Deep Learning Project: Pet Classifier using CNN

Prepration

- Extract the ipynb file and the data in the same folder

Data Set

- A production grade program as 10,000 training images
- This is a small program with 20 images of cats and 20 images of dogs.
- The evaluation set has 10 images of cats and 10 images of dogs

Runs

- The student is expected to run the 100-300 training step
- A production grade code would have about 20k-50k training steps

Import modules

In [1]:
```
from __future__ import absolute_import
from __future__ import division
```

In [2]:
```
from __future__ import print_function
import os
import glob
import cv2
```

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import random
import sys
```

In [4]:
```
# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals

# to make this notebook's output stable across runs
def reset_graph(seed=42):
    tf.reset_default_graph()
    tf.set_random_seed(seed)
    np.random.seed(seed)
```

Set hyper parameters

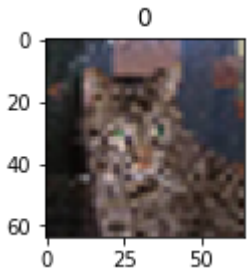
- Run the program with three num_steps : 100,200,300

In [5]:
```
reset_graph()

img_size = 32
num_channels = 3
img_size_flat = img_size * img_size * num_channels
img_shape = (img_size, img_size)
trainpath='./data/train'
testpath='./data/test'
labels = {'cats': 0, 'dogs': 1}
fc_size=32 #size of the output of final FC layer
num_steps=300 #Try 100, 200, 300. number of steps that training data should be looped. Usually 20K
tf.logging.set_verbosity(tf.logging.INFO)
```

Read the image dataset

In [6]:
```
def read_images_classes(basepath,imgSize=img_size):
    image_stack = []
    label_stack = []

    for counter, l in enumerate(labels):
        path = os.path.join(basepath, l,'*g')
        for img in glob.glob(path):
            one_hot_vector =np.zeros(len(labels),dtype=np.int16)
            one_hot_vector[counter]=1
            image = cv2.imread(img)
            im_resize = cv2.resize(image,img_shape, interpolation=cv2.INTER_CU
BIC)
            image_stack.append(im_resize)
            label_stack.append(labels[l])
    return np.array(image_stack), np.array(label_stack)

X_train, y_train=read_images_classes(trainpath)
X_test, y_test=read_images_classes(testpath)

#test a sample image
print('length of train image set',len(X_train))
print('X_data shape:', X_train.shape)
print('y_data shape:', y_train.shape)

fig1 = plt.figure()
ax1 = fig1.add_subplot(2,2,1)
img = cv2.resize(X_train[0],(64,64), interpolation=cv2.INTER_CUBIC)
ax1.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title(y_train[0])
plt.show()
```

```
length of train image set 40
X_data shape: (40, 32, 32, 3)
y_data shape: (40,)
```



Assignment: Define the tensorflow model

The model should have the following layers

- input later
- conv layer 1 with 32 filters of kernel size[5,5],
- pooling layer 1 with pool size[2,2] and stride 2
- conv layer 2 with 64 filters of kernel size[5,5],
- pooling layer 2 with pool size[2,2] and stride 2
- dense layer whose output size is fixed in the hyper parameter: fc_size=32
- drop out layer with droput probability 0.4
- predict the class by doing a softmax on the output of the dropout layers

Training

- For training fefine the loss function and minimize it
- For evaluation calculate the accuracy

Reading Material

- For ideas look at tensorflow layers tutorial

The cnn_model_fn has to be defined here by the student

In [37]:
```
def cnn_model_fn(features, labels, mode):
    #input later
    input_layer=tf.reshape(features["x"],[-1,img_size,img_size,num_channels])

    conv_layer_1=tf.layers.conv2d(inputs=input_layer,filters=32,kernel_size=[5
,5],padding="same",activation=tf.nn.relu)

    #pooling layer 1 with pool size[2,2] and stride 2
    pool_layer_1=tf.layers.max_pooling2d(inputs=conv_layer_1,pool_size=[2,2],s
trides=2)

    #conv layer 2 with 64 filters of kernel size[5,5],
    conv_layer_2=tf.layers.conv2d(inputs=pool_layer_1,filters=64,kernel_size=[
5,5],padding="same",activation=tf.nn.relu)


    #pooling layer 2 with pool size[2,2] and stride 2
    pool_layer_2=tf.layers.max_pooling2d(inputs=conv_layer_2,pool_size=[2,2],s
trides=2)
    pool_layer_2_flat=tf.reshape(pool_layer_2,[-1,8*8*64])

    #dense layer whose output size is fixed in the hyper
    dense=tf.layers.dense(inputs=pool_layer_2_flat,units=fc_size,activation=tf
.nn.relu)

    #drop out layer with droput probability 0.4
    dropout=tf.layers.dropout(inputs=dense,rate=0.4,training=mode==tf.estimato
r.ModeKeys.TRAIN)

    #logits layer
    logits=tf.layers.dense(inputs=dropout,units=2)

    #configure the predict mode
    predictions={
    "classes":tf.argmax(input=logits,axis=1),
    "probabilities":tf.nn.softmax(logits,name="softmax_tensor")
    }

    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode=mode,predictions=predictions)

    #calculate the loss for both train and eval models

    onehot_labels=tf.one_hot(indices=tf.cast(labels,tf.int32),depth=2)
    loss=tf.losses.softmax_cross_entropy(onehot_labels=onehot_labels,logits=lo
gits)

    #configure operations for traininf mode
    if mode ==tf.estimator.ModeKeys.TRAIN:
        optimizer=tf.train.GradientDescentOptimizer(learning_rate=0.01)
        train_op=optimizer.minimize(loss=loss,global_step=tf.train.get_global_
step())
        return tf.estimator.EstimatorSpec(mode=mode,loss=loss,train_op=train_o
p)

    eval_metric_ops={"accuracy":tf.metrics.accuracy(labels=labels,predictions=
```

```
predictions["classes"])}
    return tf.estimator.EstimatorSpec(mode=mode,loss=loss,eval_metric_ops=eval
_metric_ops)
```

Run the tensorflow model

This section will use the model defined by the student and run the training and evaluation step

In [38]:
```
#X_train = np.array((X_train/255.0),dtype=np.float16)
#X_test = np.array((X_test/255.0), dtype=np.float16)
X_train = np.array((X_train/255.0),dtype=np.float32)
X_test = np.array((X_test/255.0), dtype=np.float32)

pets_classifier = tf.estimator.Estimator(model_fn=cnn_model_fn, model_dir="/tmp/pets_convnet_model")
#pets_classifier = tf.estimator.Estimator(model_fn=cnn_model_fn)
tensors_to_log = {"probabilities": "softmax_tensor"}
logging_hook = tf.train.LoggingTensorHook(tensors=tensors_to_log, every_n_iter=50)
train_input_fn = tf.estimator.inputs.numpy_input_fn(x={"x": X_train}, y=y_train, batch_size=10,
                                                    num_epochs=None, shuffle=True)
pets_classifier.train(input_fn=train_input_fn, steps=num_steps, hooks=[logging_hook])
eval_input_fn = tf.estimator.inputs.numpy_input_fn(x={"x": X_test}, y=y_test, num_epochs=1,shuffle=False)
eval_results = pets_classifier.evaluate(input_fn=eval_input_fn)
print(eval_results)
```

```
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir': '/tmp/pets_convnet_model', '_tf_
random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': No
ne, '_save_checkpoints_secs': 600, '_session_config': allow_soft_placement: t
rue
graph_options {
  rewrite_options {
    meta_optimizer_iterations: ONE
  }
}
, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_s
tep_count_steps': 100, '_train_distribute': None, '_device_fn': None, '_proto
col': None, '_eval_distribute': None, '_experimental_distribute': None, '_exp
erimental_max_worker_delay_secs': None, '_service': None, '_cluster_spec': <t
ensorflow.python.training.server_lib.ClusterSpec object at 0x00000185EE33ED30
>, '_task_type': 'worker', '_task_id': 0, '_global_id_in_cluster': 0, '_maste
r': '', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0,
'_num_worker_replicas': 1}
INFO:tensorflow:Calling model_fn.
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow
\python\ops\losses\losses_impl.py:121: add_dispatch_support.<locals>.wrapper
(from tensorflow.python.ops.array_ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow
\python\training\monitored_session.py:875: start_queue_runners (from tensorfl
ow.python.training.queue_runner_impl) is deprecated and will be removed in a
future version.
Instructions for updating:
To construct input pipelines, use the `tf.data` module.
INFO:tensorflow:Saving checkpoints for 0 into /tmp/pets_convnet_model\model.c
kpt.
INFO:tensorflow:probabilities = [[0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]]
INFO:tensorflow:loss = 0.6931472, step = 1
INFO:tensorflow:probabilities = [[0.49480388 0.50519615]
 [0.49108353 0.50891644]
 [0.4973728  0.5026272 ]
 [0.49274912 0.50725085]
 [0.49115747 0.5088425 ]
 [0.49251896 0.50748104]
 [0.4934318  0.50656825]
```

```
 [0.4939052  0.50609475]
 [0.49666563 0.50333446]
 [0.4941389  0.50586104]] (1.140 sec)
INFO:tensorflow:global_step/sec: 48.2674
INFO:tensorflow:probabilities = [[0.5030735  0.4969265 ]
 [0.49653295 0.50346714]
 [0.4993669  0.50063306]
 [0.5002673  0.49973264]
 [0.4997159  0.50028414]
 [0.499534   0.5004659 ]
 [0.4982659  0.50173414]
 [0.49566975 0.5043302 ]
 [0.49798205 0.5020179 ]
 [0.49612188 0.5038782 ]] (0.932 sec)
INFO:tensorflow:loss = 0.69341576, step = 101 (2.071 sec)
INFO:tensorflow:probabilities = [[0.5050895  0.49491045]
 [0.5038395  0.49616048]
 [0.5069888  0.4930112 ]
 [0.5037201  0.49627984]
 [0.5024145  0.49758548]
 [0.5059059  0.49409404]
 [0.50502616 0.49497387]
 [0.5052998  0.4947002 ]
 [0.5049982  0.49500182]
 [0.5033504  0.4966496 ]] (1.026 sec)
INFO:tensorflow:global_step/sec: 51.3402
INFO:tensorflow:probabilities = [[0.50353676 0.4964632 ]
 [0.5041223  0.49587768]
 [0.50378615 0.49621385]
 [0.5038401  0.4961599 ]
 [0.50371236 0.4962877 ]
 [0.50487524 0.4951248 ]
 [0.50454867 0.49545136]
 [0.5045616  0.49543834]
 [0.5045541  0.49544594]
 [0.5043814  0.49561864]] (0.922 sec)
INFO:tensorflow:loss = 0.69304496, step = 201 (1.948 sec)
INFO:tensorflow:probabilities = [[0.49397385 0.50602615]
 [0.49468827 0.5053117 ]
 [0.49475884 0.5052411 ]
 [0.49467766 0.5053223 ]
 [0.49719498 0.5028051 ]
 [0.49355212 0.50644785]
 [0.4936044  0.5063956 ]
 [0.49534893 0.504651  ]
 [0.495525   0.50447494]
 [0.49677694 0.50322306]] (0.935 sec)
INFO:tensorflow:Saving checkpoints for 300 into /tmp/pets_convnet_model\mode
l.ckpt.
INFO:tensorflow:Loss for final step: 0.69675654.
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2019-10-06T22:28:38Z
INFO:tensorflow:Graph was finalized.
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow
\python\training\saver.py:1276: checkpoint_exists (from tensorflow.python.tra
ining.checkpoint_management) is deprecated and will be removed in a future ve
```

```
rsion.
Instructions for updating:
Use standard file APIs to check for files with this prefix.
INFO:tensorflow:Restoring parameters from /tmp/pets_convnet_model\model.ckpt-
300
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-10-06-22:28:38
INFO:tensorflow:Saving dict for global step 300: accuracy = 0.5, global_step
= 300, loss = 0.69328046
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 300: /tmp/pe
ts_convnet_model\model.ckpt-300
{'accuracy': 0.5, 'loss': 0.69328046, 'global_step': 300}
```

In [ ]: