Deep Learning - Waste Classification

Kairan Zhong

https://github.com/ka4on/Waste-Classification

Problem Statement

Background: Waste management is a big problem in our country. Most of the wastes end up in landfills. This leads to many issues like: Increase in landfills, Land, water and air pollution…

Suppose I'm a data science intern at a waste management service …

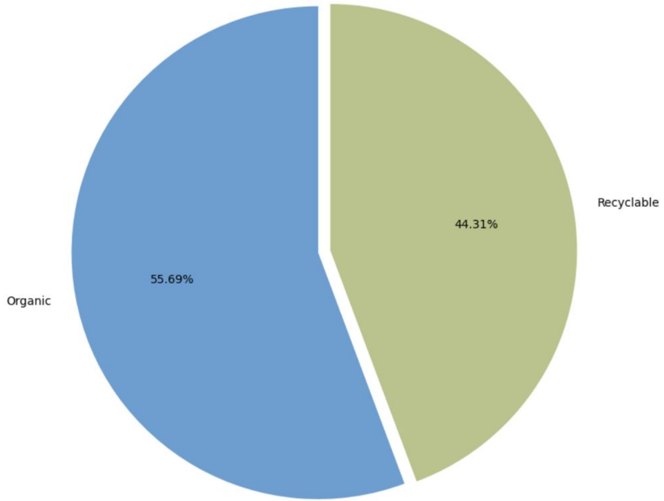
Objective: Classify organic waste and recyclable waste to Reduce toxic waste ending in landfills

Assumptions/Hypotheses

- A custom CNN network built for classification as our benchmark
- Popular models are hard to train from scratch, so we rely on transfer learning to capture features that are more particular to a specific image classification task
- We will use VGG-16 and InceptionNet to do transfer learning

EDA

- ❖ Data Source: 22500 images of organic and recyclable objects
- ❖ Data Description:
  - ➢ Training data - 22564 images
  - ➢ Test data - 2513 images
- ❖ 55.69% organic and 44.31% recyclable in training data

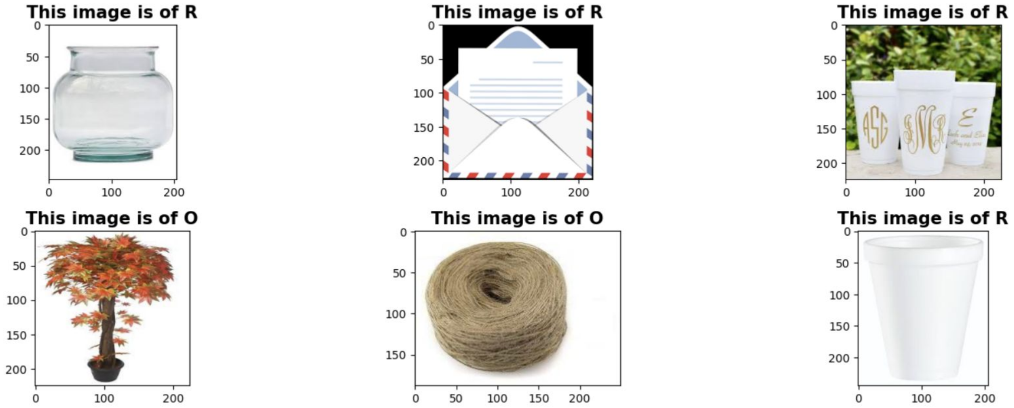


EDA

A brief overview of the classified images

‘R’ represents Recyclable

‘O’ represents Organic

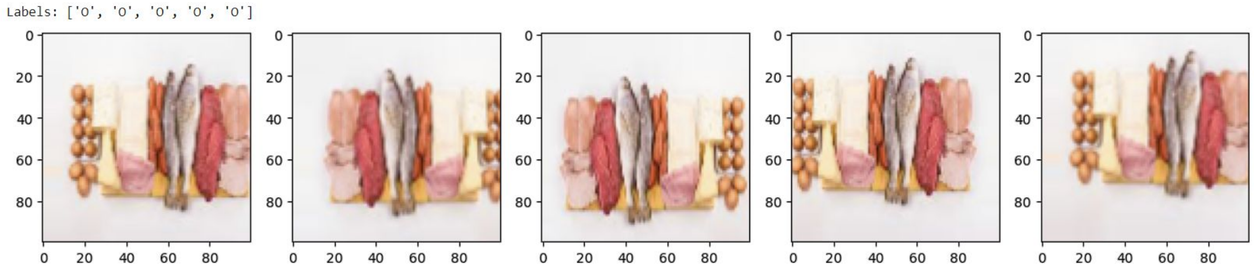


Feature Engineering & Transformations

- Load images using *ImageGenerator* Class:
  - Rescale pixel values
  - Randomly shift width and height by maximum of 0.1
  - Randomly flipping images horizontally
- Use *flow_from_directory()* to generate batches of augmented images and corresponding labels

Feature Engineering & Transformations

Look at the augmented picture samples:



The same image has been shifted and flipped randomly, each one of them is labeled as organic

Model1: Custom CNN

Architecture:

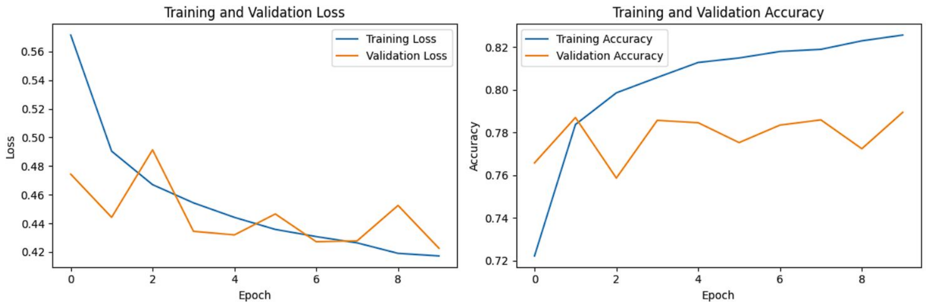
3 Convolutional layers of size 3*3 to extract features

Max pooling layers to reduce spatial dimensions of the feature maps

Dense Layer and Activation layer "relu" to learn complex patterns

Dropout layer to prevent overfitting

Final activation layer "sigmoid" for binary classification



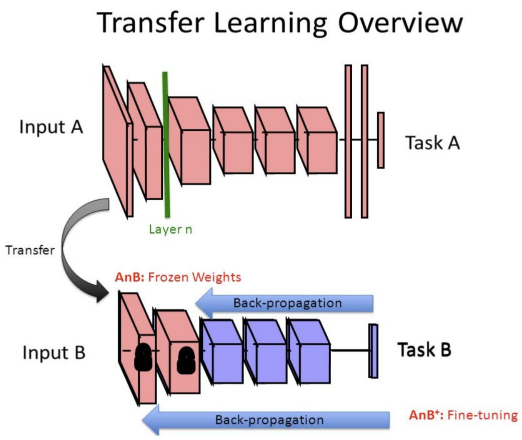
The model is a little overfitting as the train accuracy exceeds validation accuracy as epoch increases.

CNN Model

	precision	recall	f1-score	support
O	0.85	0.94	0.89	1401
R	0.91	0.79	0.84	1112
accuracy			0.87	2513
macro avg	0.88	0.86	0.87	2513
weighted avg	0.87	0.87	0.87	2513

Test Result: The test accuracy is around 0.85, not bad for our custom CNN model.

Model2: VGG 16



Transfer Learning

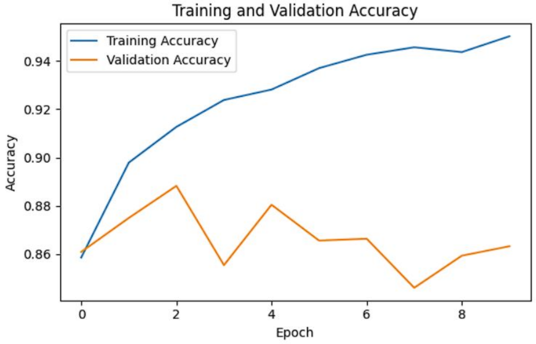
leverage a pre-trained model's weighted layers to extract generic features

Fine-tune the model on a smaller dataset specific to the target task

Benefits: require less data, save training time, and improve performance

Model2: VGG 16

- Freeze the base layer
- Create new models on top, add dropout layers for regularization



The model is a little overfitting as the train accuracy exceeds validation accuracy as epoch increases.

Architecture:

Base Layer (vgg16)

Dropout layer to prevent overfitting

Batch norm layer to stabilize learning

Dense layer for learning complex pattern/

binary classification

Extract Features Model

	precision	recall	f1-score	support
0	0.84	0.92	0.88	1401
R	0.88	0.78	0.83	1112
accuracy			0.86	2513
macro avg	0.86	0.85	0.85	2513
weighted avg	0.86	0.86	0.86	2513

Test Result: The test accuracy is around 0.87, a little improvement from previous model.

Model3: VGG 16 Fine Tuned

unfreeze convolution blocks 4 and 5 of VGG16 and add custom model on top

Architecture:

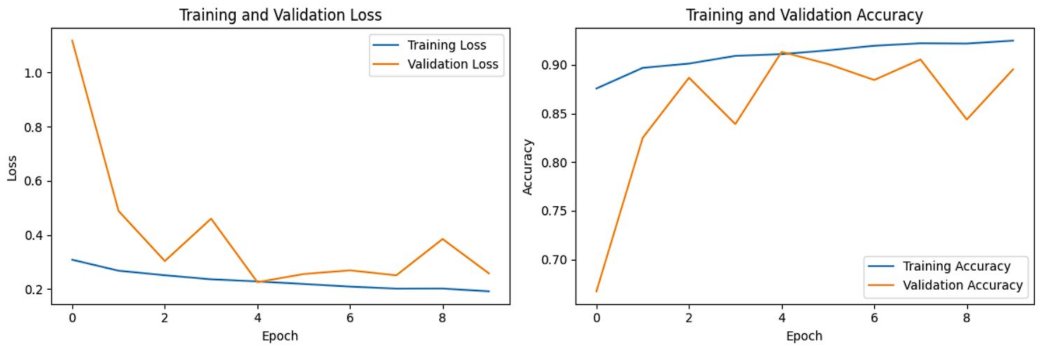
Base Layer (vgg16, block4&5 unfreeze)

Dropout layer to prevent overfitting

Batch norm layer to stabilize learning

Dense layer for learning complex pattern/

binary classification



The model does not overfit too much and has a good validation accuracy.

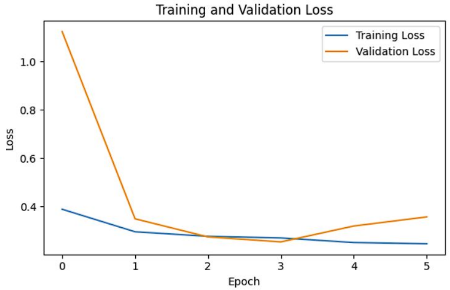
Fine-Tuned Model

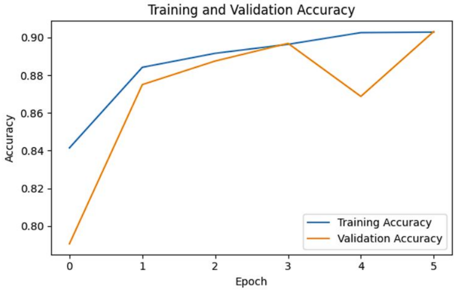
	precision	recall	f1-score	support
0	0.91	0.93	0.92	1401
R	0.91	0.89	0.90	1112
accuracy			0.91	2513
macro avg	0.91	0.91	0.91	2513
weighted avg	0.91	0.91	0.91	2513

Test Result: The test accuracy is around 0.91, improve from previous vgg16 model

Model4: InceptionNet

- Freeze the base layer
- Create new models on top, add dropout layers for regularization





The model does not overfit too much and has a good validation accuracy.

Architecture:

Base Layer (Inception)

Dropout layer to prevent overfitting

Batch norm layer to stabilize learning

Dense layer for learning complex pattern/

binary classification

Inception Model

	precision	recall	f1-score	support
O	0.86	0.97	0.91	1401
R	0.95	0.80	0.87	1112
accuracy			0.89	2513
macro avg	0.91	0.88	0.89	2513
weighted avg	0.90	0.89	0.89	2513

Test Result: The test accuracy is around 0.89, better than vgg16.

Result

Our fine-tuned model performs the best with 91% accuracy, a 6% increase in accuracy compare to custom CNN model.

Learning

Using transfer learning can help us improve the model accuracy

fine - tuning can be very useful in transfer learning if the new dataset is larger or significantly different from the original dataset

Use early stopping to avoid long training time and prevent overfitting and improve the generalization ability of a model

Future Work

1. Smaller batches and more epochs can potentially improve our model performance with faster convergence and enhanced learning of complex patterns.
2. Other pretrained models such as ResNet, VGG19 are also worth trying.
3. More complex architecture of the neural network to better capture complex patterns.

Thank You!