Antariksh Project
2nd Dec 2019 - 10 Jan 2020

Task 1: Type of waste detection

Since labelled images were unavailable, the Trashnet dataset was used. Trashnet consists of the following classes:
1) cardboard
2) glass
3) metal
4) paper
5) trash
Resnet34 was used to train this model. However, this process could not be extended to our case because of the lack of images for our classes. The images in trashnet were of objects of a single class with a plain white background.
Howhttps://docs.google.com/document/d/1m0oo1W95kD8okTN_vLk9ujoQ62R8Z6zwdRFMkOXsPJI/edit?usp=sharingever, in our case, the object categories found in the dustbin were not exclusive and the background was that of a dustbin. Also, the objects were occluded. Hence, due to the lack of images of the kind to be used for detection and the difference in the nature of images in the pre-existing data set, this model could not be used.
https://towardsdatascience.com/how-to-build-an-image-classifier-for-waste-sorting-6d11d3c9c478

Task 2: Depth of Garbage Detection

For this task top view images were used (around 8 were available). Due to very few images being available, the classical CV approach was adopted.

Approach 1:

- A single image was taken and read. It was thresholded to get garbage boundary as a mass of white pixels
- Erosion and dilation were done on the image for a couple of iterations so that stray white patches disappear.
-
- The white pixel mask was fit with minimum area bounding rectangle
- The edges of the brain work extracted using a coloured filter and a similar box was fit
- Since the dimensions of the bin were known the depth of inner box could be calculated using similarity of triangles.

However this approach could not be scaled since it was very image specific.
https://www.pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/
https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/
Approach 2:
The idea was to stick a tape on the insides of the bin so that we could develop a relation between the total length of the tape and the length visible in the image using the total pixel visible when bin is empty versus the number visible when there is garbage in the bin. However this approach is not feasible and hence the Idea was dropped.

Task 3: Placemaking - To check if the bin is in place or not

Approach 1 :

A classical CV approach was adopted initially to detect the presence of yellow lines. The idea was to detect the yellow placemaking lines and the base of the box and check if the base is outside the yellow box. Placemaking also consists of concrete walls bounding three sides of the bin. However, the yellow pixels could not be extracted completely in all images without changing some parameters for every image. This was because the shade of yellow in every image differed due to the lighting and other factors. Thus all filters used were very image specific and could not be generalised to all images.

Approach 2: Convolutional Neural Networks

The neural network has the following architecture:

```
(cnn):
  (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1))
  (1): Conv2d(16, 4, kernel_size=(7, 7), stride=(1, 1))
  (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (3): Tanh()
  (4): Conv2d(4, 1, kernel_size=(3, 3), stride=(1, 1))
  (5): MaxPool2d(kernel_size=2, stride=5, padding=0, dilation=1, ceil_mode=False)

 (fc):
  (0): Linear(in_features=4, out_features=2, bias=True)
  (1): Softmax()
```

Images were transformed to have size 32x32

Training was done in two phases. First with 48 images in-place augmented to 248 and 35 out-of-place images augmented to 235.

Second phase: 11 new in-place images were added and 18 new out-of-place images were added.

The total data set after both phases was of 360 in-place and 350 out-of-place images after augmentation.

Test and train accuracies were 91 %. Validation size was 20%.

https://towardsdatascience.com/how-to-train-an-image-classifier-in-pytorch-and-use-it-to-perform-basic-inference-on-single-images-99465a1e9bf5

https://medium.com/@thimblot/data-augmentation-boost-your-image-dataset-with-few-lines-of-python-155c2dc1baec

Task 4: Garbage Detection and Bin Boundary Detection

<PIPELINE>

For bin boundary detection Yolov3 with darknet framework was used.

Instructions were followed according to this article.

Labelling of images were done using Yolo Mark.
Caution: Only jpg images should be used for training and testing
Using yolo mark, all dustbins in the images were labelled using class name ‘dust bin’.
The labelled images were fed to yolo and weights were saved every 50 iterations.

Output of Yolo is <class name x1 y1 w h > . For every image input to yolo, each of the boxes detected are cropped out with few pixels margin and fed in sequence to the garbage detecting CNN.

Garbage Detecting CNN:
The neural network has the following architecture:

```
(cnn): Sequential(
   (0): Conv2d(3, 10, kernel_size=(3, 3), stride=(1, 1))
   (1): Conv2d(10, 4, kernel_size=(5, 5), stride=(1, 1))
   (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
   (3): Tanh()
   (4): Conv2d(4, 1, kernel_size=(3, 3), stride=(1, 1))
   (5): MaxPool2d(kernel_size=2, stride=5, padding=0, dilation=1, ceil_mode=False)
 )
 (fc): Sequential(
   (0): Linear(in_features=4, out_features=2, bias=True)
   (1): Sigmoid()
```

Images are transformed to have size 34x34:

- Clean : 151 (51 true + 100 augmented)
- Pileup : 151 (51 true + 100 augmented)
- Overflow : 127 (27 true + 100 augmented)
- Both : 132 (32 true + 100 augmented)

In this case, multi-label classification was done. Sigmoid was used as the last layer instead of softmax to make the probabilities of each class independent of eachother. ‘Binary Cross Entropy with logits loss’ was used as the loss function.
/
If the output of any neuron in the output layer is more than 0.5 it is considered to belong to that class. Thus any image can belong to more than one class.

To Run:

1) Change all paths to the path of folder in your system
2) Do not run parts of code that mention not to be executed
3) For placemaking garbage detection:
    - To retain the networks add images to respective folders and then re-run entire code (including not to be executed paath if results are not satisfactory, change learning rate or batch size
    - To transfer learn, set requires_grad to false and either add a few layers or choose layers and set requires_grad to true and then retrain network with additional dataset

4) To use placemaking code on an image change image path to that of required image and execute cell to get result

5) To detect bounding boxes of dustbins in an image go to terminal and run from the **darknet directory**:
   `./darknet detector test data/obj.data cfg/yolo-.cfg obj <path of weights file in backup> <path of test image file>`
6) To get state of garbage around each bin in an image, change image path in overflow detection notebook and execute cell.