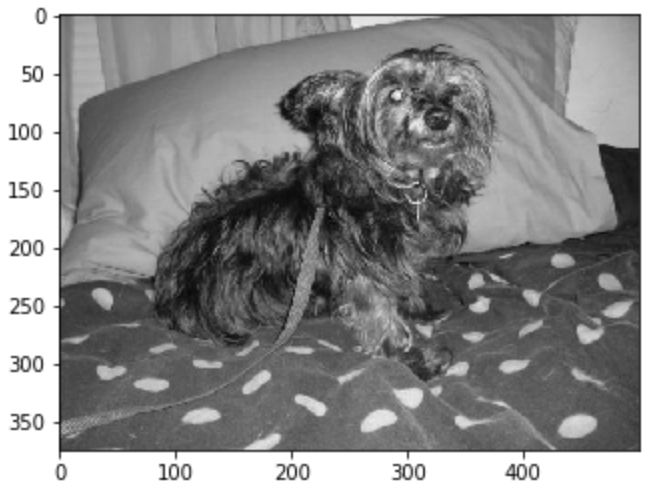
In [1]:
```
# Importing the libraries into python
import numpy as np  # To do various array operations
import pandas as pd
import matplotlib.pyplot as plt  # Used just to show the image
import os    # To iterate through directories and paths
import cv2   # To do some image operations
import time
import random # To do some random shuffling
import pickle # To save the data
import tensorflow as tf
```

In [2]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPool
ing2D
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import TensorBoard
```

In [3]:
```
#!pip install opencv-python
#!pip install tensorflow-gpu
```

In [4]:
```
DATADIR = "C:/Users/Evan Jones Boddu/Desktop/SUBJECTS/Projects/Cats and Dogs NN/PetImage
s"
CATEGORIES = ["Dog", "Cat"]
```

In [5]:
```
for category in CATEGORIES:
    path = os.path.join(DATADIR, category) # path to cats or dog dir
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap = "gray")
        plt.show()
        break
    break
```



In [6]:
```
print(img_array)
```

```
[[117 117 119 ... 133 132 132]
 [118 117 119 ... 135 134 134]
 [119 118 120 ... 137 136 136]
 ...
 [ 79  74  73 ...  80  76  73]
 [ 78  72  69 ...  72  73  74]
 [ 74  71  70 ...  75  73  71]]
```

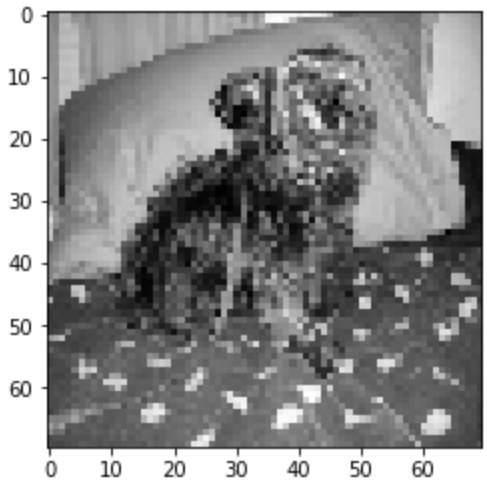
In [7]:
```
print(img_array.shape)
```

```
(375, 500)
```

In [8]:
```
IMG_SIZE = 70

new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
plt.imshow(new_array, cmap = 'gray')
plt.show
```

Out[8]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [9]:
```
training_data = []

def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category) # path to cats or dog dir
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
            except Exception as e:
                pass


create_training_data()
```

In [10]:
```
print(len(training_data))
```

```
24946
```

In [11]:
```
# Shuffling the data randomly for a better sampling

random.shuffle(training_data)
```

In [12]:
```
for sample in training_data[:10]:
    print(sample[1])
```

```
0
0
0
1
0
1
0
1
0
1
```

In [13]:
```
X = []
y = []
```

In [14]:
```
for features, label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
y = np.array(y)
```

In [15]:
```
# Saving the data, i.e x.pickle and y.pickle

pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle", "wb")
pickle.dump(y, pickle_out)
pickle_out.close()
```

In [16]:
```
pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)
```

In [17]:
```
X[1]
```

Out[17]:
```
array([[[141],
        [143],
        [145],
        ...,
        [103],
        [123],
        [109]],

       [[143],
        [145],
        [147],
        ...,
        [103],
        [125],
        [123]],

       [[144],
        [147],
        [150],
        ...,
        [104],
        [127],
        [125]],

       ...,

       [[101],
        [101],
        [106],
        ...,
        [107],
        [105],
        [105]],

       [[ 98],
        [104],
        [107],
        ...,
        [105],
        [103],
        [105]],

       [[ 90],
        [106],
        [100],
        ...,
        [111],
        [104],
        [107]]], dtype=uint8)
```

In [18]:
```
X = X/255.0
```

In [19]:
```
X.shape
```

Out[19]:
```
(24946, 70, 70, 1)
```

In [20]:

```
# Trying with different layers and different parameters

#dense_layers = [0,1,2]
#layer_sizes = [32,64,128]
#conv_layers = [1,2,3]


dense_layers = [0]
layer_sizes = [64]
conv_layers = [3]

for dense_layer in dense_layers:
    for layer_size in layer_sizes:
        for conv_layer in conv_layers:
            NAME = "{}-conv-{}-nodes-{}-dense-{}".format(conv_layer, layer_size,dense_la
yer, int(time.time()))
            #tensorboard = TensorBoard(log_dir='C:\\logs\\{}'.format(NAME))
            tensorboard = TensorBoard(log_dir='C:\\logs\\{}'.format(NAME))
            print(NAME)
            model = Sequential()

            model.add(Conv2D(layer_size, (3,3), input_shape = X.shape[1:]))
            model.add(Activation("relu"))
            model.add(MaxPooling2D(pool_size = (2,2)))

            for l in range(conv_layer-1):
                model.add(Conv2D(layer_size, (3,3)))
                model.add(Activation("relu"))
                model.add(MaxPooling2D(pool_size = (2,2)))

            model.add(Flatten()) # this converts our 3D feature maps to 1D feature vecto
r
            for l in range(dense_layer):
                model.add(Dense(layer_size))
                model.add(Activation("relu"))

            model.add(Dense(1))
            model.add(Activation('sigmoid'))

            model.compile(loss = "binary_crossentropy",
                          optimizer = "adam",
                          metrics = ['accuracy'])
            model.fit(X, y, batch_size = 32, epochs = 10, validation_split = 0.3, callba
cks= [tensorboard])

model.save('main_train')
```

```
3-conv-64-nodes-0-dense-1573835466
Train on 17462 samples, validate on 7484 samples
Epoch 1/10
17462/17462 [==============================] - 79s 5ms/sample - loss: 0.6231 - accurac
y: 0.6480 - val_loss: 0.5491 - val_accuracy: 0.7223
Epoch 2/10
17462/17462 [==============================] - 67s 4ms/sample - loss: 0.5075 - accurac
y: 0.7493 - val_loss: 0.5006 - val_accuracy: 0.7531
Epoch 3/10
17462/17462 [==============================] - 67s 4ms/sample - loss: 0.4444 - accurac
y: 0.7954 - val_loss: 0.4473 - val_accuracy: 0.7970
Epoch 4/10
17462/17462 [==============================] - 67s 4ms/sample - loss: 0.3959 - accurac
y: 0.8209 - val_loss: 0.4580 - val_accuracy: 0.7897
Epoch 5/10
17462/17462 [==============================] - 67s 4ms/sample - loss: 0.3606 - accurac
y: 0.8395 - val_loss: 0.3937 - val_accuracy: 0.8195
Epoch 6/10
17462/17462 [==============================] - 68s 4ms/sample - loss: 0.3279 - accurac
y: 0.8549 - val_loss: 0.3831 - val_accuracy: 0.8339
Epoch 7/10
17462/17462 [==============================] - 68s 4ms/sample - loss: 0.2969 - accurac
y: 0.8728 - val_loss: 0.3942 - val_accuracy: 0.8272
Epoch 8/10
17462/17462 [==============================] - 68s 4ms/sample - loss: 0.2649 - accurac
y: 0.8860 - val_loss: 0.3831 - val_accuracy: 0.8283
Epoch 9/10
17462/17462 [==============================] - 68s 4ms/sample - loss: 0.2413 - accurac
y: 0.8971 - val_loss: 0.3838 - val_accuracy: 0.8371
Epoch 10/10
17462/17462 [==============================] - 68s 4ms/sample - loss: 0.2130 - accurac
y: 0.9128 - val_loss: 0.3856 - val_accuracy: 0.8328
WARNING:tensorflow:From C:\Users\Evan Jones Boddu\Anaconda3\lib\site-packages\tensorflo
w_core\python\ops\resource_variable_ops.py:1781: calling BaseResourceVariable.__init__
(from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and wi
ll be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
INFO:tensorflow:Assets written to: main_train\assets
```

In [ ]: