step - loss: 0.5430 - accuracy: 0.7812 - val_loss: 0.7546 - val_a
Epoch 57/70
7/7 [==============================] - 7s 1s/step - loss: 0.5087 - accuracy: 0.8259 - val_loss: 0.9459 - val_a
Epoch 58/70
7/7 [==============================] - 7s 1s/step - loss: 0.5167 - accuracy: 0.8009 - val_loss: 0.8106 - val_a
Epoch 59/70
7/7 [==============================] - 7s 1s/step - loss: 0.4584 - accuracy: 0.8214 - val_loss: 0.8622 - val_a
Epoch 60/70


7/7 [==============================] - 8s 1s/step - loss: 0.4935 - accuracy: 0.7768 - val_loss: 0.6882 - val_a
Epoch 61/70
7/7 [==============================] - 7s 1s/step - loss: 0.5323 - accuracy: 0.8170 - val_loss: 0.8854 - val_a
Epoch 62/70
7/7 [==============================] - 7s 1s/step - loss: 0.4704 - accuracy: 0.8393 - val_loss: 0.7551 - val_a
Epoch 63/70
7/7 [==============================] - 7s 1s/step - loss: 0.4415 - accuracy: 0.8426 - val_loss: 0.9722 - val_a
Epoch 64/70
7/7 [==============================] - 7s 1s/step - loss: 0.5134 - accuracy: 0.8148 - val_loss: 0.6451 - val_a
Epoch 65/70
7/7 [==============================] - 7s 1s/step - loss: 0.4248 - accuracy: 0.8661 - val_loss: 0.8752 - val_a
Epoch 66/70
7/7 [==============================] - 8s 1s/step - loss: 0.4573 - accuracy: 0.8304 - val_loss: 0.6333 - val_a
Epoch 67/70
7/7 [==============================] - 7s 1s/step - loss: 0.4165 - accuracy: 0.8571 - val_loss: 0.8285 - val_a
```

In [24]:

```
h = hist.history
import matplotlib.pyplot as plt
plt.style.use('seaborn')


# Visualizing Loss
plt.plot(h['loss'],'r',label="Training Loss")
plt.plot(h['val_loss'],'b',label="Validation Loss")
plt.legend()
plt.show()

# Visualizing Accuracy
plt.plot(h['accuracy'],'r',label="Training Accuracy")
plt.plot(h['val_accuracy'],'b',label="Validation Accuracy")
plt.legend()
plt.show()
```





```
In [ ]:
```