In [1]:
```
# 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-pyth
# For example, here's several helpful packages to load

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 read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all file

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

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserv
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside o
```

```
/kaggle/input/dogs-vs-cats/test1.zip
/kaggle/input/dogs-vs-cats/train.zip
/kaggle/input/dogs-vs-cats/sampleSubmission.csv
```

Data

In [2]:
```
import os
import zipfile
import pandas as pd
from tqdm import tqdm
import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from tensorflow.keras.optimizers import RMSprop
```

In [3]:
```
work_path = './cats_and_dogs_filtered'
os.mkdir(work_path)
```

In [4]:
```
local_zip = '../input/dogs-vs-cats/test1.zip'
zip_ref = zipfile.ZipFile(local_zip,'r')
zip_ref.extractall(work_path)

local_zip = '../input/dogs-vs-cats/train.zip'
zip_ref = zipfile.ZipFile(local_zip,'r')
zip_ref.extractall(work_path)

zip_ref.close()
```

In [5]:
```
train_path = os.path.join(work_path, 'train')
test_path = os.path.join(work_path, 'test1')
```

In [6]:
```python
train_df = pd.DataFrame({'image_name':os.listdir(train_path)})
train_df['label'] =train_df['image_name'].apply(lambda x: x.split('.')[0])
train_df
```

Out[6]:

	image_name	label
0	cat.2364.jpg	cat
1	cat.4566.jpg	cat
2	cat.2311.jpg	cat
3	dog.4811.jpg	dog
4	dog.2935.jpg	dog
...	...	...
24995	cat.4039.jpg	cat
24996	cat.5098.jpg	cat
24997	cat.591.jpg	cat
24998	cat.5809.jpg	cat
24999	dog.10160.jpg	dog

25000 rows × 2 columns

In [7]:
```python
test_df = pd.DataFrame({'image_name':os.listdir(test_path)})
test_df['label'] =test_df['image_name'].apply(lambda x: x.split('.')[0])
test_df
```

Out[7]:

	image_name	label
0	8791.jpg	8791
1	10695.jpg	10695
2	8333.jpg	8333
3	6525.jpg	6525
4	6482.jpg	6482
...	...	...
12495	6756.jpg	6756
12496	6487.jpg	6487
12497	7640.jpg	7640
12498	2117.jpg	2117
12499	899.jpg	899

12500 rows × 2 columns

In [8]:
```python
dog_path_train = os.path.join(train_path, 'dog')
os.mkdir(dog_path_train)
dog_df_train = train_df[train_df.label=='dog']
for n in tqdm(dog_df_train.image_name):
    os.rename((os.path.join(train_path, n)), (os.path.join(dog_path_train, n)))
```

```
100%|██████████| 12500/12500 [00:00<00:00, 33887.98it/s]
```

In [9]:
```python
cat_path_train = os.path.join(train_path, 'cat')
os.mkdir(cat_path_train)
cat_df_train = train_df[train_df.label=='cat']
for n in tqdm(cat_df_train.image_name):
    os.rename((os.path.join(train_path, n)), (os.path.join(cat_path_train, n)))
```

```
100%|██████████| 12500/12500 [00:00<00:00, 37334.26it/s]
```

In [10]:
```python
#check

base_dir = './cats_and_dogs_filtered'

print(' Contents of base directory')
print(os.listdir(base_dir))

print('\n Contents of Train directory')
train_path = f'{base_dir}/train'
print(os.listdir(train_path))

print('\n Contents of validation directory')
print(os.listdir(test_path)[:5])
```

```
 Contents of base directory
['train', 'test1']

 Contents of Train directory
['dog', 'cat']

 Contents of validation directory
['8791.jpg', '10695.jpg', '8333.jpg', '6525.jpg', '6482.jpg']
```

In [11]:
```python
train_dir = os.path.join(base_dir,'train')
validation_dir = os.path.join(base_dir,'test1')

train_cats_dir = os.path.join(train_dir,'cat')
train_dogs_dir = os.path.join(train_dir,'dog')
```

In [12]:
```python
train_cats_names = os.listdir(train_cats_dir)
train_dogs_names = os.listdir(train_dogs_dir)

print(train_cats_names[:5])
print(train_dogs_names[:5])
```

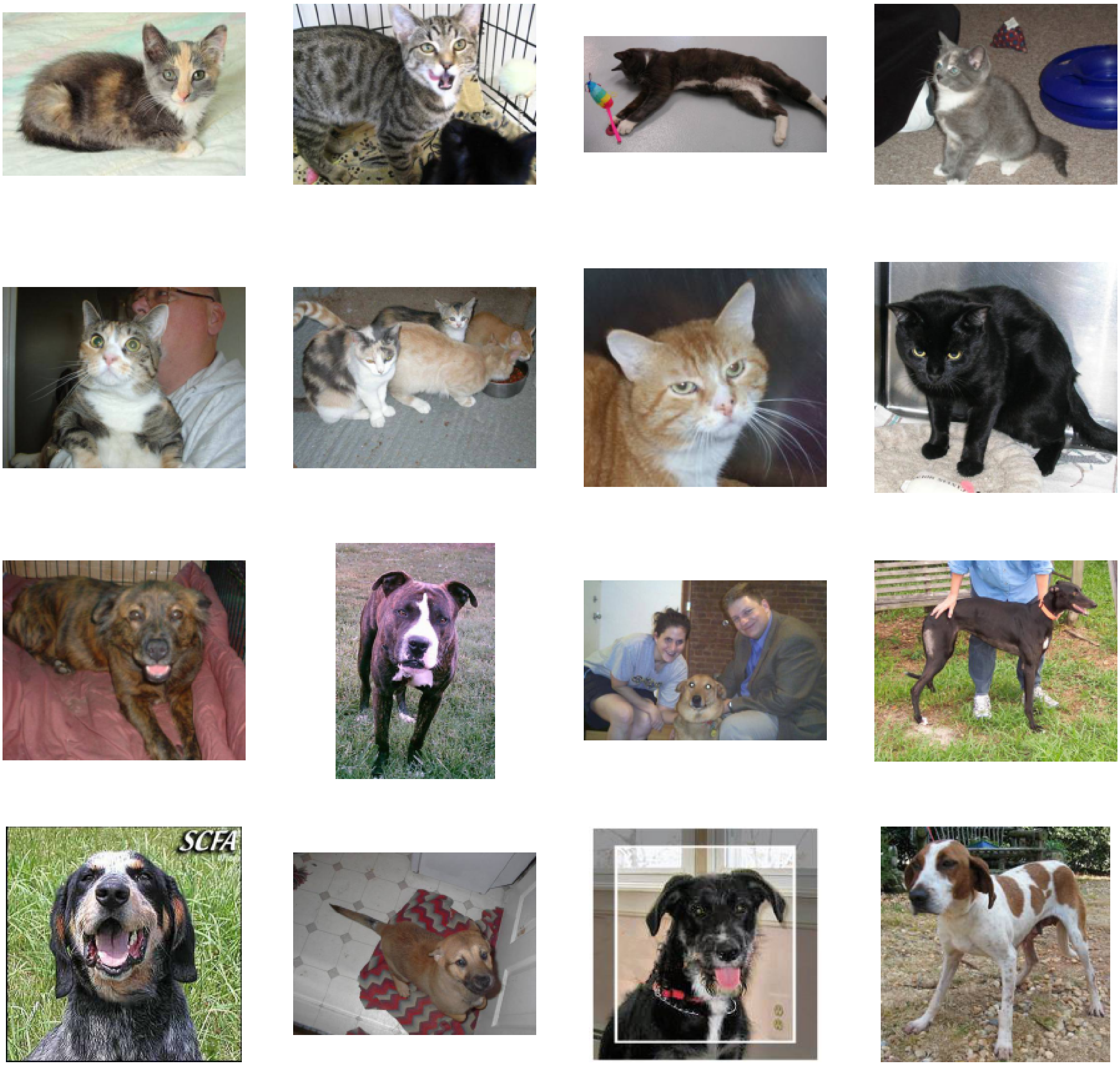
```
['cat.2364.jpg', 'cat.4566.jpg', 'cat.2311.jpg', 'cat.3824.jpg', 'cat.5978.jpg']
['dog.4811.jpg', 'dog.2935.jpg', 'dog.2309.jpg', 'dog.1948.jpg', 'dog.2809.jpg']
```

In [13]:
```python
#number

print(f'numbers of cats in training set = {len(train_cats_names)}')
print(f'numbers of dogs in training set = {len(train_dogs_names)}')
print(f'numbers of cats and dogs in validation set = {len(os.listdir(validation_dir))}')
```

```
numbers of cats in training set = 12500
numbers of dogs in training set = 12500
numbers of cats and dogs in validation set = 12500
```

In [14]:

```python
#image 확인

%matplotlib inline

nrows = 4
ncols = 4

pic_index = 0

fig = plt.gcf()
fig.set_size_inches(nrows*4,ncols*4)

next_cat_pic = [os.path.join(train_cats_dir,fname) for fname in train_cats_names[pic_ind

next_dog_pic = [os.path.join(train_dogs_dir,fname) for fname in train_dogs_names[pic_ind

for i ,img_path in enumerate(next_cat_pic+next_dog_pic):
    sp = plt.subplot(nrows,ncols,i+1)
    sp.axis('off')

    img = mpimg.imread(img_path)
    plt.imshow(img)
plt.show()
```



Model

In [15]:
```
def create_model():

  model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16,(3,3), activation = 'relu', input_shape=(150,150,3)),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(32,(3,3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(64,(3,3), activation = 'relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation = 'relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
  ])


  model.compile(optimizer=RMSprop(lr=0.001),
                loss='binary_crossentropy',
                metrics=['accuracy'])

  return model
```

In [16]:
```
model = create_model()
model.summary()
```

```
2022-05-15 16:25:50.260096: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:50.371942: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:50.372922: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:50.374218: I tensorflow/core/platform/cpu_feature_guard.cc:142] This
TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use
the following CPU instructions in performance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler f
lags.
2022-05-15 16:25:50.374557: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:50.375342: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:50.376079: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:52.560239: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:52.561257: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:52.562053: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:93
7] successful NUMA node read from SysFS had negative value (-1), but there must be at
least one NUMA node, so returning NUMA node zero
2022-05-15 16:25:52.563342: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1510] C
reated device /job:localhost/replica:0/task:0/device:GPU:0 with 15403 MB memory:  -> d
evice: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.
0
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 18496)             0
_________________________________________________________________
dense (Dense)                (None, 512)               9470464
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 9,494,561
Trainable params: 9,494,561
```

```
Non-trainable params: 0
_________________________________________________________________

/opt/conda/lib/python3.7/site-packages/keras/optimizer_v2/optimizer_v2.py:356: UserWar
ning: The `lr` argument is deprecated, use `learning_rate` instead.
  "The `lr` argument is deprecated, use `learning_rate` instead.")
```

In [17]:
```
#이미지전처리
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest',
      validation_split=0.2
                                   )

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150,150),
    batch_size=50,
    class_mode='binary',
    subset='training'
)


validation_generator = train_datagen.flow_from_directory(
    train_dir, # same directory as training data
    target_size=(150, 150),
    batch_size=50,
    class_mode='binary',
    subset='validation')
```

```
Found 20000 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
```

In [18]:
```
class mycallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self,epoch,logs={}):
        if(logs.get('val_accuracy')>=0.90):
            self.model.stop_training = True

callback = mycallback()
```

Training

In [19]:
```
history = model.fit(
    train_generator,
    steps_per_epoch = train_generator.samples//50,#batch_size,
    epochs = 30,
    verbose=1,
    validation_data = validation_generator,
    validation_steps = validation_generator.samples//50,#batch_size,
    callbacks=[callback]
)
```

```
2022-05-15 16:25:55.805190: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.c
c:185] None of the MLIR Optimization Passes are enabled (registered 2)

Epoch 1/30

2022-05-15 16:25:57.927405: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Loaded
cuDNN version 8005

400/400 [==============================] - 225s 543ms/step - loss: 0.6823 - accuracy:
0.6075 - val_loss: 0.5825 - val_accuracy: 0.7028
Epoch 2/30
400/400 [==============================] - 216s 539ms/step - loss: 0.5991 - accuracy:
0.6782 - val_loss: 0.5586 - val_accuracy: 0.6972
Epoch 3/30
400/400 [==============================] - 214s 536ms/step - loss: 0.5650 - accuracy:
0.7054 - val_loss: 0.5299 - val_accuracy: 0.7314
Epoch 4/30
400/400 [==============================] - 217s 540ms/step - loss: 0.5403 - accuracy:
0.7287 - val_loss: 0.5360 - val_accuracy: 0.7230
Epoch 5/30
400/400 [==============================] - 217s 542ms/step - loss: 0.5253 - accuracy:
0.7383 - val_loss: 0.4996 - val_accuracy: 0.7584
Epoch 6/30
400/400 [==============================] - 217s 542ms/step - loss: 0.5131 - accuracy:
0.7442 - val_loss: 0.5276 - val_accuracy: 0.7336
Epoch 7/30
400/400 [==============================] - 218s 545ms/step - loss: 0.5009 - accuracy:
0.7580 - val_loss: 0.4924 - val_accuracy: 0.7634
Epoch 8/30
400/400 [==============================] - 220s 551ms/step - loss: 0.4883 - accuracy:
0.7666 - val_loss: 0.4596 - val_accuracy: 0.7784
Epoch 9/30
400/400 [==============================] - 223s 556ms/step - loss: 0.4753 - accuracy:
0.7735 - val_loss: 0.4295 - val_accuracy: 0.8036
Epoch 10/30
400/400 [==============================] - 223s 557ms/step - loss: 0.4688 - accuracy:
0.7784 - val_loss: 0.4398 - val_accuracy: 0.7966
Epoch 11/30
400/400 [==============================] - 221s 551ms/step - loss: 0.4609 - accuracy:
0.7871 - val_loss: 0.4554 - val_accuracy: 0.7858
Epoch 12/30
400/400 [==============================] - 222s 554ms/step - loss: 0.4586 - accuracy:
0.7864 - val_loss: 0.4958 - val_accuracy: 0.7546
Epoch 13/30
400/400 [==============================] - 222s 555ms/step - loss: 0.4448 - accuracy:
0.7940 - val_loss: 0.4062 - val_accuracy: 0.8128
Epoch 14/30
400/400 [==============================] - 222s 555ms/step - loss: 0.4331 - accuracy:
0.8012 - val_loss: 0.4581 - val_accuracy: 0.7854
Epoch 15/30
400/400 [==============================] - 224s 559ms/step - loss: 0.4224 - accuracy:
0.8083 - val_loss: 0.4109 - val_accuracy: 0.8086
Epoch 16/30
400/400 [==============================] - 225s 562ms/step - loss: 0.4194 - accuracy:
0.8076 - val_loss: 0.4474 - val_accuracy: 0.7950
Epoch 17/30
```

```
400/400 [==============================] - 225s 563ms/step - loss: 0.4158 - accuracy: 0.8118 - val_loss: 0.3881 - val_accuracy: 0.8236
Epoch 18/30
400/400 [==============================] - 224s 560ms/step - loss: 0.4122 - accuracy: 0.8159 - val_loss: 0.4182 - val_accuracy: 0.8102
Epoch 19/30
400/400 [==============================] - 222s 555ms/step - loss: 0.4095 - accuracy: 0.8165 - val_loss: 0.3832 - val_accuracy: 0.8344
Epoch 20/30
400/400 [==============================] - 231s 577ms/step - loss: 0.3978 - accuracy: 0.8224 - val_loss: 0.3972 - val_accuracy: 0.8264
Epoch 21/30
400/400 [==============================] - 228s 568ms/step - loss: 0.4025 - accuracy: 0.8227 - val_loss: 0.3812 - val_accuracy: 0.8320
Epoch 22/30
400/400 [==============================] - 228s 569ms/step - loss: 0.3969 - accuracy: 0.8257 - val_loss: 0.3711 - val_accuracy: 0.8380
Epoch 23/30
400/400 [==============================] - 230s 574ms/step - loss: 0.3959 - accuracy: 0.8281 - val_loss: 0.3717 - val_accuracy: 0.8426
Epoch 24/30
400/400 [==============================] - 229s 572ms/step - loss: 0.3872 - accuracy: 0.8299 - val_loss: 0.4125 - val_accuracy: 0.8234
Epoch 25/30
400/400 [==============================] - 233s 581ms/step - loss: 0.3831 - accuracy: 0.8300 - val_loss: 0.4313 - val_accuracy: 0.8140
Epoch 26/30
400/400 [==============================] - 235s 588ms/step - loss: 0.3837 - accuracy: 0.8309 - val_loss: 0.5735 - val_accuracy: 0.7570
Epoch 27/30
400/400 [==============================] - 227s 569ms/step - loss: 0.3838 - accuracy: 0.8320 - val_loss: 0.3967 - val_accuracy: 0.8364
Epoch 28/30
400/400 [==============================] - 220s 547ms/step - loss: 0.3821 - accuracy: 0.8345 - val_loss: 0.3679 - val_accuracy: 0.8480
Epoch 29/30
400/400 [==============================] - 226s 564ms/step - loss: 0.3806 - accuracy: 0.8357 - val_loss: 0.4853 - val_accuracy: 0.8132
Epoch 30/30
400/400 [==============================] - 223s 557ms/step - loss: 0.3667 - accuracy: 0.8382 - val_loss: 0.3574 - val_accuracy: 0.8494
```

Accuracy

In [20]:
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```