Product: Recyclotron

Team: Group Eight

Abstract

We developed a car that moves on rails and integrated it with KNN categorisation of recyclable and non-recyclable trash.

We also designed a trapdoor and improved our classifier as we exceeded our project plan goals.

We have fulfilled our two goals for the demo:

- Recognise a metal can, and output a signal indicating metal recycling - This was fully achieved.
- Move waste item from one bin to another - This was fully achieved.

Practically our demo occurs in this sequence:

1. Detect a metal can using a camera on laptop,
2. Recognise category as metal can and hence recyclable,
3. Return recyclable or not (true or false) to EV3 system,
4. That signal moves the chamber over the left or right bin.

1. Project Plan Update

We both over-estimated and under-estimated the amount of time we needed for certain tasks. Hence, our project plan changed to reflect this and each team updated the tasks they needed to complete.

1.1. Software Team

The Software team made great progress, finishing most of the tasks we had already assigned in the project plan by the start of the week (27th January).

As team leader, Zhixing decided to allocate 3 new tasks in order to complete the first goal (Recognise a metal can, and output a signal indicating metal recycling):

1. Code a Hardware-Communication class to receive, classify and sort images,
2. Code an Auto-testing Class to test the hardware communication class,
3. Compare KNN, LR (Logistic Regression), NB (Naive Bayes) and GMM (Gaussian Mixture Model) Classifiers.

Old Tasks	*New Tasks*
Obtain images of cans and other generic waste items	*This is already done.*
Identify best metric to compare models and their feasibility in real life	*This is already done.*
Accurately identify if a can is present in a given image	*This is already done.*
Create a table which maps common objects to a recycling category	*This is already done.*
Take a photo with a webcam and output a true or false if recognised the EV3	Implemented in hardware communication class.
Not included.	Improvement of Classifier
Testing and Evaluation	*Also includes auto-tester.*

Figure 1. Comparison of old and new software tasks. Text in *italics* indicates clear differences.

1.2. Hardware Team

To tackle the second (Move waste item from one bin to another) Sakib set these tasks for the hardware team:

1. Get railings for movement guidance,
2. Make Raspberry Pi car to move onto train tracks.
3. Make trapdoor to fit within the car.
4. Write 3 scripts (Move Forward n seconds, Move Backwards n seconds, Open and Close Trapdoor w/ delay) to move the car and trapdoor,

Old Task	*New Task*
Attach wheels onto two rails	Get railings for movement guidance
Attach chamber platform in between two rails	Make Raspberry Pi car to move onto train tracks.
Not included in project plan until demo 2. As progress was made elsewhere we started this.	Make trapdoor that can fit within Raspberry Pi car
Not in project plan, but we felt we could include this as it was easy to do.	Measure, create and 3D-print a CAD model of the bin,
Programming movement of wheels over each bin	Write 3 scripts to move the car and trapdoor,
Receive arbitrary signal from laptop then move for certain duration to bin	Integrate with software team
Testing and Evaluation	Make sure we can move the system forward and backwards, open and close trapdoor, all on command via flask, ssh, scripts

Figure 2. Comparison of old and new software tasks. Text in *italics* indicates clear differences.

1.3. Group Organisation

As team leaders Zhixing and Sakib set out tasks they wanted to accomplish this week to reach the demo goals (See sec-

tions 1.1 and 1.2).

Delegation of work was primarily done by volunteering, with several people working on harder tasks. Everyone had to contribute in some way.

Zhao and Shivamm did software task 1,
Shivamm and Fraser did software task 2,
Zhixing did software task 3.
Martin, Anshul, Rebecca and Flora did hardware task 1,
Sakib and Flora took hardware task 2.
Rebecca, Flora and Martin did hardware task 3,
Anshul did hardware task 4.
David did the report, and Fraser did the presentation slides.

GitHub was our primary method of sharing and integrating code. Trello was used to track progress of tasks, whereas Facebook and Slack were used to communicate problems, questions and updates. Automated testing of the classification was done using an auto-testing class.

So far we have not spent any money from our £200 budget. This is because we are using LEGO and other pre-supplied materials.

1.4. Future Milestones

The old milestones in our project plan were:
Demo 2.1 - We can drop objects into a bin using a trapdoor,
Demo 2.2 - The hardware system and the ML auto-detection system are fully integrated,
Demo 3.1 - We can classify more trash objects,
Demo 3.2 - The user can receive feedback using monitor and lights ,
Demo 4.1 - Rubbish is compressed to reduce space,
Demo 4.2 - User can train the bin on new objects it has not seen before.

Here are the new milestones to replace the old ones:
Demo 2.1 - Structure is more stable (using almost no LEGO) with a fully constructed chamber on top of car,
Demo 2.2 - Chamber has cameras which are fully integrated (can detect and move chamber) into ML algorithms,
Demo 2.3 - Trigger system (to start detection) is fully integrated in both hardware and machine learning algorithms,
Demo 3.1 - The user can receive feedback using a monitor and lights,
Demo 4.1 - Rubbish is compressed to reduce space,
Demo 4.2 - User can train the bin on new objects it has not seen before.

In particular, 2.1 and 3.1 have already been started and will be fully completed for Demonstration 2. Demo 2's goals are expanded to more explicitly state what we hope to show.

2. Technical Details

2.1. Hardware

We had two ideas on how to move physically sort rubbish into bins; a car that moves along rails or a conveyor belt.

Our first idea was to use a conveyor belt. We would have two bins, over which we would have a conveyor belt. Firstly, the user puts rubbish on conveyor belt, once rubbish is identified the conveyor moves it into the left or right bin. The biggest issue with this idea is difficulty in extension. We would need additional mechanisms or conveyor belts to move the rubbish if we had 4 bins instead of 2.

The final and current design is a rail-track system. The car moves along two parallel railings on the edges of the bin. Unlike a conveyor belt, rails can be easily extended over additional bins.

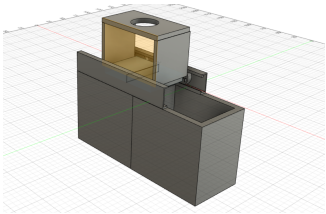

Figure 3. Picture of the CAD design model.

We started off using wheels to move our chamber because we thought that the railings would be large enough for wheels to fit inside. However, when we actually got the railings we saw that wheels do not fit.

We noticed that previous projects used rails for movement, but had "teeth" within the railings with cogs. These cogs slow the movement of the car. If the the car moves too fast, it could easily break. Hence we decided to use cogs and "toothed" rails.

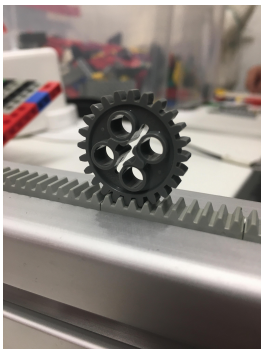

Figure 4. A cog fitting into the teeth on the rails.

Our initial idea to drop rubbish into a bin was a swinging trapdoor, where the floor of the chamber would swing open below into the bin. After realising it needed multiple motors and we couldn't figure out how to close the trapdoor afterwards, we decided against it.

We considered a bowl type door, where a bowl-shaped container would flip to empty the contents into the bin below, before rotating back into an upright position. However this would limit the size of rubbish that could be disposed of, so we decided against it.

We finally considered a drawer system, which moves using teeth via motors on either side. As the drawer moves underneath a wall, the rubbish cannot get past it and instead drops

down. This is similar to our railing system. We decided to use it as it is simple yet stable. It is able to hold objects of any size while the car moves and is easy to implement.

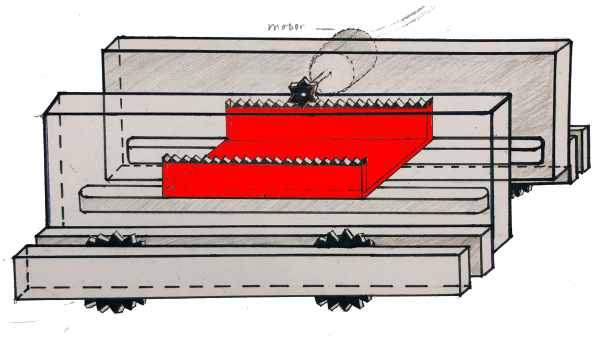

Figure 5. Moving Drawer-trapdoor highlighted in red.

2.2. Machine Learning and Software

Initially we thought about using Google's AutoML[1] vision API for classification. This is a pay-per-use cloud service where the user can submit photos and receive labels indicating which objects are present. We decided against this as the labels aren't specific enough to our recycling categories and it's too expensive.

Instead, we decided to build and use our own server to perform the calculations. Not only is it cheaper, but it is more accurate. It would cost us $0.1 per hour to handle 12k image requests on our own server (renting one high-end PC on vast.ai), whereas using Google's API would cost $1 for 1k images an hour.
This is over 100 times cheaper, and one server can be split to handle multiple bins.

Currently our model to classify an image is as follows:

1. Features are extracted from an image using a residual neural network, pre-trained on ImageNet (14M images)
2. A simpler classifier is trained on those features, using a smaller but more specific trash-image dataset (6k images).

We're using ResNet as it's highly accurate when trained on ImageNet and the feature vectors outputted can be used with a variety of classifiers.

Our current model uses ResNet50 with a KNN classifier, as it scores the highest accuracy of 89% for recyclable categorisation.

In addition, the KNN (or GMM) classifier allows us to add more training images without retraining the classifier as well as providing scores for planned confidence estimation. This will be useful for later demos where we predict an unfamiliar item's category.

See Section 3 for details of the accuracy between the 4 classifiers; KNN, LR (Logistic Regression), NB (Naive Bayes) and GMM (Gaussian Mixture Models).

There are also multiple machine learning algorithms in the scikit-learn library which could be compared with our current classifier.

We determine our recyclable and non-recyclable categories from a wide range of items. To do this we've sampled our training and testing data-set from 4 sources:

- Trashnet [3]
- imagenet [4]
- Food11 [5]
- tacoset [6]

We will also try to fine-tune the neural net on a larger data-set of common items, possibly the closest to a "large trash" data-set we can get.

3. Evaluation

3.1. Camera Detection

We have found that a camera successfully detects a metal can 7 out of 10 tests. We have found that a biscuit wrapper (non-recyclable) is likewise categorised correctly 7 out of 10 tests. We have not tested further, as this is all we are aiming for.

3.2. Hardware Testing

We have constructed our design in LEGO for the first demonstration and aim to convert it into a more stable system for the second demonstration. We feel it would be wasteful to extensively test a system which is going to change. That is why we have only tested to ensure the system fulfills the goals of moving an object from one bin to another.

3.3. Classifier Comparison

As different classifiers require different feature complexity, three tests were performed for each to search for its optimal ResNet depth. Results achieved on best depth are shown in Figure 6. They were originally output to a log file [7].

Due to a shortage of images of non-recyclable waste, food and various non-trash items are included in training and testing data-set as non-recyclable. These items would not normally be dropped in a bin - so this might inflate our accuracy to be higher than reality.

As KNN is the most accurate and balanced overall, we feel it is the best classifier out of the four.

4. Budget

As we had no costs our budget plan has not changed and remains the same as the project plan.

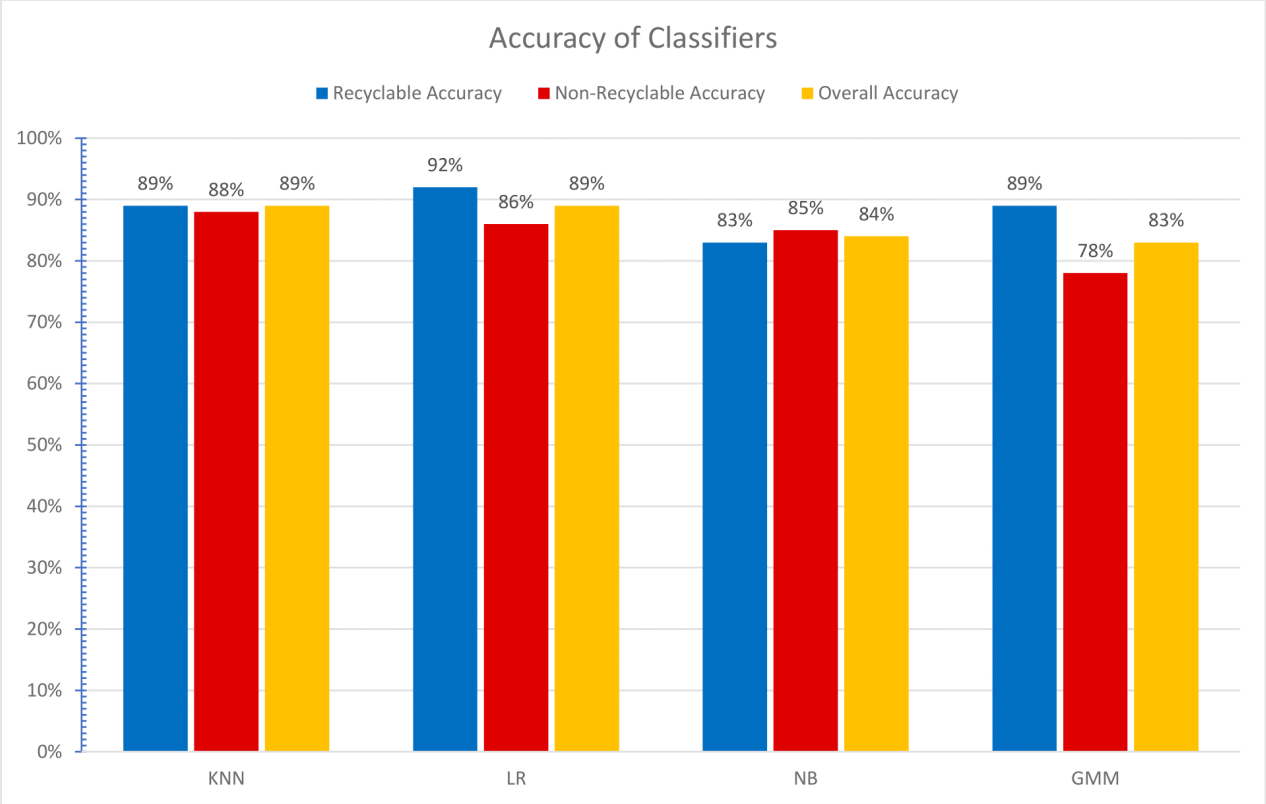

Figure 6. Comparison of classification results for three tests.

References

[1] Google Cloud Products, Cloud AutoML
Available at: https://cloud.google.com/automl/

[2] Vast.ai, Cloud GPU Rental Market
Available at: https://vast.ai/console/create/

[3] Vasant Vohra, Deep Learning based Waste Segregation
Project to classify waste images into different classes
Available at: https://github.com/vasantvohra/TrashNet

[4] trtroi, imagenet
Available at: https://www.kaggle.com/lijiyu/imagenet

[5] Md Tohidul Islam, Food11
Available at: https://www.kaggle.com/tohidul/food11

[6] Pedro Proença & Pedro Simões, Trash Annotations in Context
Available at: http://tacodataset.org/

[7] Zhixing Wang, grid1.txt
Available at: https://github.com/Sakib56/SDP-Project/blob/master/logs/grid1.txt