CS 237B: Principles of Robot Autonomy II
Section 2: Do turtles play chess?

They probably don't. But they do read numbers.

Our goals for this week's section:

1. Revisit convolutional neural network based object recognition
2. Modify Turtlebot's detector to detect digits instead of food
3. Search specific digits in the world (autonomous "digit" delivery, maybe?)

By the end of the section, you should have your Turtlebot navigate around to search a specific digit. The TAs will check off your group as soon as you can show this demo.

1 Background Information

Digit recognition is a very popular problem in machine learning. The well-known MNIST [1] dataset consists of handwritten digit images, and is often the playground of machine learning / computer vision researchers. Needless to say, it is very well-explored.

The problem we are interested in is a little more complex: We want to both detect and recognize digits. That is, the input to our system will not just be a digit image. Instead, it will be a regular photo. Our system will tell whether there is one or more digits in it, and if yes, then it will also tell what the digits are.

A practical application of this is number plate recognition. Now, please read this blog post (which is based on [2]) to get familiar with what you are going to work on.

Problem 1: What are some differences between number plate and digit recognition?

Problem 2: Remember "brute force classification" and "convolutional K×K classification" from Homework 1. How is the method described in the blog post similar to either (or both) of those?

2 Environment Setup

As you just read in the blog post, the training for automatic number plate recognition takes around 6 hours on a GPU. Plus, it requires a wide variety of background images for generalizability, which is around 36 GBs. Furthermore, even though Turtlebot has its own GPU, its memory and speed are quite limited. Luckily, we are not trying to do number plate recognition. In our case, we will only read one-digit numbers. This will make the output dimensionality smaller. Besides, we will not need as many feates as we would for number plate recognition.

We already simplified the deep neural network model. Realizing training still takes a considerable amount of time, we even trained the network for you! What you are going to do is just to plug the network in Turtlebot's detector.

Let's start with downloading the model. First, ssh into the robot:

```
ssh aa274@<robot name>.local
```

and enter the password. Then, navigate to where the detector files are:

```
cd ~/catkin_ws/src/<your project folder name>/scripts/
```

The commands below (i) download all the required files in a repository, (ii) move them to the relevant folders, and (iii) delete the repository directory.

IMPORTANT: Be very careful with the last line to avoid disasters! Specifically, make sure you write "CS237B_Section2". Otherwise, you may cause almost all files to be deleted from the robot!

```
git clone http://github.com/PrinciplesofRobotAutonomy/CS237B_Section2.git
mv CS237B_Section2/* .
mv anpr_weights.npz ../tfmodels/anpr_weights.npz
rm -rf CS237B_Section2
```

Now you have everyhing you need! Have a look at anpr_model.py, which is also in your current directory now, to understand the neural network structure.

Problem 3: What is the output size of the neural network? Do not write it in terms of variables, but give an integer value. Why is it so? HINT: You will need to take a look at anpr_common.py.

Finally, have a look at anpr_detect.py, which includes the functionality to detect and recognize digits. In particular, look at the DigitDetector class. It has two methods: initializer, and "detect". The initializer takes two arguments: "weights_file" and "min_score". The former determines from which file the neural network weights will be loaded, and the latter is a threshold to decide whether the system detected a digit or not. For the "detect" method, the only argument is the color image in the OpenCV format in which the system will look for digits. These will be important later.

3 Plugging the Network in

Now we are going to add digit detection and recognition functionality to Turtlebot. For that, we will edit "detector_mobilenet.py". Make sure you keep a copy of the old file in case you run into issues:

```
cp detector_mobilenet.py detector_mobilenet.old
```

You can modify "detector_mobilenet.py" with your favorite editor locally or on the robot. First, add the following import statements at the end of the imports:

```
from anpr_detect import DigitDetector
from anpr_common import CHARS
```

The next two lines will come after the definition of PATH_TO_LABELS. They (i) keep the path to the trained neural network weights, and (ii) suppress Tensorflow information messages (optional):

```
PATH_TO_ANPR_WEIGHTS = os.path.join(os.path.dirname(os.path.realpath(__file__)),
    '../tfmodels/anpr_weights.npz')
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '1'
```

Finally, after USE_TF definition, add

```
RECOGNIZE = 'digits' # set this either 'digits' or 'objects'
```

Previously, the code consisted of two states: USE_TF or not. USE_TF would run MobileNet, whereas a simple detection algorithm is applied to detect stop signs when USE_TF is False. Now, we will have three states:

- not USE_TF: Same as before (stop sign detector).
- USE_TF and RECOGNIZE == 'objects': Same as before (MobileNet object detector).
- USE_TF and RECOGNIZE == 'digits': The new DigitDetector is going to run.

Previously, the MIN_SCORE variable was set to be 0.5 for MobileNet, because if one class (among many) gets a probability value above 0.5, then it is probably the correct label. For DigitDetector, we have a value that indicates the probability of a digit being present in the image. Change this value properly.

Problem 4: Will you change the MIN_SCORE to be higher or lower than 0.5**? Why? HINT: Think of whether it is easier to classify hundreds of object classes than to classify digit / non-digit images.**

At this point, you have the following three variables set:

- RECOGNIZE keeps 'digits'.
- CHARS keeps '0123456789'. So CHARS[class_id] will give you the string associated with the class_id.
- PATH_TO_ANPR_WEIGHTS keeps the path to the neural network weights.

Now, you are going to modify the rest of the file to enable digit detection & recognition functionality. You will need to modify only the following three functions:

- "load_object_labels"
- "__init__" of Detector
- "run_detection" of Detector

Hint 1: You should NOT remove anything. You will be just adding your code for the case: USE_TF and RECOGNIZE == 'digits'.
Hint 2: The "__init__" method will just need to initialize the DigitDetector instance. Then, "run_detection" will just need to use that instance. Other than modifying if-else statements, these should take 1 line each.

Problem 5: Do you need to use the "filter" method while detecting digits in "run_detection"? Why or why not?

Once you are done, you can first launch ROS on the robot

```
roslaunch asl_turtlebot turtlebot3_bringup_jetson_pi.launch
```

and then locally run the following two commands (after you download detector_viz.py) to check if your digit detection & recognition system works.

```
rostb3
python detector_viz.py
```

If everything works, you can remove your backup file in the "scripts" folder:

```
rm -f detector_mobilenet.old
```

4 Autonomous Digit Delivery

Finally, repeat your AA274A demo with the digit of your choice (instead of donuts, cakes, etc.). Show the demo to the TAs and they will check off your group. This is it, well done!

References

[1] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[2] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint arXiv:1312.6082*, 2013.