Recycling Bot

Abstract

My goal with this project was to design a recycling bot that would be able to detect and classify trash from a conveyor belt and sort them into recycling buckets of paper, plastic, glass and metal.

The problem was split into two main tasks: first, to detect the object and calculate the steps to pick it up and secondly to collect visual and sensory data to classify the object.

Furthermore, the project also discusses the necessary hardware and software tools required for the recylcing bot.

1. High level architecture

The problem can be decomposed into 2 subtasks:

a. Picking up the object
b. Placing it into the correct bin

A. To pick up the object we can use the following measurements:

- Geometry of the objects
  - Width + depth to position the robot arm
  - E.g. height is important to determine how high to position the robot arm over the object before grabbing it → overhead camera is not enough
  - Multiple camera angles to capture object's dimensions in different axes
  - Theoretically, a one-sided camera is enough - if we want to determine the height of the object only from this point of view
- Material
  - Tactile information can help not only distinguishing among classes, but it's also important not to apply more force to a particular object than it can tolerate. E.g. a piece of rigid, thin piece of glass may shatter if gripping force is too high
- Deformation properties over different axes
  - A soft object may have certain angles that are more preferable during picking up
  - Try to determine convex hull and the centre of mass
  - Pick up in a position of the above predicted centre of mass of the object
- Weight
  - Important to decide the grabbing force applied, since heavier objects require larger force. The bot needs to understand how to handle fragile but heavy items.

B. To classify an object
Classification of the objects is discussed in further detail throughout this project outline.

2. Sensors

a. Multi-angle camera for visual perception
b. Weight sensor:
   i. Can be used as input feature to the bin classification problem and can be used to detect falling when the robot arm tried to pick up an object but was dropped
c. Electric conductivity sensor
d. Infrared sensor:
   i. Different materials can have different reflection properties when they are under IR lighting

   ii. Not sure which exact materials can be distinguished this way, but it might be worth a try to analyze the IR pictures and the IR spectrum of a few objects (To be decided whether IR is a sensor worth including)
e. Lidar sensor:
   i. 3D reconstruction of the images can serve well as input for our classifier.
   ii. If lidar's resolution is good enough, we can have a much more accurate representation of the 3D structure of our object than just the 2D images from conventional cameras
f. Pressure sensor
   i. To measure the squeezability of the objects
   ii. Similarly looking objects might be made of different materials and thus can react differently to squeezing.
   (E.g. shiny plastic objects might look as if they were made from some metal)
   iii. The difference in material might not be realized by only viewing the object exclusively even for a human.

3. Training Data Generation

a. Human-labeled data
   i. Make photos from different angles of objects already sorted by manual workforce - they can be treated as "golden source of truth" data with very high confidence on the labels
   ii. Overhead photos can be augmented by making rotations over the vertical axis.
b. Open-source datasets
c. Training data enhancement through GANs
   i. Generating new training samples (combination of image and other metrics)
   ii. Ensuring generated samples have different conditions
   e.g. lighting (via image translation: i.e. translate image from low-lighting conditions to highly-lit conditions)

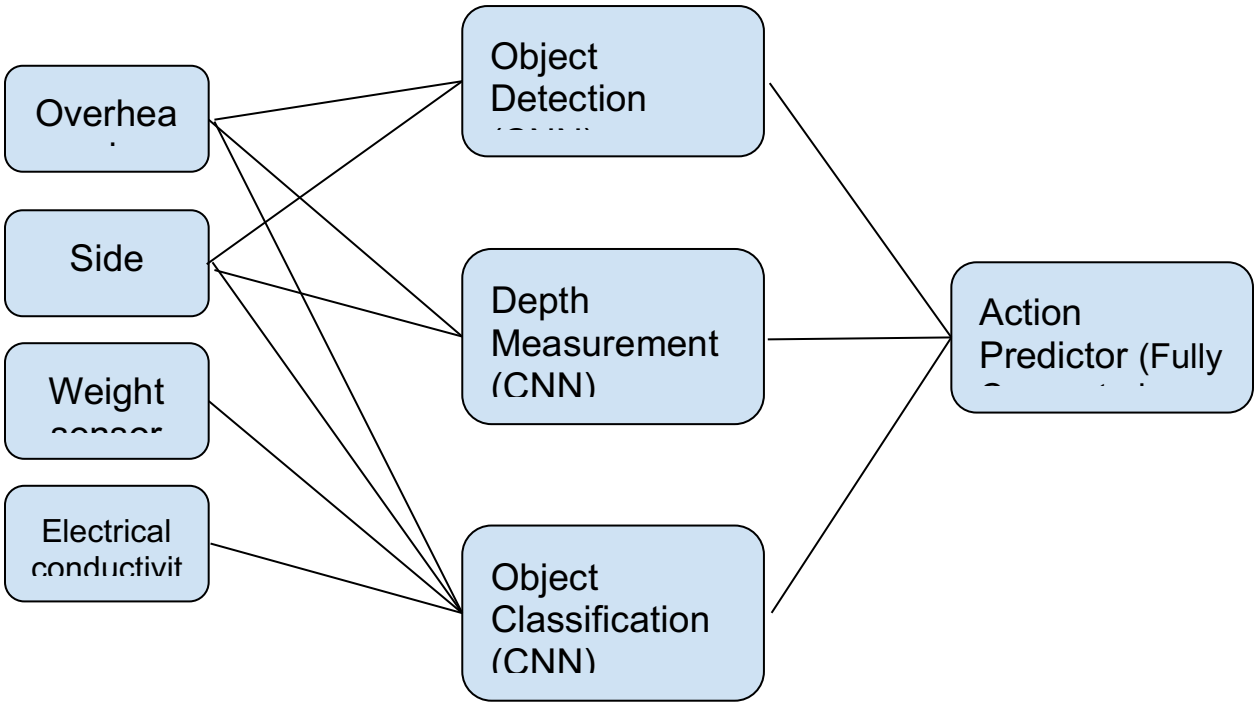
How much data is enough?

Most of the deep learning models with high representational capacity are especially data hungry.
To avoid underfitting, it is a good rule of thumb to have a few thousand labeled samples for each class

However, we can test our system before releasing it to a production environment with all the labeled data that we already have.

With cross validation, we can have a baseline

4. Architectural Components



The recycling bot is designed to receive sensory inputs from the multi-angle camera system as well as the weight sensor and electric arm. These inputs will be fed into the classifier system that is made up of multiple CNN's.

The output of the classifier system will then become the input of the recycling bot's action predictor, a fully connected, dense neural network. Here, the network will compute the bot's next move. (Pick up an object or place an object into a specified bin?)

For this, the fully connected network needs two kinds of information to make decisions about the bot's next move:

- Output of all CNN's understanding of a subcomponent of the task.

5. Modeling

We use different CNNs to learn some representation of the picture. Our goal is to condense the pixel level features with a high dimension count to a lower dimensional learned representation. This is important because these tensors will be flattened and will serve as the input for our fully connected

network. To avoid that network's parameter count exploding, we have to reduce the number of dimensions of our picture.

Our final layer is a fully connected dense neural network with a softmax output. Its output layer has 1 neuron for each possible class. These logits will be normalized to form a softmax distribution over the classes.

However, we don't return the largest probability class from the softmax distribution. First, we try to assign some confidence value to our predicted class. If the softmax distribution has a high peak for any given class, we consider the model's prediction highly confident. In that case, we return the predicted class label.
On the other hand, if the softmax distribution is close to uniform, we don't want to return our unconfident prediction. In that case, we'll classify the sample as "unknown" and return that result. This way we can keep a high precision - we decrease the ratio of incorrectly classified object. Meanwhile, we can use the "unknown" bucket for further online learning.

The CNNs and the fully connected network are trained together end-to-end.

As an alternative, we might use 1 single CNN for the different camera + LIDAR images, where the channel count is formed by the concatenation of the images. For example we have 256x256 pixel normalized images from each of our sensors. Channel 1-3 would be the RGB channels of the overhead camera, channel 4-6 would be the picture of the side camera etc. In that case, we could decrease the parameter count using a unified CNN instead of 3 separate ones.

6. Online learning

To learn "on the fly" we will utilize the unknown bucket that was "generated" by the model during modeling. All samples/objects that the model had low confidence level in are to be found in this aforementioned bucket and will be given to a human labeler along with some samples that have been classified to be one of the specified categories: glass, metal, plastic or paper, which will be chosen at random to ensure consistent labeling of the data and training throughout.

7. Metrics

To evaluate the model, a simple precision-recall metric would be used, for which the aforementioned "unknown" bin will be especially important.

For any object that the recycling bot detects and classifies, the prediction will be labeled as:

- True positive if object was detected and placed into correct bin
- False positive if object was detected and placed into incorrect bin **OR** if object was NOT detected

- True negative if the object detected does not belong to any of the specified classes/bins, i.e. glass, metal, plastic, paper and was put into the unknown bucket
- False negative if the object was detected but was not classified as any of the specified classes, i.e. glass, metal, plastic, paper and was incorrectly put into the unknown bucket.

By adding an additional “unknown” bucket to the list of possible classes, the precision of the model can be increased. Since recycling plants rely heavily on “clean” and un-tempered materials, it is especially important that all objects be placed into the correct bucket, even if some objects will be “thrown away” by incorrectly classifying them as unknown.

8. Software tools

For the recycling bot, the following libraries will be used:

- TensorFlow 2.0
- Keras API

Since Keras API follows best practices for using the TensorFlow library, it is an ideal API to use to design the neural network, allowing to re-use ready-made CNN architectures for our application, as well - e.g. VGG, resnet.

9. Hardware tools

I'd discuss training and prediction time hardware requirements separately due to the different workloads in the two stages of an ML system.

9.1 Training

For training we use deep CNNs with lots (potentially millions) of trainable parameters. To have reasonable training epoch times, we have to utilize GPU resources.
They can be used from our datacenter capacities, but if don't have such powerful hardwares on-premises, we can use one of the large cloud providers - AWS, MS Azure, Google Cloud Platform e.g.
These cloud providers grant access to the latest generations of datacenter grade GPUs which have the compute capacity and VRAM even for training the largest neural networks.
For our use case a single VM with a strong single GPU (NVIDIA V100 e.g.) might be enough.
After having a good enough result on cross-validation we might early stop the training and capture the weights of the model. E.g. as a HDF5 file in case on TensorFlow

9.2 Prediction

For prediction we can load the trained weights into our model on different hardware capable of running our model.

This hardware most likely won't have to be as powerful as the one used during training, because we use it for single instance prediction instead of large batches - we have to only predict for one or at most a few objects in the production system.
Besides classical GPUs, there are hardwares with lower cost optimized specifically for inference that might fit into a production system. This can be important especially if we try to increase our sorting system's capacity by running more sorting tables in parallel