Neural network are individual nodes that form the layer in the network. Just like how neurons in our brain are connecteed to different areas.

There are typically 3 layers in a Neural Network:

1) Input Layer

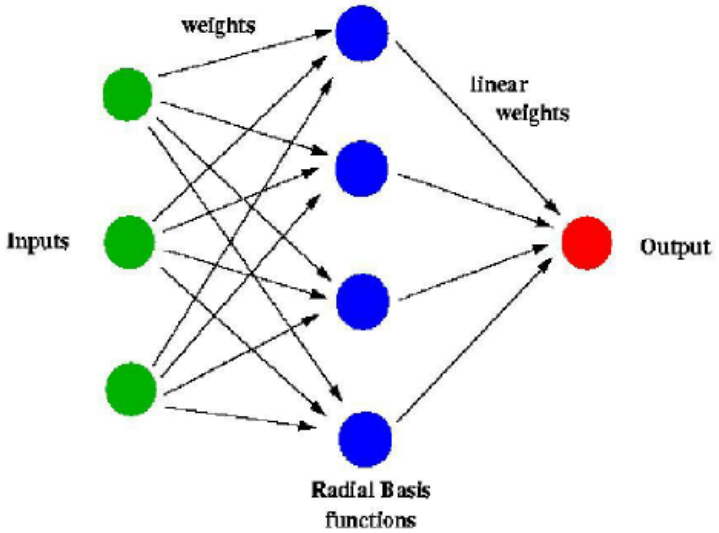
2) Hidden Layer

3) Output Layer

The input layer usually is a single layer, and it will have a weight that is assigned to them that changes the effect on the overall prediction result.

The neural network takes all the training in the input layer, then it passes the data through the hidden layer transforming the values based on the weights at each node.

Then it returns a value in the output layer.



In [ ]:
```
1
```

CNN

A convolutional neural network is a specific kind of neural network with multiple layers. It processes data that has a grid like arrangement and then it extracts important features.

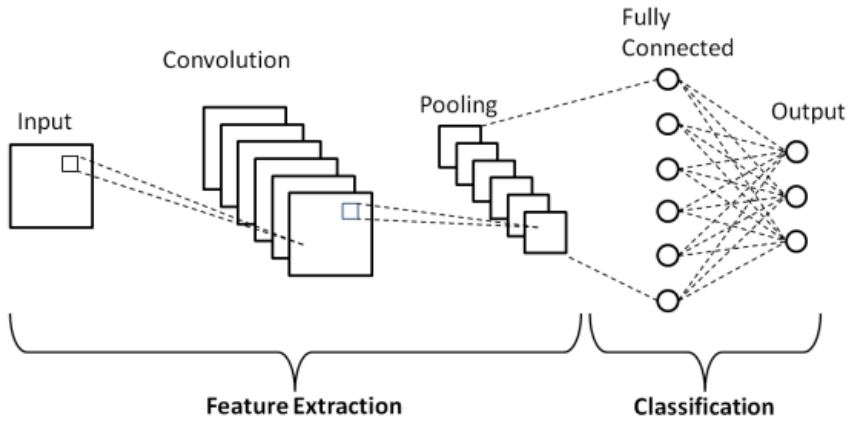
In this type of neural network they take image as an input & learn the various features on the image through filters.

This allows them to learn the important object present in the image.

For example: In CNN it will learn specific features of the cat that differentiate from the dogs. It can easily differntiate between the two.

One of the most important feature of CNN is that it sets apart from other Machine Learning Algorithm its ability to pre process the data by itself.

You dont have to spend lots of time in data pre-processing



In [ ]:
```
1
```

example

In [ ]:
```
1
```

In [1]:
```
import numpy as np
import pandas as pd
from keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import os
import random
```

In [ ]:
```
1
```

In [2]:
```
# filenames[0].split('.')[0]
```

In [3]:
```
# reading the data and storing it into a Dataframe
```

In [4]:
```
filenames = os.listdir('/home/punit/Downloads/Python-Batch/PG-DS/train')
```

In [5]:
```
for val in filenames:
    print(val)
```

```
cat.12371.jpg
dog.6189.jpg
cat.1208.jpg
dog.4534.jpg
cat.21.jpg
dog.11973.jpg
dog.10250.jpg
dog.8844.jpg
dog.6560.jpg
cat.2362.jpg
cat.1426.jpg
dog.11578.jpg
cat.8728.jpg
dog.763.jpg
cat.9261.jpg
dog.8855.jpg
cat.2495.jpg
cat.415.jpg
dog.3379.jpg
```

In [6]:
```
categories = []
for file_name in filenames:
    category = file_name.split('.')[0]
#     print(category)
    if category == 'dog':
        categories.append(1)
    else:
        categories.append(0)
```

In [7]:
```
print(categories)
```

```
[0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1,
1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1,
1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0,
0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1,
0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1,
0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
```

In [8]:
```
df = pd.DataFrame({
    'filename':filenames,
    'category':categories
})
```

In [9]:
```
df.head()
```

Out[9]:

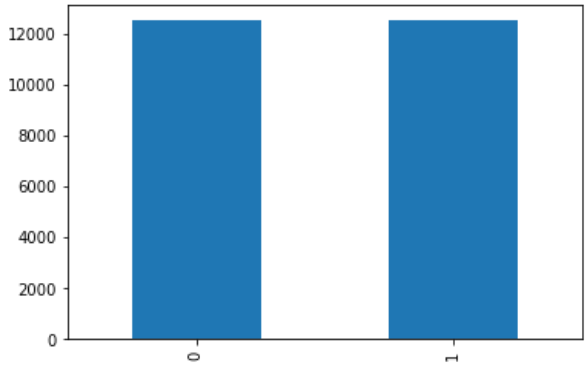
	filename	category
0	cat.12371.jpg	0
1	dog.6189.jpg	1
2	cat.1208.jpg	0
3	dog.4534.jpg	1
4	cat.21.jpg	0

In [ ]:
```

```

In [10]:
```
df['category'].value_counts().plot.bar()
```

Out[10]: <AxesSubplot:>



In [ ]:
```

```

In [11]:
```
sample = random.choice(filenames)
```

In [12]:
```
sample
```

Out[12]: 'cat.1429.jpg'

In [13]:
```
img = load_img('train/'+sample)
```

In [14]:
```
1 plt.imshow(img)
```

Out[14]: <matplotlib.image.AxesImage at 0x7fe2d856da90>

In [15]:
```
1 plt.imshow(img)
```

Out[15]: <matplotlib.image.AxesImage at 0x7fe2d84d0be0>

In [ ]:
```
1
```

Building the Model

Input Layer: represent input image data. we will reshape image into single dimension array.

for instance: 64 * 64 (px) image will be converted into 4096 ----> which will be in array (4096,1) array

Convolutional Layer : This layer will extract features from the image.

Pooling Layer : Will take common feature in pool / also it will reduce spatial volume of input image after convolution.

Fully Connected Layer: This will connect the network from one layer to another.

Output Layer: It is the predicted value of the layer.

In [16]:
```
IMAGE_WIDTH = 128
IMAGE_HEIGHT = 128
IMAGE_SIZE = (IMAGE_WIDTH,IMAGE_HEIGHT)
IMAGE_CHANNELS = 3

from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalization
model = Sequential()

model.add(Conv2D(32,(3,3),activation='relu',input_shape=(IMAGE_WIDTH,IMAGE_HEIGHT,IMAGE_CHANNELS)))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))
```

In [ ]:
```

```

Batch Normalization: It normalizes or applies a transformation that maintains the mean output close to 0 and the output of the standard deviation close to 1. It typically works during the training of the data.

Drop-out: it randomly sets the input to 0, which helps in preventing overfitting of the Data

In [ ]:
```

```

In [17]:
```
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

In [18]:
```
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

In [19]:

```
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(2,activation='softmax')) #the output data should be in 0,1 (cat / dog)


model.compile(loss='categorical_crossentropy',optimizer='rmsprop',metrics=['accuracy'])


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 batch_normalization (BatchN  (None, 126, 126, 32)     128
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 dropout (Dropout)           (None, 63, 63, 32)        0

 conv2d_1 (Conv2D)           (None, 61, 61, 64)        18496

 batch_normalization_1 (Batc  (None, 61, 61, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 30, 30, 64)        0

 conv2d_2 (Conv2D)           (None, 28, 28, 128)       73856

 batch_normalization_2 (Batc  (None, 28, 28, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 14, 14, 128)      0
 2D)

 dropout_2 (Dropout)         (None, 14, 14, 128)       0

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 batch_normalization_3 (Batc  (None, 512)              2048
 hNormalization)

 dropout_3 (Dropout)         (None, 512)               0
```

```
 dense_1 (Dense)              (None, 2)                 1026

=================================================================
Total params: 12,942,786
Trainable params: 12,941,314
Non-trainable params: 1,472
_________________________________________________________________
```

In [20]:
```
1 IMAGE_SIZE
```

Out[20]: (128, 128)

In [ ]:
```
1
```

In [21]:
```
1 df.head()
```

Out[21]:

	filename	category
0	cat.12371.jpg	0
1	dog.6189.jpg	1
2	cat.1208.jpg	0
3	dog.4534.jpg	1
4	cat.21.jpg	0

In [22]:
```
1 df['category'] = df['category'].replace({0:'cat',1:'dog'})
```

In [23]:
```
1 df
```

Out[23]:

	filename	category
0	cat.12371.jpg	cat
1	dog.6189.jpg	dog
2	cat.1208.jpg	cat
3	dog.4534.jpg	dog
4	cat.21.jpg	cat
...	...	...
24995	dog.9568.jpg	dog
24996	dog.1219.jpg	dog
24997	dog.12366.jpg	dog
24998	dog.7367.jpg	dog
24999	cat.3781.jpg	cat

25000 rows × 2 columns

In [24]:
```
1 train_df, validate_df = train_test_split(df,test_size=0.20,random_state=42)
2 train_df = train_df.reset_index(drop=True)
```

In [25]:
```
1 train_df.head()
```

Out[25]:

	filename	category
0	cat.8381.jpg	cat
1	cat.8034.jpg	cat
2	dog.6311.jpg	dog
3	cat.6820.jpg	cat
4	dog.8439.jpg	dog

In [26]:
```
1 validate_df = validate_df.reset_index(drop=True)
```

In [27]:
```
1 validate_df.head()
```

Out[27]:

	filename	category
0	cat.3451.jpg	cat
1	cat.6768.jpg	cat
2	dog.10764.jpg	dog
3	cat.4895.jpg	cat
4	dog.9377.jpg	dog

In [28]:
```
1 len(train_df)
```

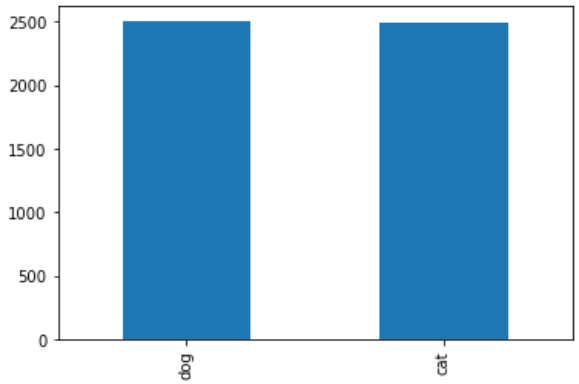
Out[28]: 20000

In [29]:
```
1 len(validate_df)
```

Out[29]: 5000

In [30]:
```
1 validate_df['category'].value_counts().plot.bar()
```

Out[30]: <AxesSubplot:>



In [31]:
```
1 validate_df['category'].value_counts()
```

Out[31]:
```
dog    2505
cat    2495
Name: category, dtype: int64
```

In [32]:
```
1 train_df['category'].value_counts()
```

Out[32]:
```
cat    10005
dog     9995
Name: category, dtype: int64
```

In [33]:
```
total_train = train_df.shape[0]
```

In [34]:
```
total_train
```

Out[34]: 20000

In [35]:
```
total_validate = validate_df.shape[0]
```

In [36]:
```
total_validate
```

Out[36]: 5000

In [37]:
```
batch_size = 15
```

In [ ]:
```

```

In [ ]:
```

```

Training generator

In [38]:
```
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
    )
```

In [39]:
```
train_generator = train_datagen.flow_from_dataframe(train_df,
                                                    'train/',
                                                    x_col='filename',
                                                    y_col='category',
                                                    target_size=IMAGE_SIZE,
                                                    class_mode='categorical',
                                                    batch_size=batch_size
                                                   )
```

```
Found 20000 validated image filenames belonging to 2 classes.
```

In [40]:
```
validation_datagen = ImageDataGenerator(
    rescale=1./255)
```

In [41]:
```
validation_generator = validation_datagen.flow_from_dataframe(validate_df,
                                                              'train/',
                                                              x_col='filename',
                                                              y_col='category',
                                                              target_size=IMAGE_SIZE,
                                                              class_mode='categorical',
                                                              batch_size=batch_size
                                                             )
```

```
Found 5000 validated image filenames belonging to 2 classes.
```

In [ ]:
```

```

Example Generator

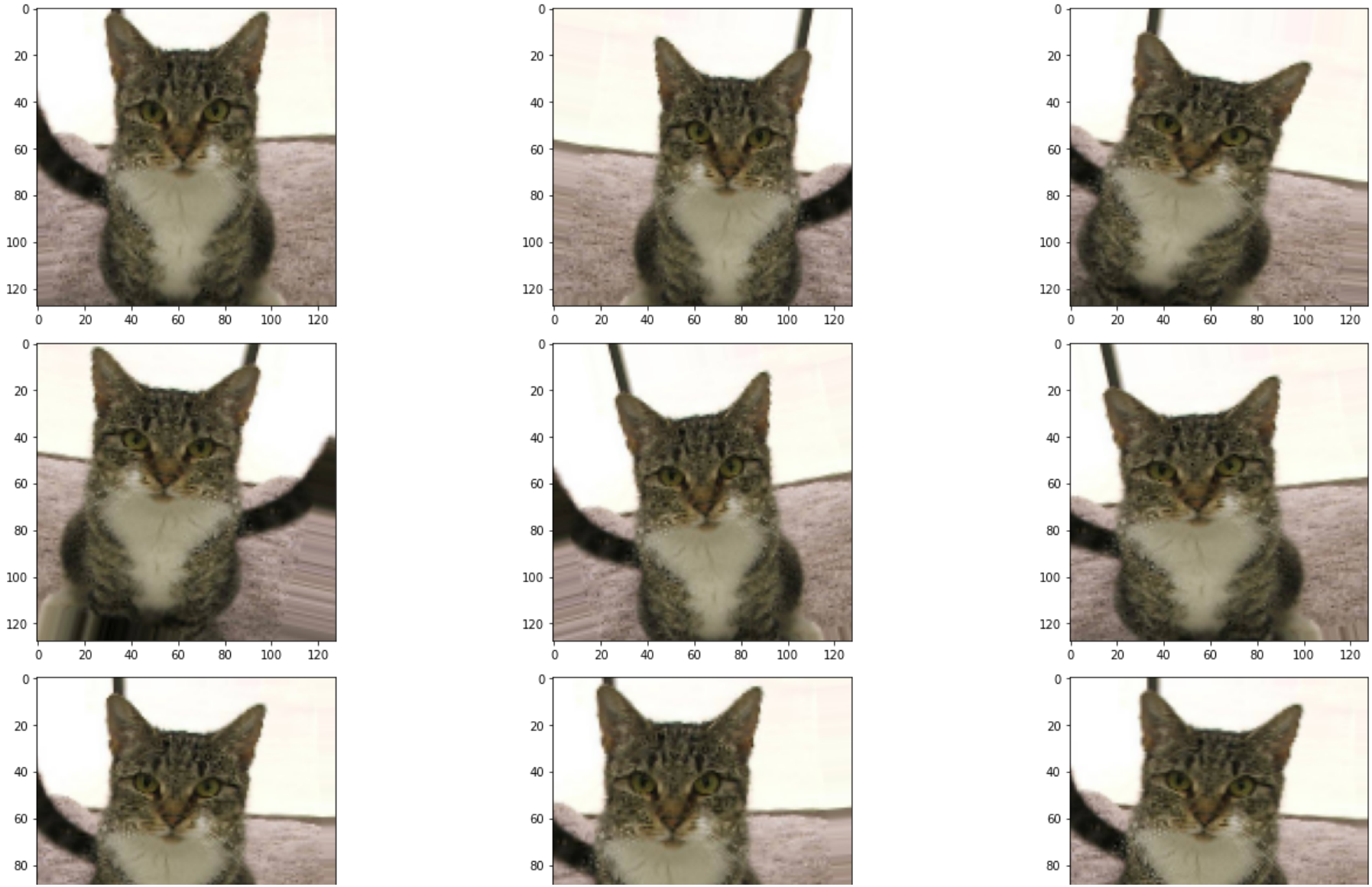
In [43]:
```
example_df = train_df.sample(n=1).reset_index(drop=True)
```

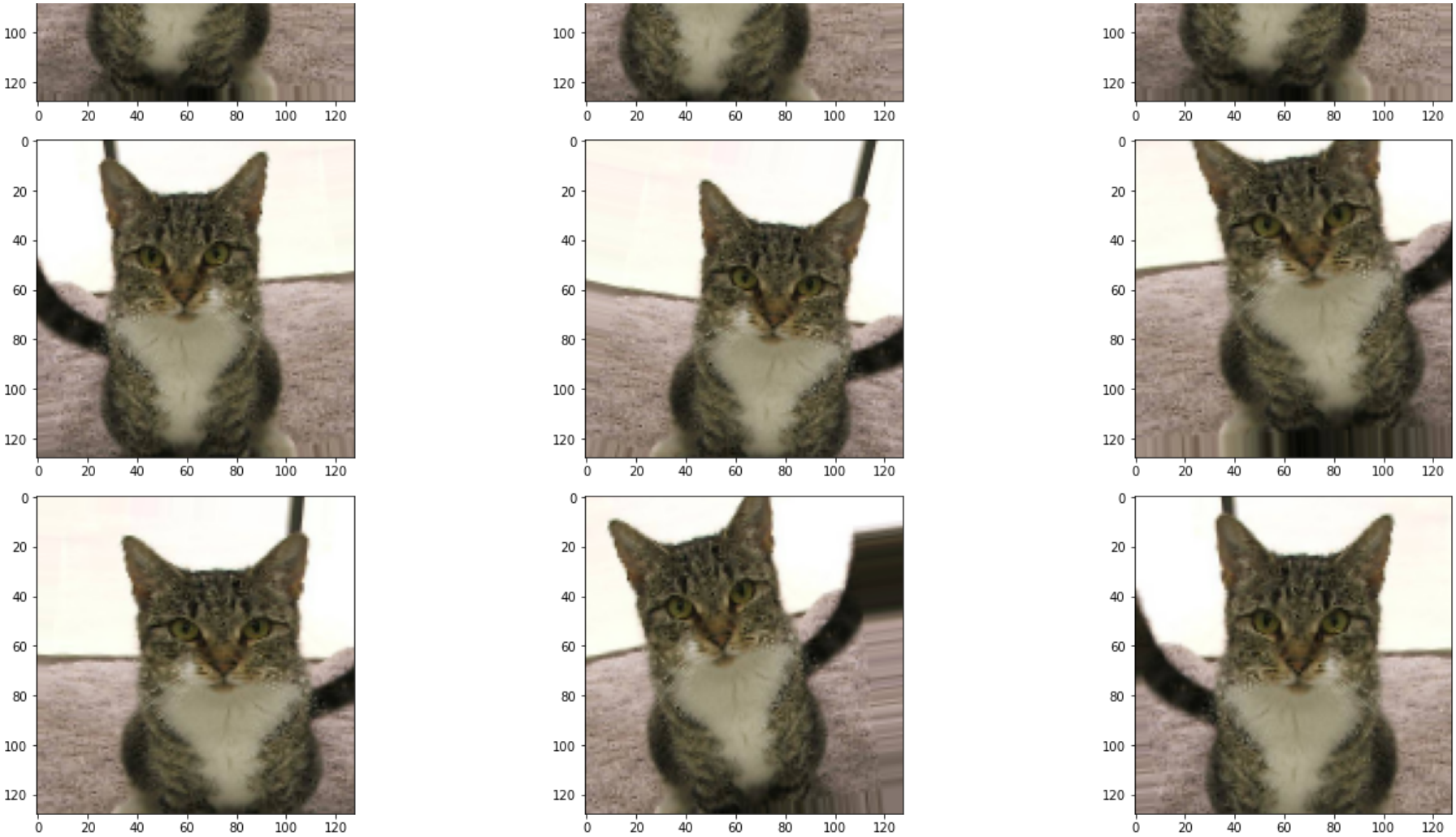
In [44]:
```
example_generator = train_datagen.flow_from_dataframe(example_df,'train/',x_col='filename',y_col='category',
                                                      target_size=IMAGE_SIZE,class_mode='categorical')
```

```
Found 1 validated image filenames belonging to 1 classes.
```

In [45]:

```
plt.figure(figsize=(20,20))
for i in range(0,15):
    plt.subplot(5,3,i+1)
    for X_batch,Y_batch in example_generator:
        image = X_batch[0]
        plt.imshow(image)
        break
plt.tight_layout()
plt.show()
```





In [50]:
```
1 total_train // batch_size
```

Out[50]: 1333

In [47]:
```
history = model.fit_generator(
    train_generator,
    epochs=3,
    validation_data= validation_generator,
    validation_steps= total_validate // batch_size,
    steps_per_epoch = total_train // batch_size,
)
```

```
<ipython-input-47-0e7355e3de92>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(

Epoch 1/3
1333/1333 [==============================] - 641s 480ms/step - loss: 0.7618 - accuracy: 0.6334 - val_loss: 0.5757 - val_accuracy: 0.7041
Epoch 2/3
1333/1333 [==============================] - 628s 471ms/step - loss: 0.5505 - accuracy: 0.7272 - val_loss: 0.5915 - val_accuracy: 0.7369
Epoch 3/3
1333/1333 [==============================] - 639s 479ms/step - loss: 0.4968 - accuracy: 0.7665 - val_loss: 0.4697 - val_accuracy: 0.7880
```

In [81]:
```

```

Out[81]:
```
[0.6334250569343567, 0.7272454500198364, 0.7665249109268188]
```

Visualize Training

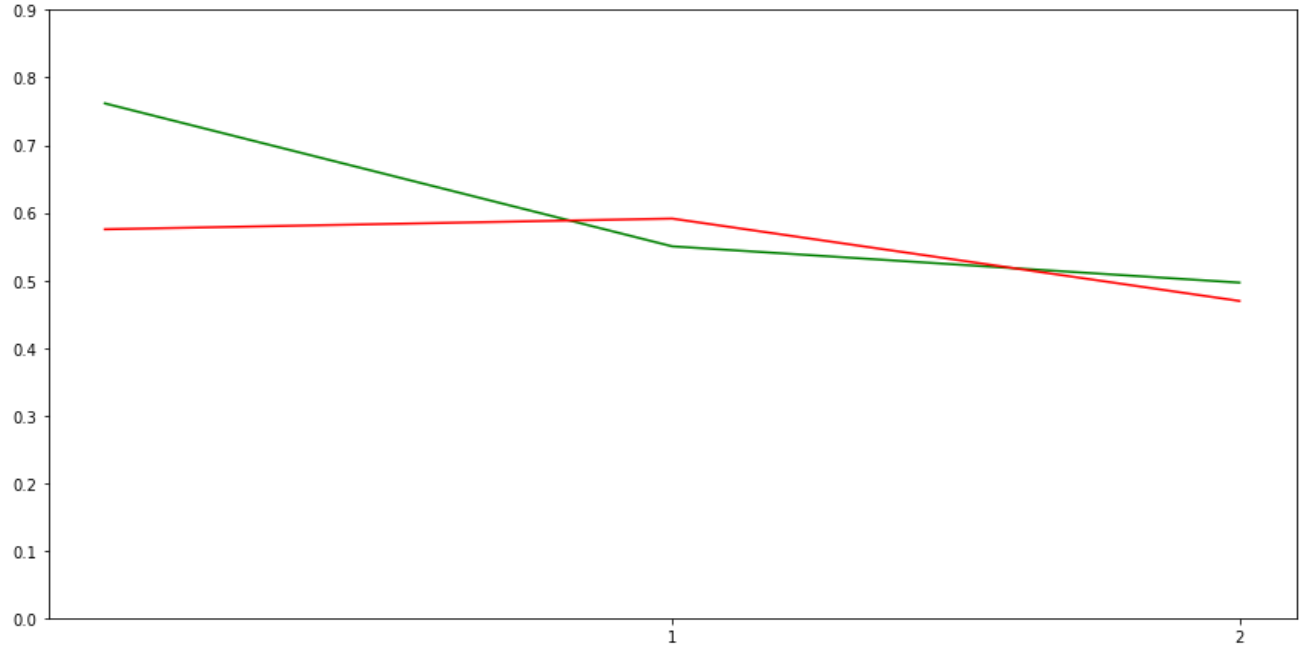
In [78]:

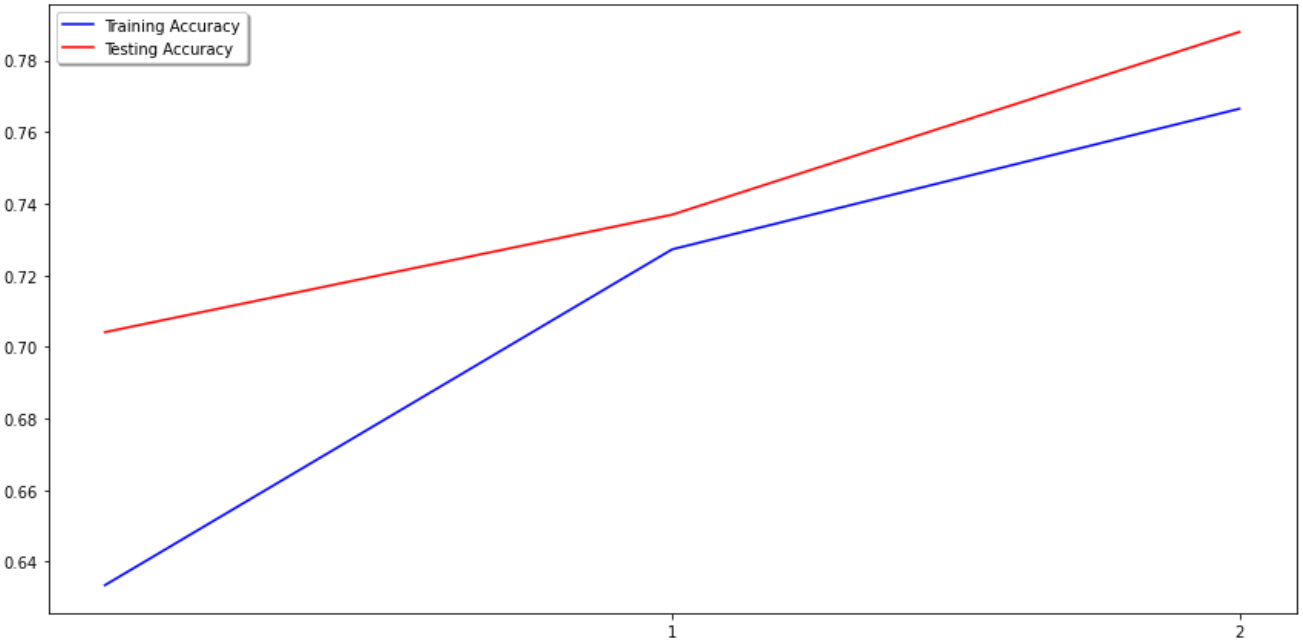
```
fig,(ax1,ax2) = plt.subplots(2,1,figsize=(12,12))

ax1.plot(history.history['loss'],color='g',label='Training Data Loss')
ax1.plot(history.history['val_loss'],color='r',label='Testing Data Loss')
ax1.set_xticks(np.arange(1,3,1))
ax1.set_yticks(np.arange(0,1,0.1))

ax2.plot(history.history['accuracy'],color='b',label='Training Accuracy')
ax2.plot(history.history['val_accuracy'],color='r',label='Testing Accuracy')
ax2.set_xticks(np.arange(1,3,1))

legend = plt.legend(loc='best',shadow=True)
plt.tight_layout()
plt.show()
```


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1
2


Training Accuracy
Testing Accuracy
0.78
0.76
0.74
0.72
0.70
0.68
0.66
0.64
1
2

In [ ]:
```
1
```

In [55]:
```
1 test_file = os.listdir('test1/')
```

In [ ]:
```
1
```

In [57]:
```
1 test_df = pd.DataFrame({
2     'filename':test_file
3 })
```

In [61]:
```
1 test_df
```

Out[61]:

	filename
0	9895.jpg
1	11342.jpg
2	6449.jpg
3	9662.jpg
4	7947.jpg
...	...
12495	3796.jpg
12496	10089.jpg
12497	3565.jpg
12498	10366.jpg
12499	5025.jpg

12500 rows × 1 columns

In [58]:
```
1 nb_sample = test_df.shape[0]
```

In [59]:
```
1 nb_sample
```

Out[59]:
```
12500
```

In [84]:
```
1 round(nb_sample / batch_size)
```

Out[84]:
```
833
```

In [62]:
```
test_gen = ImageDataGenerator(rescale=1./255)
test_generator = test_gen.flow_from_dataframe(test_df,
                                              'test1/',
                                              x_col='filename',
                                              y_col=None,
                                              class_mode=None,
                                              target_size=IMAGE_SIZE,
                                              batch_size=batch_size,
                                              shuffle=False)
```

```
Found 12500 validated image filenames.
```

In [65]:
```
predict = model.predict_generator(test_generator,steps=np.ceil(nb_sample/batch_size))
```

```
<ipython-input-65-46522c80dc03>:1: UserWarning: `Model.predict_generator` is deprecated and will be removed in a future version. Please use `Mode
l.predict`, which supports generators.
  predict = model.predict_generator(test_generator,steps=np.ceil(nb_sample/batch_size))
```

In [85]:
```
predict
```

Out[85]:
```
array([[0.38720885, 0.6127911 ],
       [0.04973018, 0.9502699 ],
       [0.03544782, 0.96455216],
       ...,
       [0.9778145 , 0.0221855 ],
       [0.04795221, 0.95204777],
       [0.69287425, 0.30712578]], dtype=float32)
```

In [67]:
```
test_df['category'] = np.argmax(predict,axis=-1)
```

In [68]:
```
1 test_df
```

Out[68]:

	filename	category
0	9895.jpg	1
1	11342.jpg	1
2	6449.jpg	1
3	9662.jpg	0
4	7947.jpg	1
...	...	...
12495	3796.jpg	1
12496	10089.jpg	0
12497	3565.jpg	0
12498	10366.jpg	1
12499	5025.jpg	0

12500 rows × 2 columns

In [87]:
```
1 for k,v in train_generator.class_indices.items():
2     print(k,v)
```

```
cat 0
dog 1
```

In [69]:
```
1 label_map = dict((val,key) for key,val in train_generator.class_indices.items())
```

In [73]:
```
1 label_map
```

Out[73]:
```
{0: 'cat', 1: 'dog'}
```

In [71]:
```
1 test_df['category'] = test_df['category'].replace(label_map)
```

In [72]:
```
1 test_df
```

Out[72]:

	filename	category
0	9895.jpg	dog
1	11342.jpg	dog
2	6449.jpg	dog
3	9662.jpg	cat
4	7947.jpg	dog
...	...	...
12495	3796.jpg	dog
12496	10089.jpg	cat
12497	3565.jpg	cat
12498	10366.jpg	dog
12499	5025.jpg	cat

12500 rows × 2 columns

In [74]:
```
1 test_df.head(20)
```

Out[74]:

	filename	category
0	9895.jpg	dog
1	11342.jpg	dog
2	6449.jpg	dog
3	9662.jpg	cat
4	7947.jpg	dog
5	6217.jpg	dog
6	2611.jpg	cat
7	5147.jpg	dog
8	7145.jpg	dog
9	4591.jpg	cat
10	2661.jpg	dog
11	12120.jpg	dog
12	2328.jpg	cat
13	9788.jpg	dog
14	11114.jpg	dog
15	5964.jpg	dog
16	4201.jpg	cat
17	5911.jpg	dog
18	11524.jpg	dog
19	3518.jpg	cat

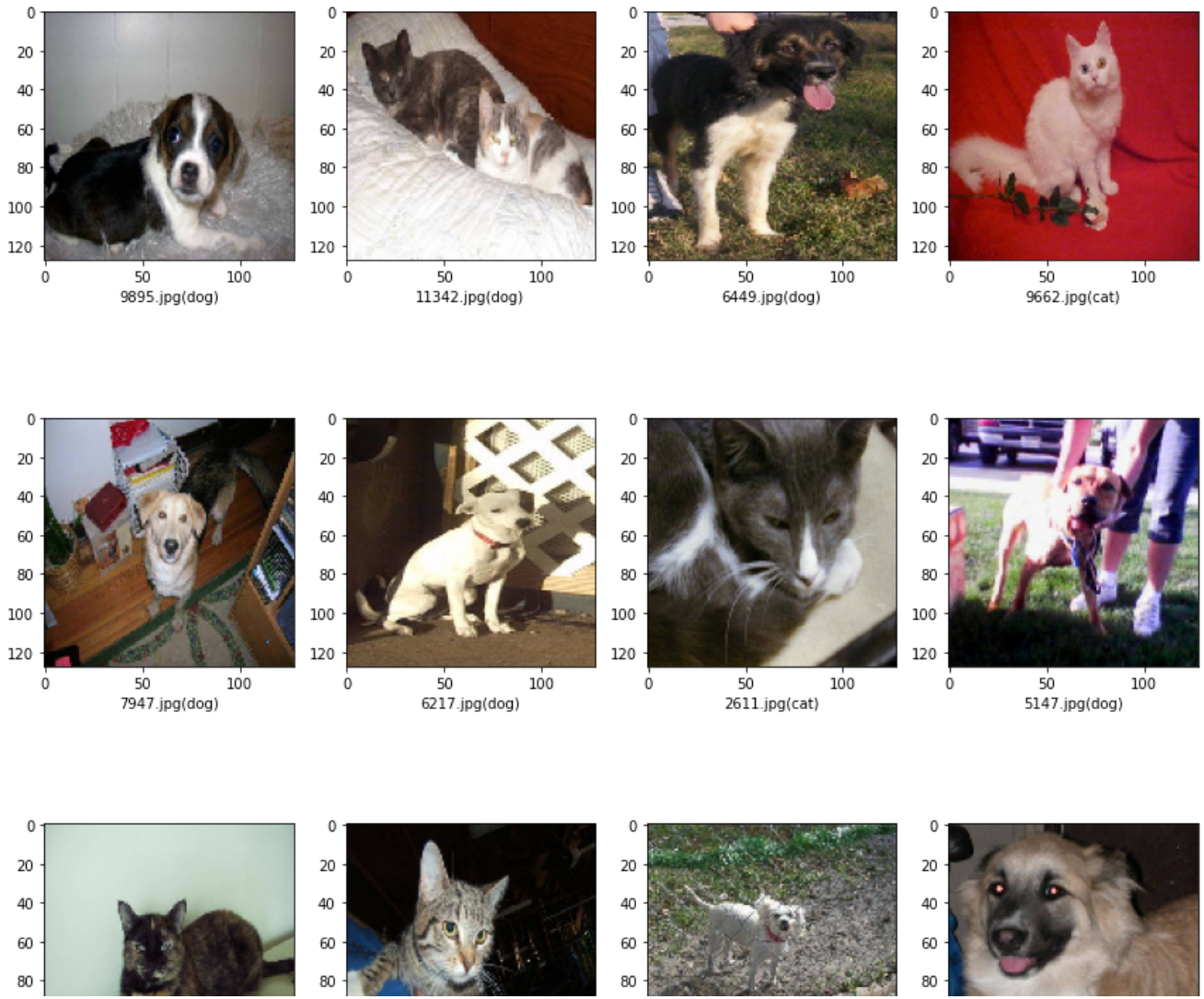
In [88]:
```
1 sample_data
```

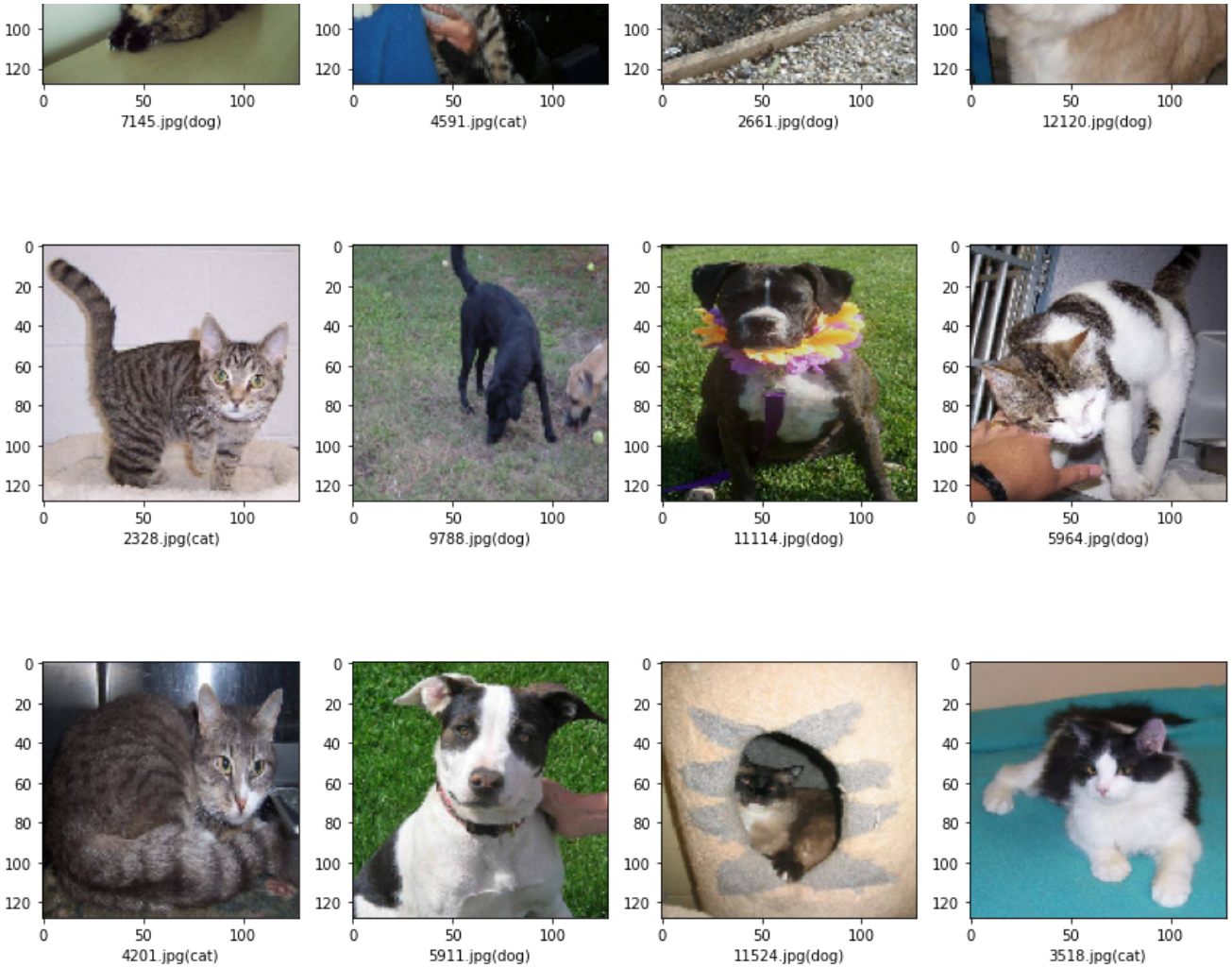
Out[88]:

	filename	category
0	9895.jpg	dog
1	11342.jpg	dog
2	6449.jpg	dog
3	9662.jpg	cat
4	7947.jpg	dog
5	6217.jpg	dog
6	2611.jpg	cat
7	5147.jpg	dog
8	7145.jpg	dog
9	4591.jpg	cat
10	2661.jpg	dog
11	12120.jpg	dog
12	2328.jpg	cat
13	9788.jpg	dog
14	11114.jpg	dog
15	5964.jpg	dog
16	4201.jpg	cat
17	5911.jpg	dog
18	11524.jpg	dog
19	3518.jpg	cat

In [77]:

```
sample_data = test_df.head(20)
sample_data.head()
plt.figure(figsize=(12,24))
for index,row in sample_data.iterrows():
    filename = row['filename']
    category = row['category']
    img = load_img('test1/'+filename,target_size=IMAGE_SIZE)
    plt.subplot(6,4,index+1)
    plt.imshow(img)
    plt.xlabel(filename+'('+"{}".format(category)+')')
plt.tight_layout()
plt.show()
```




7145.jpg(dog)
4591.jpg(cat)
2661.jpg(dog)
12120.jpg(dog)
2328.jpg(cat)
9788.jpg(dog)
11114.jpg(dog)
5964.jpg(dog)
4201.jpg(cat)
5911.jpg(dog)
11524.jpg(dog)
3518.jpg(cat)

In [ ]:
```
1
```