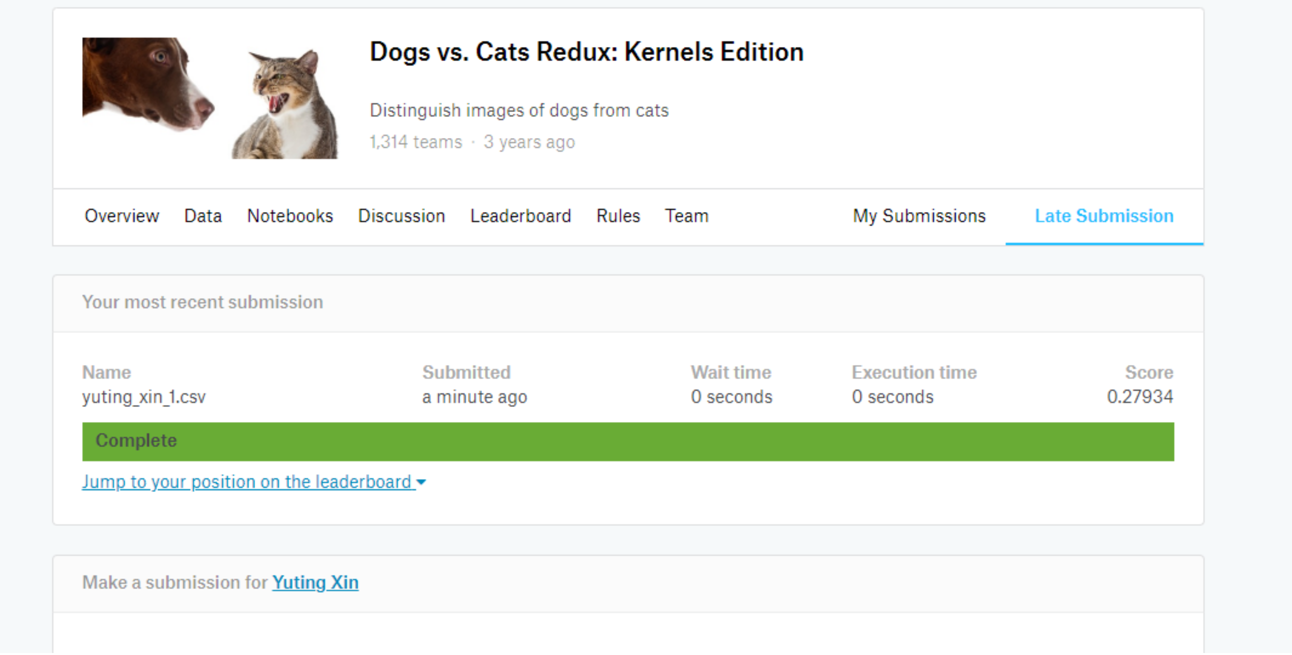
1 Import Data

In [1]:

```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-pytho
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/dogs-vs-cats-redux-kernels-edition/sample_submission.csv
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.4745.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/dog.3992.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.9877.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.11275.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.8771.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.12308.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.11200.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.2908.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/dog.11268.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.7664.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/dog.1091.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.11400.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.4180.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.6570.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.10565.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/dog.4465.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.4909.jpg
/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/cat.11362.jpg
```



Dogs vs. Cats Redux: Kernels Edition

Distinguish images of dogs from cats

1,314 teams · 3 years ago

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
yuting_xin_1.csv	a minute ago	0 seconds	0 seconds	0.27934

Complete

Jump to your position on the leaderboard ▾

Make a submission for Yuting Xin

In [2]:

```
import matplotlib.pyplot as plt

from keras import layers
from keras import models
from keras import optimizers
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Activation, Batch
from keras.preprocessing.image import load_img, img_to_array, ImageDataGenerator
from keras import applications
from keras.callbacks import EarlyStopping, ReduceLROnPlateau

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import random
import glob
```

```
Using TensorFlow backend.
```

In [3]:

```
#We regulate width, height, and channels here
IMAGE_WIDTH=224
IMAGE_HEIGHT=224
IMAGE_CHANNELS=3
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
INPUT_SHAPE=(IMAGE_WIDTH, IMAGE_HEIGHT,IMAGE_CHANNELS)
```

In [4]:

```
#Here we want to import our training images into a data frame.
objects = os.listdir("/kaggle/input/dogs-vs-cats-redux-kernels-edition/train")
objects[0]
```

Out[4]:

```
'cat.4745.jpg'
```

In [5]:

```
#We want tocreate a dataframe with one column showing the number of
#picture and the other one showing the category of the image
categories = []

for n in objects:
    category = n.split('.')[0]
    if category == 'dog':
        categories.append('dog')
    else:
        categories.append('cat')

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

In [6]:

```
df.head()
```

Out[6]:

	filename	category
0	cat.4745.jpg	cat
1	dog.3992.jpg	dog
2	cat.9877.jpg	cat
3	cat.11275.jpg	cat
4	cat.8771.jpg	cat

In [7]:

```
# Take a sample to see whether we successfully read the image

image = load_img("/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/"+objects[0])
plt.imshow(image)
```

Out[7]:

```
<matplotlib.image.AxesImage at 0x7f9ec1893128>
```

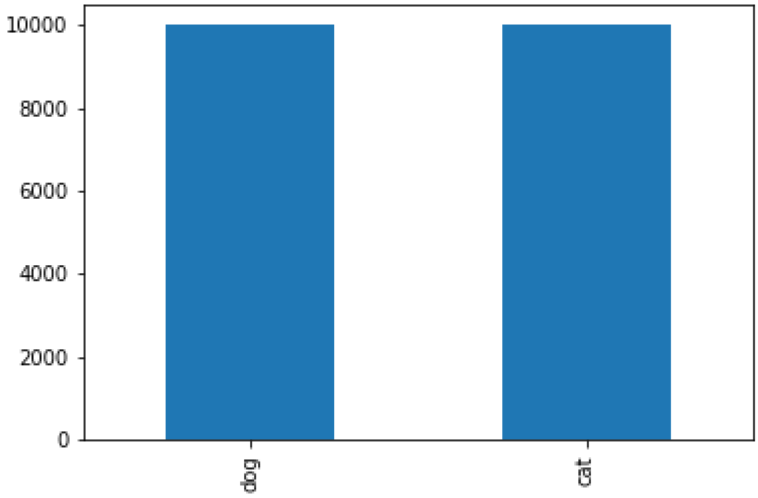
2 Prepare Data

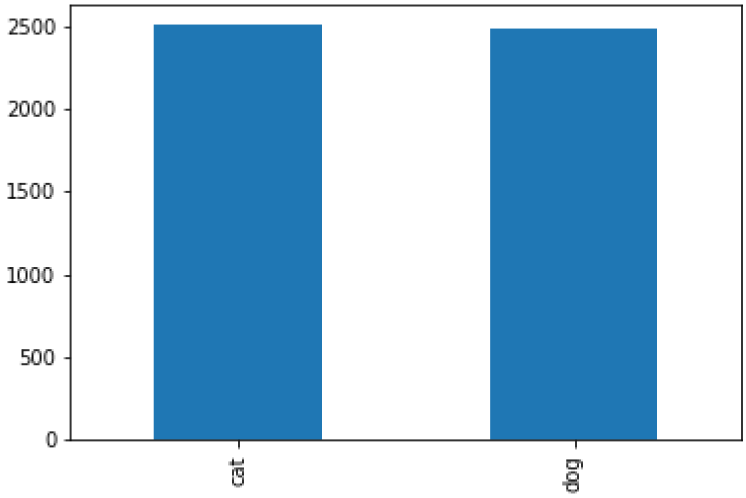
In [8]:

```
#Here we want to split our df into training and testing set. We take 20% of data as our t
train, test = train_test_split(df, test_size=0.2, random_state=100)
train = train.reset_index(drop=True)
test = test.reset_index(drop=True)
```

In [9]:

```
#We want to make sure our two categories in our two sets are balance.
f=plt.figure(1)
train['category'].value_counts().plot.bar()

g=plt.figure(2)
test['category'].value_counts().plot.bar()

plt.show()
```





In [10]:

```
#First we need to define some constrants
#We define batch size to be 10

batch_size=10
```

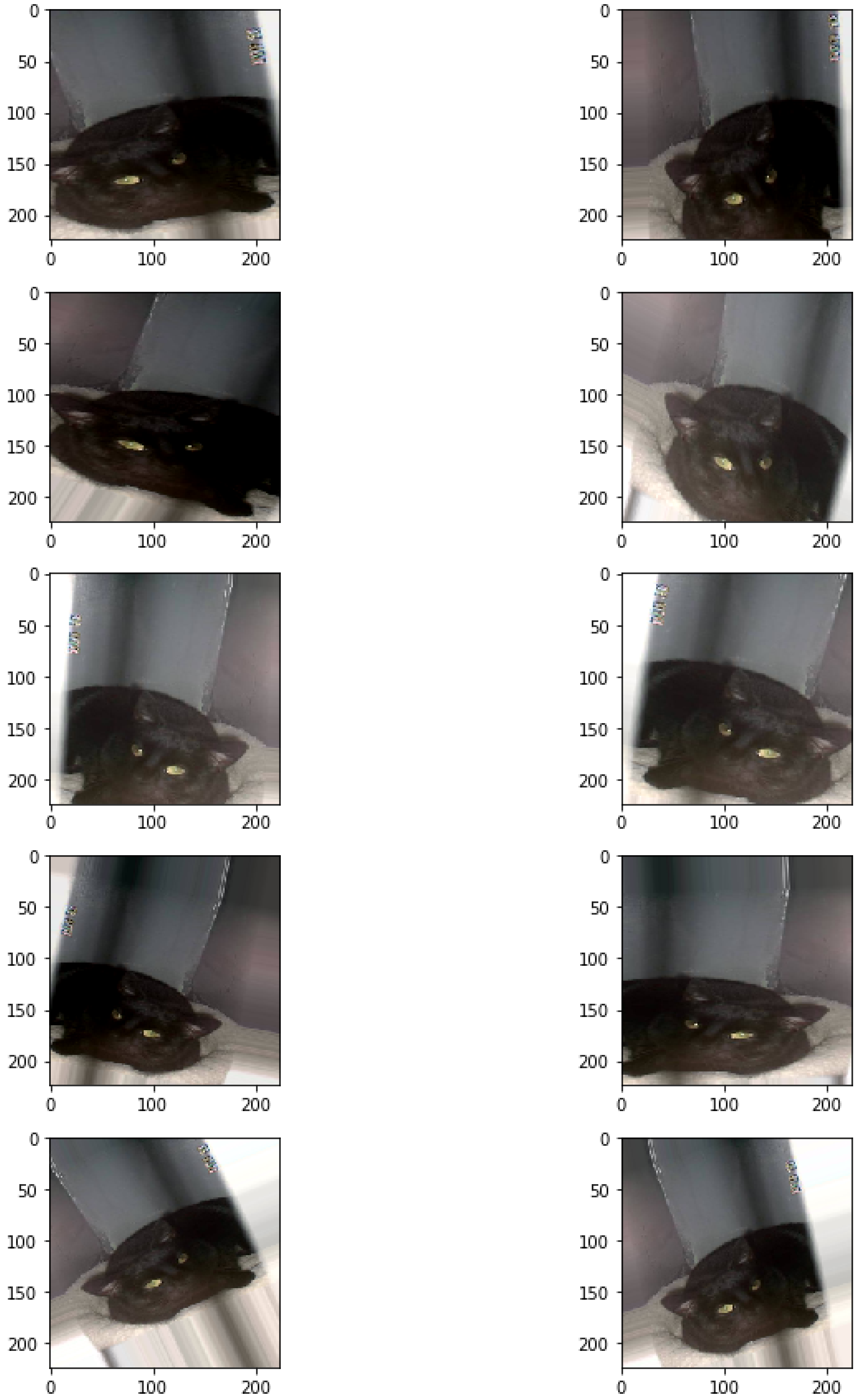
In [11]:

```
#Here we want to transfer both training and testing datasets into a form that can be proc
#Since there are many kinds of dogs and cats and image quality is different, the current
#be enough to construct a useful model. Hence, we use image data generator to generate 10
#for each of pictures.
train_generator = ImageDataGenerator(
    rotation_range=30,#randomly rotate in an angle < 30 degrees
    rescale=1./255,#rescale the values into the model
    shear_range=0.2,#change x or y cordinates while the other one remain the same
    zoom_range=0.3,#zoom in the image
    horizontal_flip=True,#randomly flip the picture horizontally
    width_shift_range=0.1,#shift the image horizontally and vertically
    height_shift_range=0.1,
    channel_shift_range=30#change color of the picture
)
```

In [12]:

```
#Check 10 images we get from one sample image
#Not so good, we need to have a smaller rotation range
example_df = train.sample(n=1).reset_index(drop=True)
example_generator = train_generator.flow_from_dataframe(
    example_df,
     "/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical'
)
plt.figure(figsize=(12, 12))
for i in range(0, 10):
    plt.subplot(5, 2, i+1)
    for x, y in example_generator:
        image = x[0]
        plt.imshow(image)
        break
plt.tight_layout()
plt.show()
```

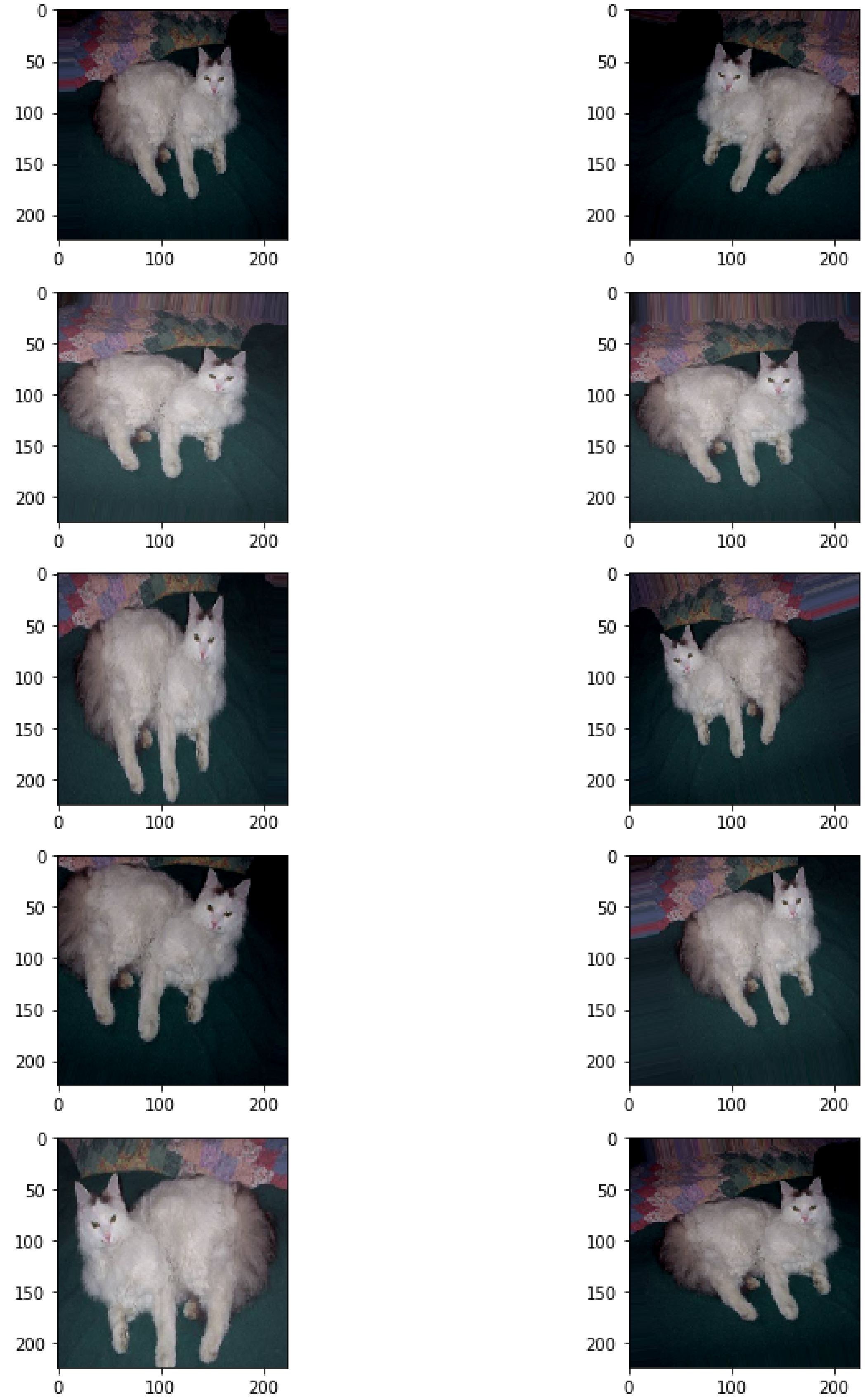
```
Found 1 validated image filenames belonging to 1 classes.
```


0
50
100
150
200
0
100
200

In [13]:

```
#This time it looks ok
train_generator = ImageDataGenerator(
    rotation_range=20,#randomly rotate in an angle < 30 degrees
    rescale=1./255,#rescale the values into the model
    shear_range=0.2,#change x or y cordinates while the other one remain the same
    zoom_range=0.3,#zoom in the image
    horizontal_flip=True,#randomly flip the picture horizontally
    width_shift_range=0.1,#shift the image horizontally and vertically
    height_shift_range=0.1,
    channel_shift_range=30#change color of the picture

)
example_df = train.sample(n=1).reset_index(drop=True)
example_generator = train_generator.flow_from_dataframe(
    example_df,
     "/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical'
)
plt.figure(figsize=(12, 12))
for i in range(0, 10):
    plt.subplot(5, 2, i+1)
    for x, y in example_generator:
        image = x[0]
        plt.imshow(image)
        break
plt.tight_layout()
plt.show()
```

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


0
50
100
150
200
0
100
200

In [14]:

```
#We generate batches of image from our data
train_after = train_generator.flow_from_dataframe(
    train,
    "/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

```
Found 19999 validated image filenames belonging to 2 classes.
```

```
/opt/conda/lib/python3.6/site-packages/keras_preprocessing/image/dataframe_i
terator.py:273: UserWarning: Found 1 invalid image filename(s) in x_col="fil
ename". These filename(s) will be ignored.
  .format(n_invalid, x_col)
```

In [15]:

```
#We want to rescale our testing dataset
test_generator = ImageDataGenerator(rescale=1./255)
test_after = test_generator.flow_from_dataframe(
    test,
    "/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical',
    batch_size=batch_size
)
```

```
Found 5001 validated image filenames belonging to 2 classes.
```

3 Traditional CNN

In [16]:

```
earlystop = EarlyStopping(patience=8)
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=2,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.0001)
callbacks = [earlystop, learning_rate_reduction]
```

In [17]:

```
#In our model we have three convolutional layers. Our maxplooing size is 2X2 for each lay
#Drop our rate is 0.25 for each layer except the flatten step
#Our filter size is always 3x3
#We start with 32 nodes and double this every time
#According to articles, relu is an ideal activation for this kind of problem
model_CNN = Sequential()

model_CNN.add(Conv2D(32, (3, 3), activation='relu', input_shape=(IMAGE_WIDTH, IMAGE_HEIGH
model_CNN.add(BatchNormalization())
model_CNN.add(MaxPooling2D(pool_size=(2, 2)))
model_CNN.add(Dropout(0.25))

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

model_CNN.add(Conv2D(128, (3, 3), activation='relu'))
model_CNN.add(BatchNormalization())
model_CNN.add(MaxPooling2D(pool_size=(2, 2)))
model_CNN.add(Dropout(0.25))

model_CNN.add(Flatten())
model_CNN.add(Dense(64, activation='relu'))
model_CNN.add(BatchNormalization())
model_CNN.add(Dropout(0.25))
model_CNN.add(Dense(2, activation='softmax'))

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

model_CNN.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896       
_________________________________________________________________
batch_normalization_1 (Batch (None, 222, 222, 32)      128       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 32)      0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 111, 111, 32)      0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 109, 109, 64)      18496     
_________________________________________________________________
batch_normalization_2 (Batch (None, 109, 109, 64)      256       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 64)        0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 54, 54, 64)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 128)       73856     
_________________________________________________________________
batch_normalization_3 (Batch (None, 52, 52, 128)       512       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 26, 26, 128)       0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 26, 26, 128)       0         
```

```
_________________________________________________________________
flatten_1 (Flatten)          (None, 86528)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                5537856   
_________________________________________________________________
batch_normalization_4 (Batch (None, 64)                256       
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130       
=================================================================
Total params: 5,632,386
Trainable params: 5,631,810
Non-trainable params: 576
_________________________________________________________________
```

In [18]:

```
history = model_CNN.fit_generator(
    train_after,
    epochs=15,
    validation_data=test_after,
    validation_steps=test.shape[0]//batch_size,
    steps_per_epoch=train.shape[0]//batch_size,
    callbacks=callbacks
)
```

```
Epoch 1/15
2000/2000 [==============================] - 344s 172ms/step - loss: 0.6949
- accuracy: 0.6137 - val_loss: 0.5667 - val_accuracy: 0.6742
Epoch 2/15
   3/2000 [..............................] - ETA: 1:22 - loss: 0.6400 - accu
racy: 0.6000

/opt/conda/lib/python3.6/site-packages/keras/callbacks/callbacks.py:1042: Ru
ntimeWarning: Reduce LR on plateau conditioned on metric `val_acc` which is
not available. Available metrics are: val_loss,val_accuracy,loss,accuracy,lr
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning

2000/2000 [==============================] - 286s 143ms/step - loss: 0.583
9 - accuracy: 0.6841 - val_loss: 0.4487 - val_accuracy: 0.6704
Epoch 3/15
2000/2000 [==============================] - 281s 141ms/step - loss: 0.553
6 - accuracy: 0.7129 - val_loss: 0.4714 - val_accuracy: 0.7536
Epoch 4/15
2000/2000 [==============================] - 279s 140ms/step - loss: 0.537
2 - accuracy: 0.7302 - val_loss: 0.3287 - val_accuracy: 0.7908
Epoch 5/15
2000/2000 [==============================] - 278s 139ms/step - loss: 0.499
6 - accuracy: 0.7559 - val_loss: 0.3791 - val_accuracy: 0.8091
Epoch 6/15
2000/2000 [==============================] - 279s 140ms/step - loss: 0.479
7 - accuracy: 0.7717 - val_loss: 0.2969 - val_accuracy: 0.7914
Epoch 7/15
2000/2000 [==============================] - 279s 139ms/step - loss: 0.457
3 - accuracy: 0.7847 - val_loss: 0.4088 - val_accuracy: 0.8275
Epoch 8/15
2000/2000 [==============================] - 279s 140ms/step - loss: 0.437
3 - accuracy: 0.7969 - val_loss: 1.2013 - val_accuracy: 0.7115
Epoch 9/15
2000/2000 [==============================] - 279s 140ms/step - loss: 0.423
9 - accuracy: 0.8062 - val_loss: 0.2248 - val_accuracy: 0.7676
Epoch 10/15
2000/2000 [==============================] - 279s 140ms/step - loss: 0.409
8 - accuracy: 0.8123 - val_loss: 0.4792 - val_accuracy: 0.8553
Epoch 11/15
2000/2000 [==============================] - 279s 139ms/step - loss: 0.412
2 - accuracy: 0.8147 - val_loss: 0.4266 - val_accuracy: 0.8369
Epoch 12/15
2000/2000 [==============================] - 280s 140ms/step - loss: 0.402
2 - accuracy: 0.8187 - val_loss: 0.3876 - val_accuracy: 0.8782
Epoch 13/15
2000/2000 [==============================] - 280s 140ms/step - loss: 0.397
8 - accuracy: 0.8218 - val_loss: 0.4434 - val_accuracy: 0.8453
Epoch 14/15
2000/2000 [==============================] - 280s 140ms/step - loss: 0.378
```

```
7 - accuracy: 0.8329 - val_loss: 0.3150 - val_accuracy: 0.8622
Epoch 15/15
2000/2000 [==============================] - 279s 140ms/step - loss: 0.372
5 - accuracy: 0.8378 - val_loss: 0.2304 - val_accuracy: 0.8383
```

In [ ]:

```

```

4 VGG16 Approach

In [1]:

```
#After comparing my traditional CNN approach with noteboards on Kaggle, I found that they
#which produce a much better results. VGG 16 is a CNN model designed by researchers from
#The model is good at analyzing images tasks. So I am going to try it here.
```

In [21]:

```
from keras.applications import VGG16
```

In [24]:

```
from keras.applications import VGG16
from keras.models import Model
from keras.layers import GlobalMaxPooling2D

#load imagenet weights for the networks
pre_trained_model = VGG16(input_shape=INPUT_SHAPE, include_top=False, weights="imagenet")

for layer in pre_trained_model.layers[:15]:
    layer.trainable = False

for layer in pre_trained_model.layers[15:]:
    layer.trainable = True

last_layer = pre_trained_model.get_layer('block5_pool')
last_output = last_layer.output


model = Sequential()
model.add(ZeroPadding2D((1,1),input_shape=(3,224,224)))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(128, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(256, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(ZeroPadding2D((1,1)))
model.add(Convolution2D(512, 3, 3, activation='relu'))
model.add(MaxPooling2D((2,2), strides=(2,2)))

model.add(GlobalMaxPooling2D())
model.add(Dense(512, activation='relu'))
model.add(Dropout=(0.5))
model.add(Dense(1,activation='sigmoid'))
```

```
model_VGG.compile(loss='binary_crossentropy',
              optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
              metrics=['accuracy'])

model_VGG.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_3 (InputLayer)         (None, 224, 224, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
global_max_pooling2d_1 (Glob (None, 512)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 512)               262656    
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 513       
=================================================================
Total params: 14,977,857
Trainable params: 7,342,593
```

```
Non-trainable params: 7,635,264
_________________________________________________________________
```

In [94]:

```
#We generate batches of image from our data
train_after = train_generator.flow_from_dataframe(
    train,
    "/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='binary',
    batch_size=batch_size,

)
#We want to rescale our testing dataset
test_generator = ImageDataGenerator(rescale=1./255)
test_after = test_generator.flow_from_dataframe(
    test,
    "/kaggle/input/dogs-vs-cats-redux-kernels-edition/train/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='binary',
    batch_size=batch_size,

)
```

```
Found 19999 validated image filenames belonging to 2 classes.
Found 5001 validated image filenames belonging to 2 classes.
```

In [ ]:

```
earlystop = EarlyStopping(patience=2)

callbacks = [earlystop, learning_rate_reduction]
```

In [95]:

```
history = model_VGG.fit_generator(
    train_after,
    epochs=5,
    validation_data=test_after,
    validation_steps=test.shape[0]//batch_size,
    steps_per_epoch=train.shape[0]//batch_size,
    callbacks=callbacks
)
```

```
Epoch 1/5
2000/2000 [==============================] - 282s 141ms/step - loss: 0.0693
- accuracy: 0.9726 - val_loss: 0.0198 - val_accuracy: 0.9682
Epoch 2/5
2000/2000 [==============================] - 281s 141ms/step - loss: 0.0696
- accuracy: 0.9723 - val_loss: 0.0172 - val_accuracy: 0.9726
Epoch 3/5
2000/2000 [==============================] - 283s 142ms/step - loss: 0.0653
- accuracy: 0.9736 - val_loss: 0.2926 - val_accuracy: 0.9715
Epoch 4/5
2000/2000 [==============================] - 283s 142ms/step - loss: 0.0608
- accuracy: 0.9767 - val_loss: 0.0152 - val_accuracy: 0.9701
Epoch 5/5
2000/2000 [==============================] - 282s 141ms/step - loss: 0.0583
- accuracy: 0.9766 - val_loss: 0.0013 - val_accuracy: 0.9721
```

In [ ]:

```
#We see that this time val_accuracy is very high and val_loss is low
```

5 Predict

In [98]:

```
#import testing data
result_filenames = os.listdir("/kaggle/input/dogs-vs-cats-redux-kernels-edition/test/test
result = pd.DataFrame({
    'filename': result_filenames
})
#read the testing data
n=result.shape[0]
```

In [99]:

```
result
```

Out[99]:

	filename
0	10550.jpg
1	11840.jpg
2	6842.jpg
3	10646.jpg
4	10394.jpg
...	...
12495	3472.jpg
12496	4871.jpg
12497	7269.jpg
12498	6718.jpg
12499	12203.jpg

12500 rows × 1 columns

In [85]:

```
batch_size=10
```

In [100]:

```
#Process data the same way we do to testing data
#Due to some issue with imagedatagenerator, we cannot rescale it here and we tell the mod
#so that predictions will match labels
result_generator = ImageDataGenerator()
result_after = result_generator.flow_from_dataframe(
    result,
    "/kaggle/input/dogs-vs-cats-redux-kernels-edition/test/test",
    x_col='filename',
    y_col=None,
    target_size=IMAGE_SIZE,
    class_mode=None,
    batch_size=batch_size,
    shuffle=False

)
```

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

In [101]:

```
prediction = model_VGG.predict_generator(result_after)
```

In [60]:

```
prediction
```

Out[60]:

```
array([[9.9997693e-01],
       [2.2469968e-02],
       [2.3683906e-04],
       ...,
       [2.9802322e-08],
       [9.2008930e-01],
       [9.9999976e-01]], dtype=float32)
```

In [ ]:

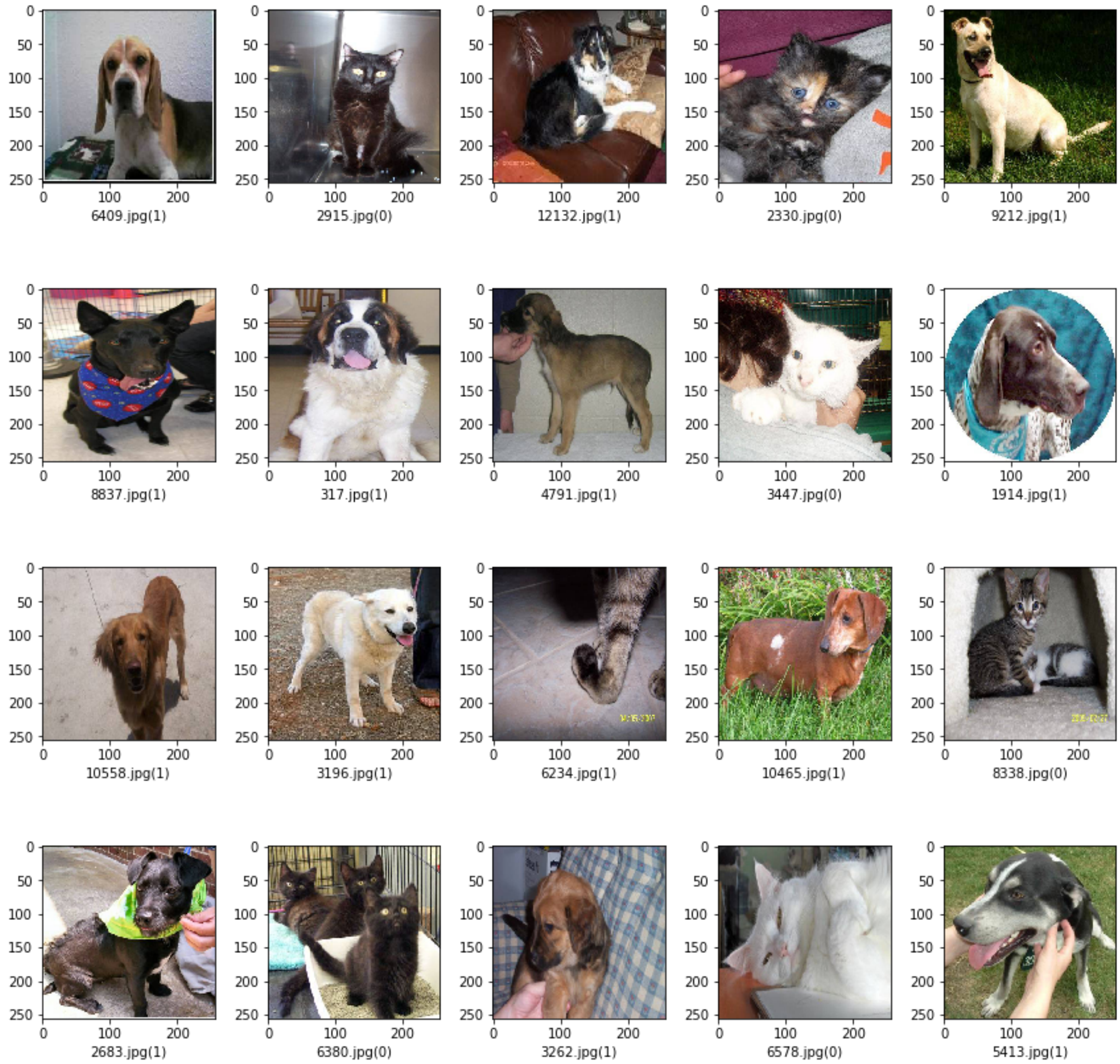
```
submission = result.copy()
submission['id'] = submission_df['filename'].str.split('.').str[0]
submission['label'] = prediction
submission.drop(['filename'], axis=1, inplace=True)
```

In [102]:

```
#Greater than 0.5 is dog, otherwise cat
result['category'] = np.where(prediction > 0.5, 1,0)
```

In [103]:

```
#we run a sample test to check our predictions match labels
sample_test = result.sample(n=20).reset_index()
sample_test.head()
plt.figure(figsize=(12, 12))
for index, row in sample_test.iterrows():
    filename = row['filename']
    category = row['category']
    img = load_img("/kaggle/input/dogs-vs-cats-redux-kernels-edition/test/test/"+filename
    plt.subplot(4, 5, index+1)
    plt.imshow(img)
    plt.xlabel(filename + '(' + "{}".format(category) + ')')
plt.tight_layout()
plt.show()
#We can see here out of 20 pictures, only one is wrong.
#VGG is an excellent model!
```



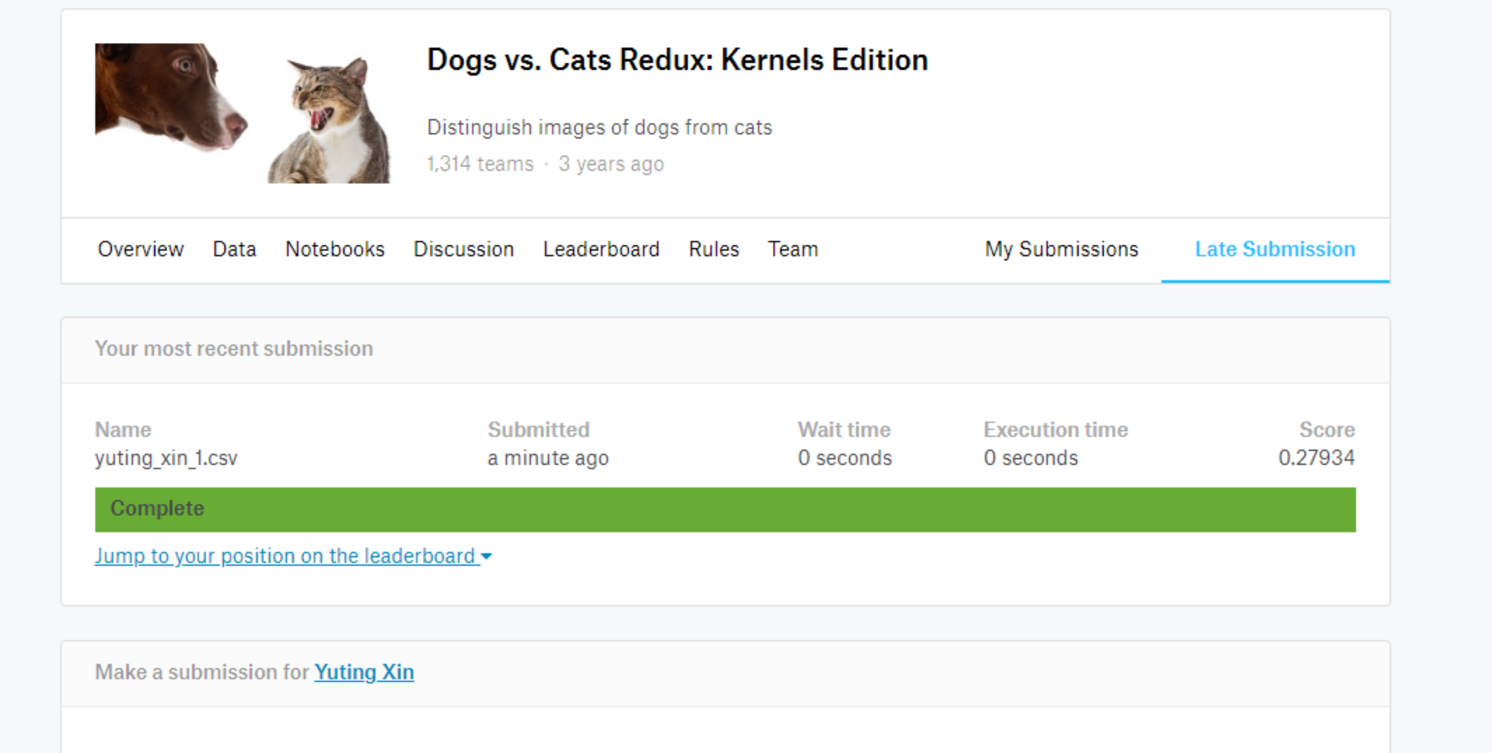
6 Submit result

In [90]:

```
submission.to_csv('submission_yuting_xin_1.csv', index=False)
```

7 Source Material

https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification (https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification) https://www.kaggle.com/bulentsiyah/dogs-vs-cats-classification-vgg16-fine-tuning (https://www.kaggle.com/bulentsiyah/dogs-vs-cats-classification-vgg16-fine-tuning) https://www.kaggle.com/shivamb/cnn-architectures-vgg-resnet-inception-tl (https://www.kaggle.com/shivamb/cnn-architectures-vgg-resnet-inception-tl)



Dogs vs. Cats Redux: Kernels Edition

Distinguish images of dogs from cats

1,314 teams · 3 years ago

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
yuting_xin_1.csv	a minute ago	0 seconds	0 seconds	0.27934

Complete

Jump to your position on the leaderboard

Make a submission for Yuting Xin