Trash Classification using Convolutional Neural Networks

CS5242 Neural Networks & Deep Learning

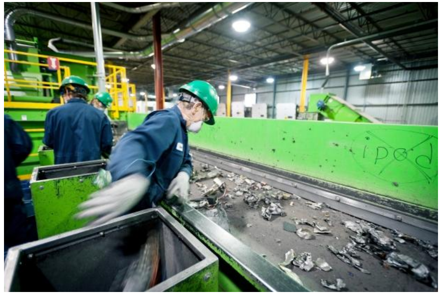
Group 03 - Akankshita, Spatika, Trinh, Ganeshkumar

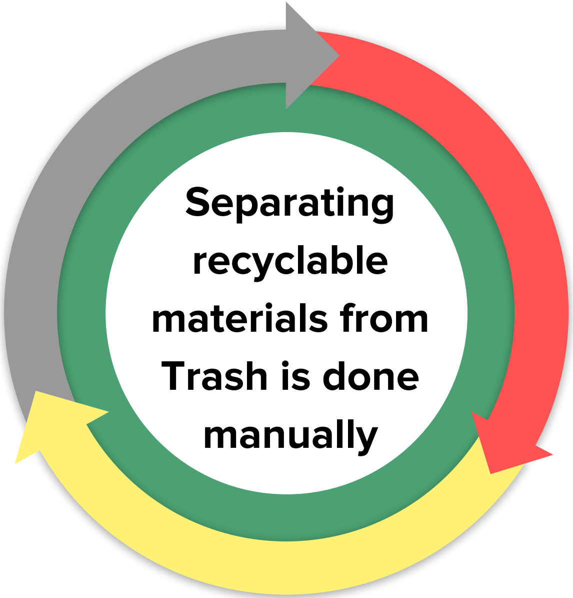
Outline

1. Problem Overview
2. Challenge and Novelty
3. Dataset
4. Model Building
5. Results and Analysis
6. Conclusion

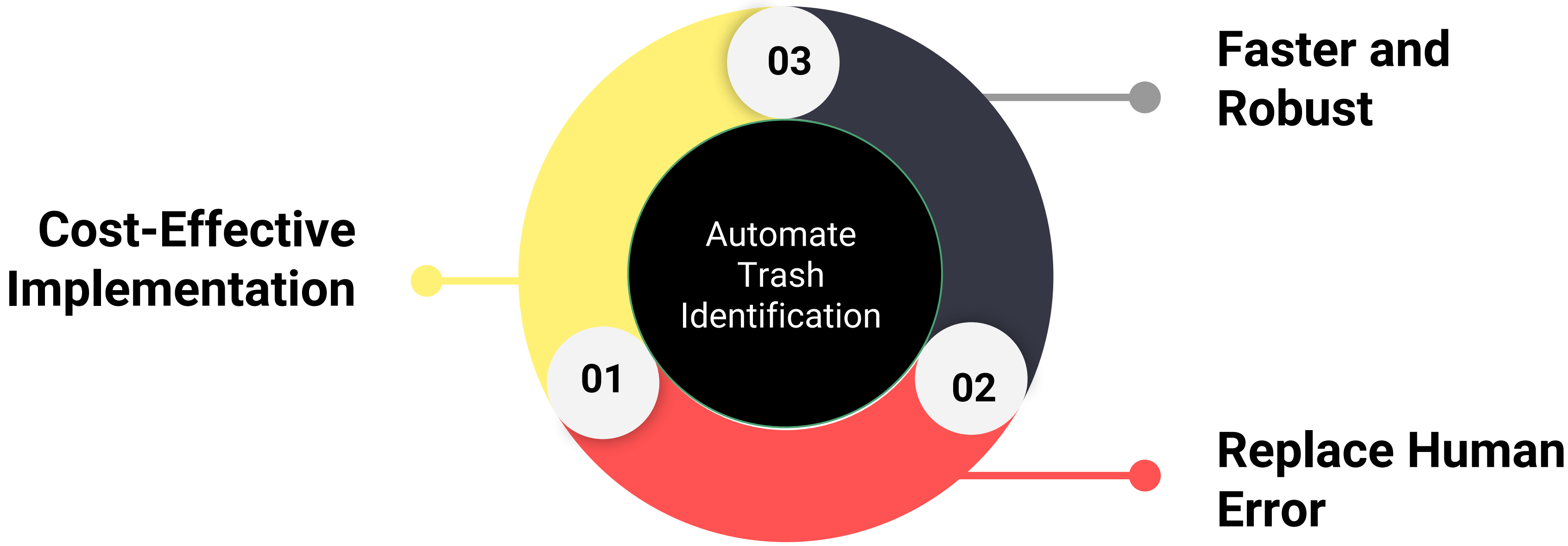
1. Problem Overview

Target Use Cases

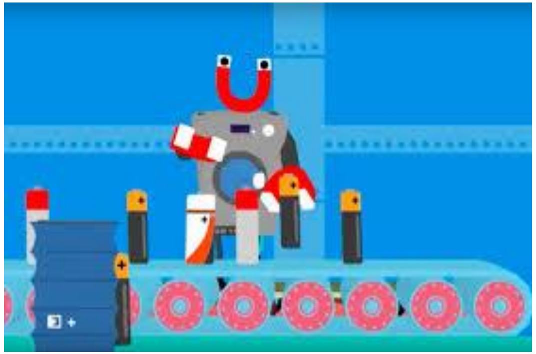




Problem Statement



Key Goals: Trash Identification by AI




Recycling Industries
Human
Earth

2. Challenge and Novelty

Challenges

- Needs to work for trash identification in real-life scenarios
  - Can be almost anything: squished/stained/damaged, obscured by other trash/spills
- Dealing with highly imbalanced dataset
  - 137 trash images vs. 400-500 for the 5 recyclable categories
- Designing CNN architecture for base models
  - Several different combinations of layers to try
- Extensive hyperparameter tuning needed for base models
  - Kernel size, padding, stride, pooling-layer parameters

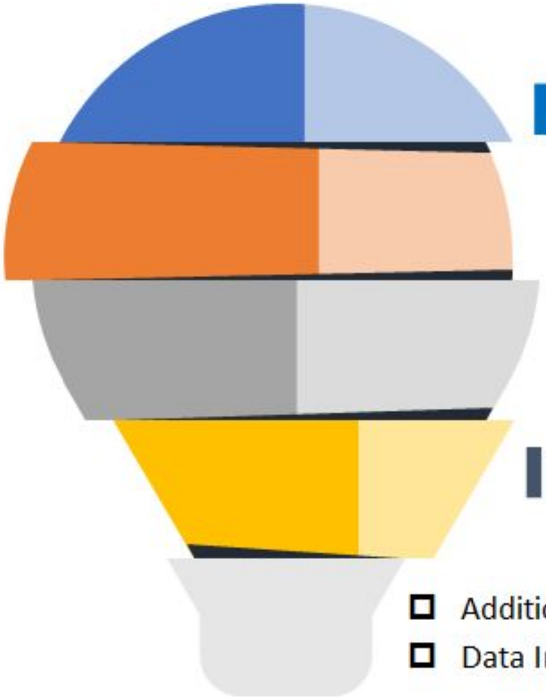
Novelty

- Adding to the original TrashNet [1] dataset
  - Tedious work. Described in Section 3.
- Not a common image classification problem due to the unlimited possibilities/challenging nature of collecting images of trash
- Few papers for reference material
- Assist in automated waste management
- SmartNation

3. Dataset

Image Dataset for Classification

Dataset Preparation



- Additional images to enrich training dataset with variety of features
- Data Image resizing inorder to bring all images of same size and shape

Additional Images

- Adapt ‘Trashnet’ to Singapore/Asian context
- Combat class-imbalance problem
- Added 30 images of trash, like these:


Data Augmentation

Training Data:

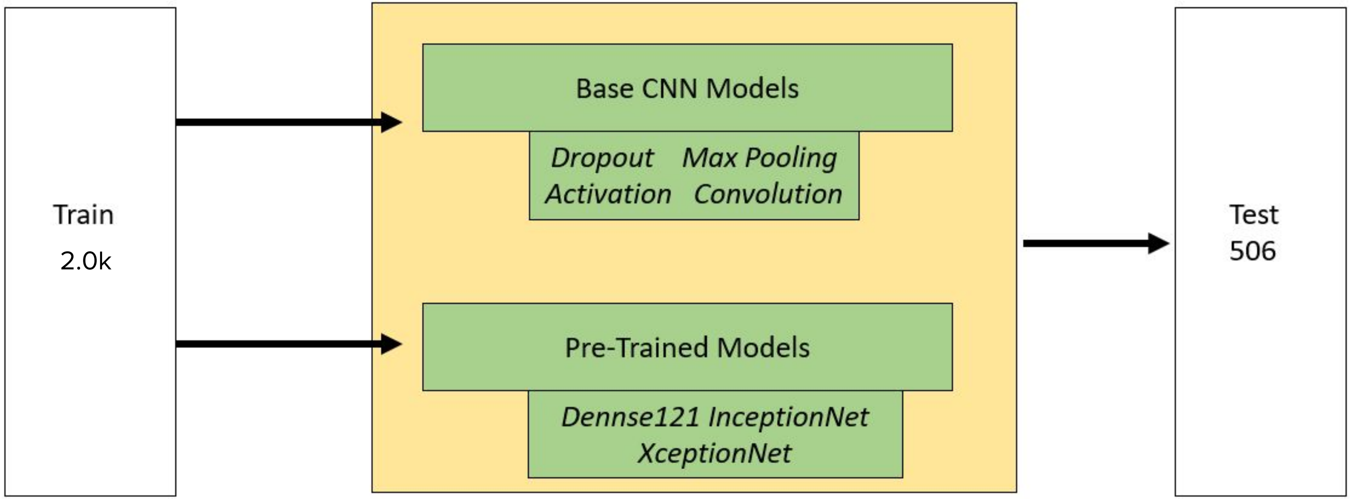
- Flipping images vertically and/or horizontally
- Cropping
- Shear
- Zoom

Test Data:

- Resizing only

4. Model Building

Modelling Process



Basic CNN Model

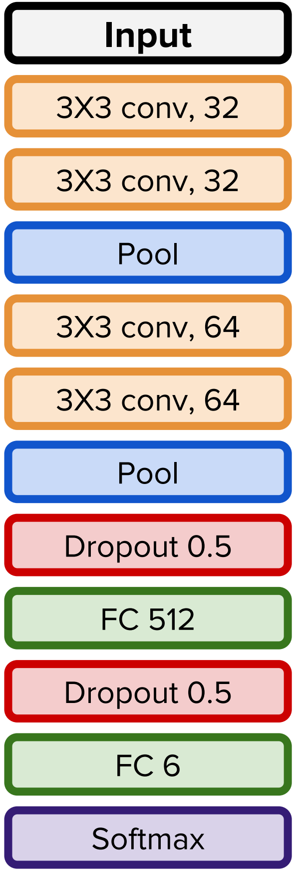
01	**Convolution**	• 4 Conv2D Layers • Input layers of 32 and 64 • Same Padding
02	**Activation**	• Non Linear Activation 'RELU' used at every Convolution block • Softmax function used for multi class classification in final layer
03	**Dropout**	• Helps to avoid Overfitting of model • Added at each convolution block • Values of 0.25, 0.5 used
04	**Max Pooling**	• Used to provide the max value in the feature map • Pool_size (2,2) is used

Architecture

```python
def getModel(num_classes):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=X_train.shape[1:]))
    model.add(Activation('relu'))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes))
    model.add(Activation('softmax'))
    return model
```



Tools/Technology

- Python3
- Keras (TensorFlow), Numpy, Pandas
- PyTorch (FastAI)
- ImageNet
- Google Colab + GPU (Tesla K80)

Fine-Tuned Pre-trained (ImageNet) Models

- **Keras**
  1. InceptionResNetV2
  2. Densenet121
  3. Xception

Optimizers: Adam, Adadelta, RMSProp

- **FastAI**
  1. Resnet34
  2. Resnet50

Optimizers: AdamW

Tuning the Model Hyperparameters

- Grid Search

- Cyclical Learning Rates (FastAI)

- Bayesian Optimization (Keras)

5. Results and Analysis

Original Results

- Thung's original SVM model achieved 63% accuracy.
- Their neural network model achieved 27% accuracy.

Our models performed significantly better.

vs Our Model Results

Architecture	**Batch**	**Optimizer**	**Epochs**	**Data Augmentation, Tuning**	**Accuracy**
AkankshitaNet	32	Adadelta	10	**Tuning**: Bayesian Opt.	23.9%
SpatikaNet	32	Adam	5	Yes - shear, zoom, horizontal flip **(updated dataset)**	**50.97%**
TrinhNet	32	Adam	10	Shear, Zoom, Horizontal,Vertical flip (**original dataset**)	25.30%
GaneshNet	32	Adam	4	Rescale, Shear, Zoom, Horizontal flip (**original**)	23.33%
InceptionNetV2	20	Adadelta	100	None	89.13%
DenseNet121	20	Adam	40	None **(original dataset)**	86.3%
XceptionNet	32	Adam	10	Shear, Zoom, Horizontal,Vertical flip (**original dataset**)	91.9%
Resnet50 (FastAI)	20	AdamW	30	Learning Rate tuned using Cyclical Learning Rate	**93.38%**

Analysis of Results

- Basic CNN without augmentation does not give optimal results on original dataset, but improves on updated dataset.
- All pre-trained models have quite good results (> 80% accuracy) compared to basic models. Thus, learning of waste sorting is feasible with modern deep learning based approaches.
- FastAI Model gives best results since it utilizes cyclical learning rates technique, which finds the optimal learning rate.
- Training with **cyclical learning rates** also achieves improved classification accuracy without a need to tune and often in fewer iterations.

6. Conclusion

Takeaways

- Data augmentation only slightly improves the result in some models (for our case)
- Pre-trained models work better than our own basic models.
- Building CNN Model from scratch needs extensive research in terms of layers architecture or hyperparameter tuning of parameters
- Diversity helps: different representation of images always helps in model training
- Data science is all about applying different ideas; there’s always room for improvement
- Larger training dataset (balanced) can increase the prediction accuracy

Future Extension

- To expand to more classification labels
- To increase the training image set
- To identify more than one object in the same frame
- Identifying particular scope of trash, example: food wastage, agro-products wastage, good yield vs. bad yield
- Stratified K-fold CV instead of just train-test split to reduce bias

Thank You!

7. Appendix

References

1. G. Thung. Trashnet. GitHub Repository, 2016.
2. Leslie N Smith. Cyclical learning rates for training neural networks. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pp. 464–472. IEEE, 2017.

Workload Split

Akankshita:

1. Base code (image loading, resizing, basic model)
2. Models: Base CNN 1 with Bayesian Opt, InceptionNetV2, FastAI
3. Presentation content

Spatika:

1. Collection of new trash images, basic data augmentation code
2. Models - Base CNN 2 with updated dataset, DenseNet121
3. Presentation content

Trinh:

1. Models - Base CNN 3 on original dataset, XceptionNet, data augmentation
2. Pretrained model code
3. Presentation content

Ganeshkumar:

1. Skeleton presentation
2. Models - Base CNN 4 on original dataset, DenseNet201, data augmentation
3. Presentation content