PROJECT: Pet Classification Tensorflow Model Using CNN

Project Objective

To build a CNN model that classifies the given pet images correctly into dog and cat images.

The project scope document specifies the requirements for the project “Pet Classification Model Using CNN.” Apart from specifying the functional and nonfunctional requirements for the project, it also serves as an input for project scoping.

Project Description and Scope

We are provided with the following resources that can be used as inputs for your model:

A collection of images of pets, that is, cats and dogs. These images are of different sizes with varied lighting conditions. Code template containing the following code blocks: a. Import modules (part 1) b. Set hyper parameters (part 2) c. Read image data set (part 3) d. Run TensorFlow model (part 4) You are expected to write the code for CNN image classification model (between Parts 3 and 4) using TensorFlow that trains on the data and calculates the accuracy score on the test data.

Project Guidelines

Begin by extracting ipynb file and the data in the same folder. The CNN model (cnn_model_fn) should have the following layers: ● Input layer ● Convolutional layer 1 with 32 filters of kernel size[5,5] ● Pooling layer 1 with pool size**[2,2] **and stride 2 ● Convolutional 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 ● Dropout layer with dropout probability 0.4 Predict the class by doing a softmax on the output of the dropout lay bold text

This should be followed by training and evaluation: For the training step, define the loss function and minimize it ● For the evaluation step, calculate the accuracy Run the program for 100, 200, and 300 iterations, respectively. Follow this by a report on the final accuracy and loss on the evaluation data. Prerequisites To execute this project, refer to the installation guide in the downloads section of LMS.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
```

```
import cv2                # working with, mainly resizing, images
import numpy as np        # dealing with arrays
import os                 # dealing with directories
from random import shuffle # mixing up or currently ordered data that might lead our netwo
from tqdm import tqdm


TRAIN_DIR = '/content/drive/MyDrive/PETDB/train'
TEST_DIR = '/content/drive/MyDrive/PETDB/test'
IMG_SIZE = 224
LR = 1e-3

MODEL_NAME = 'pet_classifier_2_Conv_basic'


def label_img(folder_name):

    # conversion to one-hot array [cat,dog]
    #                            [much cat, no dog]
    if folder_name == 'cats': return [1,0]
    #                             [no cat, very doggo]
    elif folder_name == 'dogs': return [0,1]


def create_train_data():
    training_data = []
    for n,folder in enumerate(os.listdir(TRAIN_DIR)):
      images = os.listdir(os.path.join(TRAIN_DIR,folder))
      for i,image in enumerate(images):
        label = label_img(folder)
        path = os.path.join(TRAIN_DIR,folder,image)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data


X = create_train_data()
```

```
/usr/local/lib/python3.7/dist-packages/numpy/lib/npyio.py:528: VisibleDeprecationWar
  arr = np.asanyarray(arr)
```

```
os.getcwd()
```

```
'/content'
```

```
def create_test_data():
    test_data = []
    for n,folder in enumerate(os.listdir(TEST_DIR)):
```

```
        images = os.listdir(os.path.join(TEST_DIR,folder))
        for i,image in enumerate(images):
          label = label_img(folder)
          path = os.path.join(TEST_DIR,folder,image)
          img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
          img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
          img = img/255
          test_data.append([np.array(img),np.array(label)])
      shuffle(test_data)
      np.save('test_data.npy', test_data)
      return test_data


Y = create_test_data()
```

```
/usr/local/lib/python3.7/dist-packages/numpy/lib/npyio.py:528: VisibleDeprecationWar
  arr = np.asanyarray(arr)
```

Model Specification

The CNN model (cnn_model_fn) should have the following layers:

- Input layer
- Convolutional layer 1 with 32 filters of kernel size[5,5]
- Pooling layer 1 with pool size**[2,2] **and stride 2
- Convolutional 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
- Dropout layer with dropout probability 0.4 Predict the class by doing a softmax on the output of the dropout lay bold text

```
!pip install tflearn
```

```
Requirement already satisfied: tflearn in /usr/local/lib/python3.7/dist-packages (0.
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from t
```

```
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression

#Input layer
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

#Convolutional layer 1 with 32 filters of kernel size[5,5]
```

```
convnet = conv_2d(convnet, 32, 5, activation='relu')

#Pooling layer 1 with pool size**[2,2] **and stride 2
convnet = max_pool_2d(convnet, 2,strides = 2)

#Convolutional layer 2 with 64 filters of kernel size[5,5]
convnet = conv_2d(convnet, 64, 5, activation='relu')

#Pooling layer 2 with pool size[2,2] and stride 2
convnet = max_pool_2d(convnet, 2,strides = 2)

#Dense layer whose output size is fixed in the hyper parameter: fc_size=32
convnet = fully_connected(convnet, 32, activation='relu')

#Dropout layer with dropout probability 0.4 Predict the class by doing a softmax on the ou
convnet = dropout(convnet, 0.4)
convnet = fully_connected(convnet, 2, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_crosse
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow/python/com
Instructions for updating:
non-resource variables are not supported in the long term
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tflearn/initializatio
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the const
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow/python/uti
Instructions for updating:
Use tf.initializers.variance_scaling instead with distribution=uniform to get equiva
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tflearn/initializatio
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the const
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow/python/uti
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_pro
```

```
train = X
test = Y
```

```
X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
test_y = [i[1] for i in test]
```

```
history= model.fit({'input': X}, {'targets': Y}, n_epoch=300, validation_set=({'input': te
    snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
| Adam | epoch: 186 | loss: 11.51489 - acc: 0.4972 | val_loss: 11.51293 - val_acc
--
Training Step: 187  | total loss: 11.51470 | time: 1.117s
| Adam | epoch: 187 | loss: 11.51470 - acc: 0.4975 | val_loss: 11.51293 - val_acc
--
```

```
Training Step: 188  | total loss: 11.51452 | time: 1.116s
| Adam | epoch: 188 | loss: 11.51452 - acc: 0.4977 | val_loss: 11.51293 - val_acc
--
Training Step: 189  | total loss: 11.51436 | time: 1.114s
| Adam | epoch: 189 | loss: 11.51436 - acc: 0.4979 | val_loss: 11.51293 - val_acc
--
Training Step: 190  | total loss: 11.51422 | time: 1.123s
| Adam | epoch: 190 | loss: 11.51422 - acc: 0.4982 | val_loss: 11.51293 - val_acc
--
Training Step: 191  | total loss: 11.51409 | time: 1.119s
| Adam | epoch: 191 | loss: 11.51409 - acc: 0.4983 | val_loss: 11.51293 - val_acc
--
Training Step: 192  | total loss: 11.51397 | time: 1.118s
| Adam | epoch: 192 | loss: 11.51397 - acc: 0.4985 | val_loss: 11.51293 - val_acc
--
Training Step: 193  | total loss: 11.51387 | time: 1.115s
| Adam | epoch: 193 | loss: 11.51387 - acc: 0.4987 | val_loss: 11.51293 - val_acc
--
Training Step: 194  | total loss: 11.51377 | time: 1.119s
| Adam | epoch: 194 | loss: 11.51377 - acc: 0.4988 | val_loss: 11.51293 - val_acc
--
Training Step: 195  | total loss: 11.45785 | time: 1.114s
| Adam | epoch: 195 | loss: 11.45785 - acc: 0.5014 | val_loss: 11.51293 - val_acc
--
Training Step: 196  | total loss: 11.46336 | time: 1.112s
| Adam | epoch: 196 | loss: 11.46336 - acc: 0.5013 | val_loss: 11.51293 - val_acc
--
Training Step: 197  | total loss: 11.46832 | time: 1.120s
| Adam | epoch: 197 | loss: 11.46832 - acc: 0.5011 | val_loss: 11.51293 - val_acc
--
Training Step: 198  | total loss: 11.47278 | time: 1.124s
| Adam | epoch: 198 | loss: 11.47278 - acc: 0.5010 | val_loss: 11.51293 - val_acc
--
Training Step: 199  | total loss: 11.47679 | time: 1.111s
| Adam | epoch: 199 | loss: 11.47679 - acc: 0.5009 | val_loss: 11.51293 - val_acc
--
Training Step: 200  | total loss: 11.48040 | time: 1.112s
| Adam | epoch: 200 | loss: 11.48040 - acc: 0.5008 | val_loss: 11.51293 - val_acc
--
Training Step: 201  | total loss: 11.48366 | time: 1.121s
| Adam | epoch: 201 | loss: 11.48366 - acc: 0.5007 | val_loss: 11.51293 - val_acc
--
Training Step: 202  | total loss: 11.48658 | time: 1.117s
| Adam | epoch: 202 | loss: 11.48658 - acc: 0.5007 | val_loss: 11.51293 - val_acc
--
Training Step: 203  | total loss: 11.48922 | time: 1.114s
| Adam | epoch: 203 | loss: 11.48922 - acc: 0.5006 | val_loss: 11.51293 - val_acc
--
Training Step: 204  | total loss: 11.49333 | time: 1.115s
| Adam | epoch: 204 | loss: 11.49333 - acc: 0.4980 | val_loss: 11.51293 - val_acc
--
Training Step: 205  | total loss: 11.49529 | time: 1.118s
```

```
history
```

```
model.save('/content/drive/MyDrive/PETDB/{}'.format(MODEL_NAME))
```

```
INFO:tensorflow:/content/drive/MyDrive/PETDB/pet_classifier_2_Conv_basic is not in a
```

✓ 0s completed at 11:53