Multiclass classification of garbage Using a CNN approach

Fenton Reid

***Abstract*— The UK recycling process is cumbersome and inefficient with manual hand-sorting required for filtering incorrectly sorted items. This causes a bottleneck in the recycling process that can cause wrongly sorted waste to pass through. This waste has the potential of contaminating the correctly sorted recycling, resulting in the whole batch being sent to landfill. Automatic sorting of garbage is highlighted as one possible technique to improve efficiency. For this paper a start on automatic sorting is undertaken through the development of a convolutional neural network which can classify an image of waste material into one of six categories. This report covers the results of the original CNN model and its flaws that were addressed in the comparison study. Resulting in an improved CNN model being developed that uses image augmentation and dropout layers, neglected in the original model to reduce overfitting which increased the mean testing accuracy from 61% to 73%. Overall, there is significant hope that automatic sorting is feasible with further research, with a particular interest in a collaborative dataset and smart sorting bins in households.**

I. Introduction

"The UK generated 222.2 million tonnes of total waste in 2018" [1] with only 44.9% of waste in Scotland being recycled [2]. The onus of recycling is up to everyone, with the goal of preventing further harm to the environment. For this reason, even small changes in your household such as ensuring proper sorting of recyclable and non-recyclable garbage can make a huge difference. The word `garbage` is a loose term that covers a range of waste materials. For the definition of this report the term is used to refer to; cardboard, glass, metal, paper, plastic, and unusable materials termed trash. The key distinction being that all materials except trash can generally be recycled with specific restrictions for each material dependent on the recycling centers processing this waste.

One of the main issues with the recycling system is the sorting process, incorrectly sorted items must be removed by hand, and this not only slows down the recycling process but can damage equipment also [3]. This shows that there is a bottleneck in the current UK recycling system that is impacting the efficiency of recycling. By being able to automatically classify waste material into its correct category it can be passed through the recycling system more efficiently without the need for manual hand sorting. This report focusses on the development of correctly classifying waste material images into one of six categories with the specific logistics of automatic sorting left for future research.

Relevant reading was undertaken to understand the possibility of waste sorting using deep learning. A study by Atay Moltem et al. covers just this. Proposing intelligent waste sorting using deep neural networks. This paper uses transfer learning on a pre-trained Inception-v4 model, from this the neural network produced a result of 95% accuracy on the testing set that classifies waste into one of six categories [4]. This paper outlines the strengths of transfer learning and produces a very reasonable accuracy on the testing set.

The aim for this paper is to develop a deep neural network from scratch that is self-contained compared to the transfer learning methodology proposed by the intelligent waste sorting paper, that requires the use of pre-built models. With the aim of developing a model that can be better fine-tuned to the garbage classification dataset [5] with quick training times as the priority. The neural network architecture chosen for garbage classification is a convolutional neural network (CNNs), which is a type of deep learning technique that is inspired by the primary visual cortex found in mammals and a few other species [6].

II. Methodology

A. An introduction into the dataset

The required dataset is provided by Kaggle user, cchangcs and contains six categories for garbage: cardboard, glass, metal, paper, plastic, and trash, this dataset is therefore a multiclass classification problem. The required zip file can be downloaded and inside are six folders containing the images for each of the categories. The number of images per category is shown below in Figure 1.

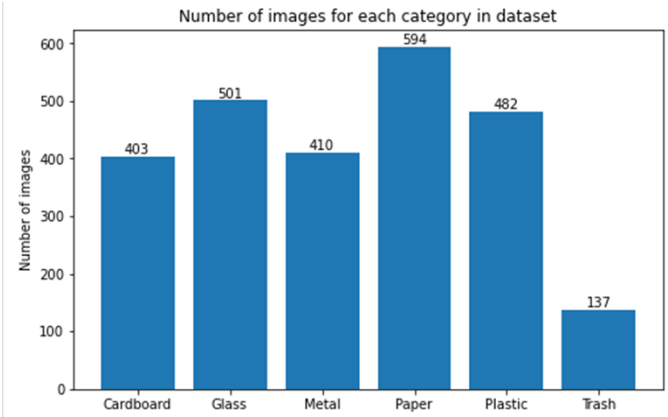

Fig. 1. Number of images for each category in the garbage collection dataset

From Figure 1 the trash category has the least number of images with a value of only 137 compared to the mean of 421 for all 2527 images. Overall, the other five categories are within an acceptable range. The input dimensions for each image in the dataset are 512x384 pixels however, the input dimensions to the CNN will be lowered for quicker training.

B. Overview of convolutional neural networks

Convolutional neural networks are very suited to working with images, this is because the underlying theory relating to CNNs is based on the biological mechanisms found in the eye. To begin to classify our input images the structure of the CNN model needs to be defined, the process and steps taken to achieve this are outlined below.

For each image in our training data the image is converted from a 3-channel jpg image into a multi-dimensional array of float numbers, normalised between 0 and 1 where each element in the array translates to the pixel value of the image. The CNN model will first apply a convolutional layer to this input array. Which involves multiplying each region of the original array with a filter that will produce a hidden layer of smaller size than that of the input layer.

Generally, an activation function is applied after the convolutional layer, the activation layer chosen for this purpose was reLU as it provides "better performance and generalization in deep learning compared to the Sigmoid and tanh activation functions" [7].

Once the activation function has been applied, a pooling layer is often applied next. With the aim of localising the features identified by the convolutional layer and to also reduce the dimension size of the hidden layer hence, reducing the number of trainable parameters required.

The application of adding a convolutional then pooling layer is often repeated multiple times. The reasoning is so that the earlier layers in the model detect small features, such as edges and lines, into more complex shapes nearer the final layers of the CNN.

Once these convolutional layers have been added a fully connected neural network is often applied at the top layer of the model to convert the convolutional data into a classification. In the case of garbage classification this would be one of six classes.

C. Framework of the application

It is planned that the open-source neural network library Keras will be used to create the CNN model. This is because Keras provides an easy-to-use API that comes pre-packaged with many useful features such as built-in training and evaluation metrics, the sequential model and more.

To verifying that the dataset images are stored in the correct NumPy array format, matplotlib and the Python Imaging Library (PIL) will be used to visualise these images and to plot the modified image augmentations that are used by the comparison study model.

D. Proposed Steps

It is proposed that the development of the application should be split up into three stages the details of which are outlined below.

1) Pre-Processing

The downloaded dataset needs to be read into the program. To do this the six folders need to be combined in code and a dedicated label needs to be given to each image for use in the validation stage of the model. A simple dictionary object variable can handle this transformation.

Each image needs to be read into the program and converted into a multi-dimensional array. The RGB values of the image range from (0 to 255) inclusive, so these values should be normalised into the 0 to 1 range.

Once the images have been processed generating the training and test data should be undertaken and training/test split decided.

2) Build the model

In this stage the general CNN model can be implemented, once achieved the model should be trained and with this the next stage of tuning hyper-parameters can begin to provide the best CNN model possible.

3) Evaluation

With the model complete and hyper-parameters tuned the training of the model can take place. The validation split and the number of epochs used can be fine-tuned by comparing the accuracy and loss metrics further aided through produced graphs. From this the performance of the model on the testing set can be undertaken to determine how well the model performs on new data, further tuning of the model undertaken if necessary.

Finally, a new dataset called predictions is included, which includes six images, one for each output class. With the aim of determining how well the model would cope with images taken in vastly different circumstances.

III. Results

A. Model diagram and input size consideration

The implemented CNN model is represented in a LeNet style diagram as shown below.

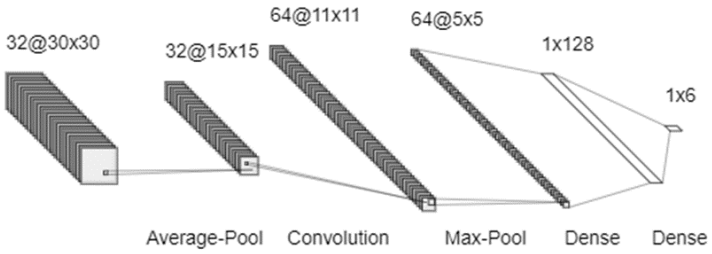

Fig. 2. CNN model architecture diagram generated with http://alexlenail.me/NN-SVG/LeNet.html

The input to the model is an array representation of a 32x32 pixel image, this is a decrease in the original size of 512x384. The decision to reduce the input image size was based on performance and the time taken to train the model. Generally, the larger the input size the more layers that are required and hence an increase in the number of trainable parameters. For this model with input dimensions of 32x32 the number of trainable parameters is around 260,000 compared to 1.6 million parameters for an input image only double the size. It becomes less feasible to fine-tune the hyper parameters if training time increases as less iterations of the CNN model can take place. For this model it takes 1 second to calculate each epoch, a quick training time.

The general structure of the CNN model followed the same steps as identified in the methodology stage of this report with the specific parameters outlined. Two convolutional and pooling layers are used firstly average pooling is applied and then in the next pooling layer max pooling from this a fully connected network was added with one hidden layer and output layer, in total this CNN model has six layers, four convolutional and two fully connected.

B. Tuning hyper-parameters

Many iterative improvements to the CNN model were made through trial and error which helped to achieve the best model possible. The reasoning behind these choices is clarified below.

1) Filter size

The filter size of the convolutional layers (convolutional and pooling layers) is increased from 32 in the first layer to 64 in the second this is because with each convolution more complex patterns are detected, and therefore a larger filter size is needed to accommodate this. It was found that the 32/64 combination produced the best results.

2) Kernel size

Optimising the kernel size was tricky, generally smaller kernel sizes reduce "computational costs and weight sharing" [8]. For this reason, the first convolutional layer had a kernel size of (3x3) identified as the most optimal out of all small kernel sizes. For the second convolutional layer as the input size is 12x12x64 (width, height, depth) the kernel size was increased to (5x5) as computationally an increase in kernel size at these dimensions does not affect the overall computation time but does increase performance.

3) Pooling layers

For the first convolution layer average pooling was used which tends to smooth out the input features. This worked better than max or min pooling for the first convolutional layer. Whereas, for the second layer maximum pooling was applied which is a standard pooling layer for CNNs and helped to identify the key features of the image.

Pooling size is also an important factor to consider for the CNN model a size of (2,2) worked best, this essentially downsizes the input by half, a large pooling size would reduce the layer dimensions two significantly and reduce accuracy as less neurons will be present in the fully connected neural network needed for prediction.

4) Activation function choice

As mentioned in the methodology, reLU was used as the activation function for the convolutional layers as it performs better than tanh or sigmoid generally. Interestingly however, sigmoid outperformed reLU when applied to the fully connected layer with around a 5% increase in training set accuracy than reLU.

5) Dataset splitting

A training test split of 70/30 worked best for the CNN model and is generally accepted to be the default split to use. An 80/20 split was also undertaken but the testing accuracy suffered severely for this change.

A validation split of 15% of the training dataset was used, which is around 10% of the total training dataset. This split was found to be the most optimal value with higher values underfitting the model.

C. Performance on the training and validation set

With an epoch count of 35 and validation split of 15% the training and validation results for accuracy and loss were plotted with cross entropy being the loss function used for this model. Figure 3 shows the result of plotting training accuracy vs validation accuracy for the CNN model.

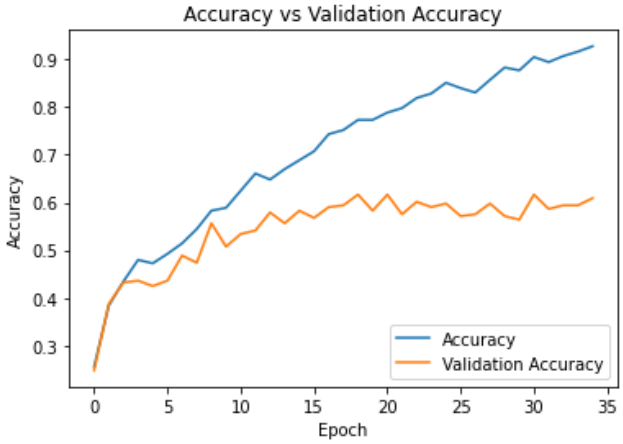

Fig. 3. Training accuracy vs validation accuracy on the original CNN model

From Figure 3 it is shown that the CNN model is severely overfitting the data. This is apparent because the training accuracy achieved over 90% accuracy whereas the validation accuracy sits at around 60% at an epoch of 10 and onwards. Visually the orange and blue lines can be seen to diverge at around epoch 2 and do not converge after, the accuracy at this point is around 45%.

More extreme signs of overfitting can be seen in Figure 4 that plots the training loss vs validation loss.

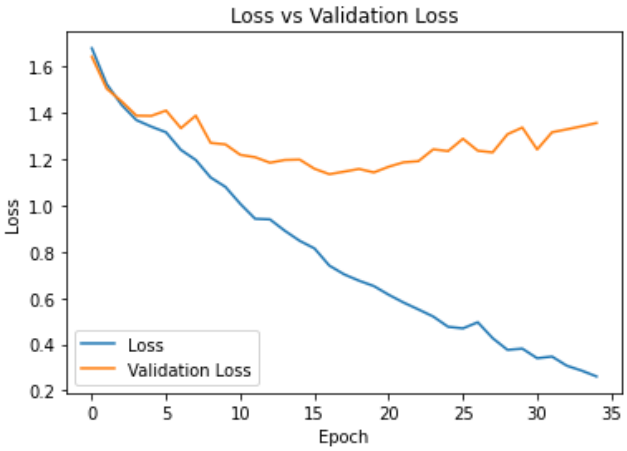

Fig. 4. Training loss vs validation loss on the original CNN model

From Figure 4 the comparison between training and validation loss can be visualised. Interestingly as the training loss tends to 0 the validation loss increases slightly per epoch. At around epoch 4 the first signs of divergence occur with a loss value of around 1.4. It is also worth noting the variability in the validation loss as can be seen by the spikey nature of the line whereas the training loss is smoother. This hints that the model is memorising the training set features rather than generalising these features for application to the validation set. This variability is also very apparent in the validation accuracy seen in Figure 4. The main crutch to this CNN model is overfitting which is addressed in the comparison study model.

D. Performance on the testing set

Although overfitting is a worrying issue for this CNN model, the model performs well above a guessing level of ~17% with a mean testing accuracy of 61% for 10 iterations as shown in Table 1.

TABLE I. MEAN TESTING ACCURACY AND LOSS FOR TEN ITERATIONS

Iteration	Testing accuracy	Testing loss
1	0.6214	1.2225
2	0.6332	1.1074
3	0.6412	1.1092
4	0.6306	1.1943
5	0.5567	1.3883
6	0.5712	1.3429
7	0.6266	1.1996
8	0.6042	1.2356
9	0.6517	1.1339
10	0.6095	1.187
Mean	0.61463	1.21207

The highest accuracy achieved on the testing set was 65% and the lowest loss was 1.1. As the training/testing split for this model was 70/30 on average the CNN model can accurately classify 502 of 758 images.

E. Performance on the prediction set

Another dataset was created to check the effectiveness of the model on images taken in different conditions. Conditions unlike those found in the original dataset are likely to be encountered if the model is used in households or recycling centres. The dataset is small with only six images, one for each output class. The hoped outcome of this developed CNN model is that these images are classified accurately however, this is not the case as shown in Table 2.

TABLE II. MEAN NUMBER OF CORRECTLY PREDICTED IMAGES FOR TEN ITERATIONS ON TH E PREDICTION DATASET

Iteration	Correctly predicted
1	2
2	2
3	1
4	2
5	1
6	2
7	3
8	1
9	1
10	1
Mean	1.6

The result is disappointing with around 1 of 6 images being classified correctly in the dataset with rounding down applied.

IV. COMPARISON STUDY

The performance of the CNN model was non-optimal with an issue of overfitting that caused the training set to be more accurate than both the validation and testing sets. The aim of this comparison study is to improve the current CNN model by addressing the issue of overfitting. To increase the accuracy of the model on the training/testing/validation and prediction datasets. This is achieved through the application of image augmentation to increase the training set size and the addition of dropout layers to manage overfitting.

A. Image Augmentation

Image augmentation is a technique used to increase the size of the training dataset by taking an image and applying certain modifications to it. The hope with image augmentation for this project is to improve the "model prediction accuracy" and "reduce the data overfitting" [9].

The TensorFlow image library allows the simple modification of images originally; flipping left and right, applying saturation, brightness, centre cropping, and rotating was applied to produce twelve images from one. This approach effectively multiplied the training dataset by 12 however, both the training and validation accuracies were extremely low with a value of around 33% at epoch 100. It was clear that the number of augmented images was too high, so the number was reduced to seven and randomness introduced.

Three possible differing values for saturation, brightness, centre cropping, and rotating was applied, with the program randomly selecting one for each to add to the dataset. Producing a total of seven when flipping and the original image was added. Image augmentation added variability to the training dataset which improved the ability for the model to generalise while also increasing the training set size which improves training accuracy.

B. Dropout

Dropout is a common technique used in neural networks to reduce overfitting by temporarily removing neurons "from the network, along with all its incoming and outgoing connections" [10]. From this paper it is stated that the optimal value for dropout seems to be around 0.5 for a wide range of networks. Using the Keras dropout layer this parameter is called rate which determines the "fraction of input units to drop" [11]. For the current CNN model, it was determined that dropout should be applied only after the pooling layers. This is because overfitting can occur when the model is too complex and "can end up over-fitting to random effects that are present only in the specific data-set used for training" [12].

C. Updated CNN model

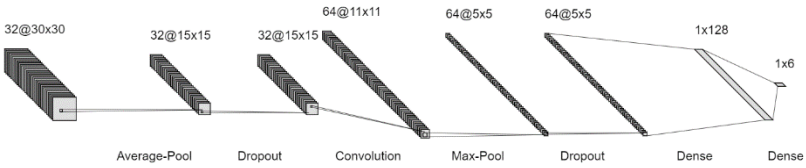

Fig. 5. Updated CNN model architecture diagram with the addition of two dropout layers, generated with http://alexlenail.me/NN-SVG/LeNet.html

The architecture for the updated CNN model is shown in Figure 5. Two new dropout layers have been added, one after the average pooling layer and another after the max pool layer. The filter-sizes, kernel-sizes, pooling sizes, and activation functions stayed constant from the original model so will not be discussed here.

The rates for the dropout layers are important, as addressed by Aysegul Takimoglu the optimal rate should be around 0.5. This was found to be the case with both dropout layers having a value slightly under at 0.45. Decreasing this rate more, caused overfitting and a general decrease in testing accuracy whereas, increasing this rate caused underfitting. Overall, 0.45 was a good value and finds the balance between over and under fitting.

The number of trainable parameters stays the same at around 260,000 as dropout layers do not require the use of trainable parameters. The calculation time per epoch does increase as compared to the original CNN model. This is due to the increase in training set size from 1769 (70% of the original dataset size) to 12383 in the modified CNN model. As a result, the time to calculate each epoch is increased from around 1 second to 9 seconds. Overall, still a quick training time as the number of epochs required for a high accuracy is low at around 35 for the original to 20 on the modified so, fine-tuning of the dropout layers could still be done quickly.

D. Performance on the training and validation set

With an epoch count of 20 and validation split of 15% the training and validation results for accuracy and loss were once again plotted with the accuracy being plotted first as shown in Figure 6.

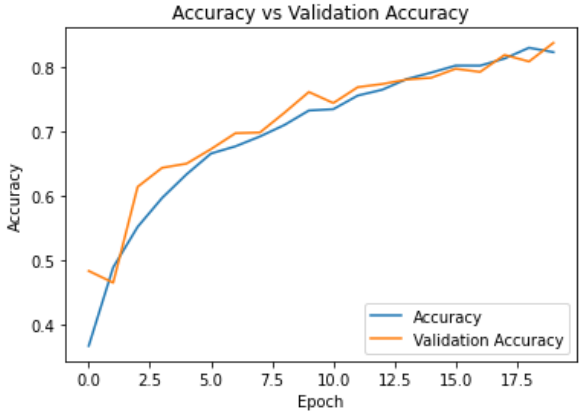

Fig. 6. Training accuracy vs validation accuracy on the updated CNN model

Unlike in the original CNN model, the number of fluctuations in the validation accuracy has decreased and overall, the validation and training accuracy is very similar with slight overlap at points. Both the validation and training accuracies reach 80% at an epoch of 16 onwards which is a vast improvement to the original model which overfit the training set at the expense of validation accuracy. Overfitting has ceased due to a combination of factors; the addition of dropout layers, image augmentation and reducing the epoch count.

Similarly, to the original model the training loss vs validation loss was also plotted as shown in Figure 7.

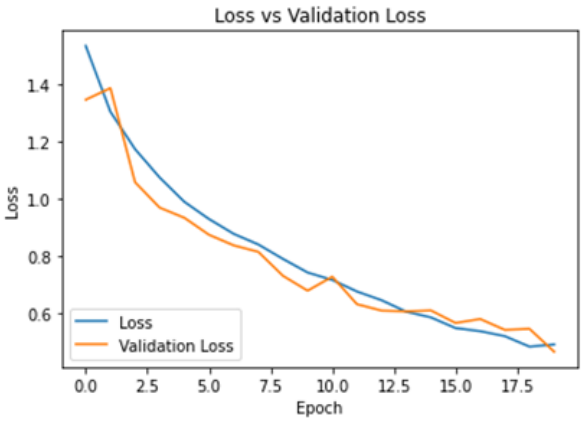

Fig. 7. Training loss vs validation loss on the updated CNN model

From Figure 7 the performance of the cross-entropy loss function can be seen. Similarly, to Figure 6 the model reduces the number of fluctuations in the validation set as compared to the original model and smooths out the training loss line which was more jagged before. Both training and validation sets have seen improvements because of the measures taken to prevent overfitting. This is once again shown by the similarities of both lines. There are three intersections occurring in Figure 7 the final intersection is of particular interest and occurs at an epoch of ~18 which has a loss value that is around 0.55. This is the lowest loss value calculated by the updated model and is vastly improved from the validation loss of 1.4 at an epoch of 35 as shown in Figure 4.

E. Performance on the testing set

The testing accuracy and loss for 10 iterations of the updated CNN model is shown in Table 3 with the mean calculated also.

TABLE III. MEAN TESTING ACCURACY AND LOSS FOR TEN ITERATIONS

Iteration	Testing accuracy	Testing loss
1	0.7533	0.7657
2	0.748	0.7882
3	0.7335	0.8051
4	0.7243	0.8075
5	0.7269	0.7997
6	0.7203	0.832
7	0.7137	0.809
8	0.7441	0.7788
9	0.7243	0.7878
10	0.7203	0.8255
Mean	0.73087	0.79993

The updated CNN model performs well with a mean accuracy of 73%. With the highest accuracy being 75% for the first iteration and 71% being the lowest at iteration 7. The testing set is still the same size between both models with the updated CNN model being able to classify 553 of 758 images, 51 more than the original model.

Table 4 shows the results of comparing the min, max and mean accuracies for both models.

TABLE IV. MIN, MAX AND MEAN METRICS FOR ACCURACIES COMPARED TO THE ORIGINAL AND UPDATED CNN MODELS

Metric	Original CNN	Modified CNN
Min	0.5567	0.7137
Max	0.6517	0.7533
Mean	0.61297	0.73087

As shown in Table 4 the accuracies do not overlap between both models. For instance, the minimum accuracy in the updated CNN model is still on average 6% higher than the maximum value produced by the original. Additionally, the difference between min and max accuracies for the updated model is slightly over four whereas, for the original model the difference is ten. This hints that the updated model is less prone to fluctuations and has learned to generalise from the training set to the testing set. Overall, the updated CNN model proves to be more suited to garbage classification due to the implementation of image augmentation and the addition of two dropout layers, providing an increase in mean accuracy of 10% compared to the mean produced by the original model.

F. Performance on the prediction set

The accuracy of the updated CNN model on the prediction dataset is shown in Table 5.

TABLE V. MEAN NUMBER OF CORRECTLY PREDICTED IMAGES FOR TEN ITERATIONS ON TH E PREDICTION DATASET

Iteration	Correctly predicted
1	3
2	2
3	3
4	3
5	4
6	3
7	4
8	4
9	3
10	4
Mean	3.3

A mean value of 3.3 was calculated compared to the original model that produced a mean value of 1.6. Although an improvement the result is still disappointing, that only half of the prediction dataset is being correctly identified when rounding down is applied. This is no doubt to do with the

differing image conditions found in the prediction dataset than compared to the Kaggle garbage classification dataset. The results of the prediction set poses future questions about improvements that can be made to both the model and dataset.

V. Discussion

A CNN model was developed with the intention of correctly classifying an image into one of six categories namely: carboard, glass, metal, paper, plastic, and trash. The original model was prone to overfitting and achieved 61% on average on the testing set. A comparison study was then conducted with the goal of creating a new improved model to prevent overfitting with the hopes of improving overall testing accuracy of the model. Two key improvements to the implementation were made, firstly the training set size was increased by a factor of seven using image augmentation. Secondly, two dropout layers were added, after the average and maximum pooling layers, with the dropout rate being fine-tuned to manage the overfitting of the training set. This fine tuning resulted in a rate of around 0.45 which was found to be an optimum value. With these changes applied the modified CNN model produced a mean testing accuracy of 73%, a 12% increase from the original. Furthermore, the modified model addressed the issue of overfitting as both the validation and training sets produced similar accuracies and loss values as shown in Figures 6 and 7.

Both CNN models were tested against the original Kaggle dataset all in images in the dataset will have similar image features for example the brightness, shadows and texture will be reasonably close. This is shown by the comparison of the testing and prediction datasets, where the prediction dataset was created myself that includes six images, one for each class.

The original model on average can only classify one of six images correctly and the modified model can classify three. The improvement was based on the changes made to the original CNN model to address overfitting. Namely the new model introduced variability into the training set by modifying the image brightness, saturation and more. With this said the improved CNN model is still lacklustre in its application to the prediction dataset due to these extreme differences in image conditions between the prediction and testing sets.

Overall, the result of the CNN model was a success with the hypothesis being partially accomplished. Where partially indicates that waste material images were classified correctly regarding the original dataset however, adaptability to differing image conditions is a work in progress. That is ultimately necessary for potential deployment of this model in households or recycling centres. In the future work section remedies to this problem are addressed and future research and ideas explored.

VI. Future work

Garbage is hard to classify because of the countless variations possible. For instance, differing colours of glass, paper and cardboard exist that still belong to that specific category but may have little trained data available for this specific type. Another consideration is the quality, and condition of garbage. For example, certain metals will discolour with time or perhaps a piece of paper has been crumpled or stained.

Another consideration to mention is the need for further classes of garbage. Six is most likely not quite broad enough and so perhaps the introduction of a new category is needed. Garbage can quite commonly coincide between two or more classes such as cardboard with a plastic film or paper cups with plastic lids. This new category could be labelled combined and allow for a more diverse classification of waste material.

The general issue being addressed is that the number of possible variations garbage can take is extreme. Which really indicates the need for a rigorous diverse dataset to help achieve this goal. A collaborative dataset is one such improvement that could solve these issues where classified images are uploaded by users worldwide with differing image conditions and types of waste. This combined dataset would hopefully increase the variability of the data and allow the model to make more accurate predictions on never-before-seen waste.

If the collaborative dataset performs well the automatic sorting of waste material could be developed. With one possible idea being the use of a smart bin in households that captures the waste material as an image, performs a prediction on this which classifies then sorts the waste into the correct section. User feedback could also be employed to improve the combined dataset with local regional data influencing the sorting process.

Acknowledgements

With thanks to the open-source TensorFlow GitHub repository [13] for providing useful insights and implementations for the Keras sequential model and matplotlib visualisations for the dataset images. The author would also like to thank Kaggle user cchangcs for the garbage classification dataset that has been used throughout this paper.

References

[1] Government Statistical Service. (2021). “UK Statistics on Waste”. Department for Environment, Food and Rural Affairs. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/1002246/UK_stats_on_waste_statistical_notice_July2021_accessible_FINAL.pdf

[2] Moore, Darrel. (2021). “UK recycling rate increases 1.2%”. Circular. https://www.circularonline.co.uk/news/uk-recycling-rate-increases-1-2/

[3] Recycle Now. (2021). “RECYCLING CENTRES”. https://www.recyclenow.com/recycling-knowledge/how-is-it-recycled/recycling-centre

[4] C. Bircanoğlu, M. Atay, F. Beşer, Ö. Genç and M. A. Kızrak. (2018). "RecycleNet: Intelligent Waste Sorting Using Deep Neural Networks".

2018 Innovations in Intelligent Systems and Applications (INISTA), 2018, pp. 1-7. https://ieeexplore.ieee.org/document/8466276

[5] CCHANG. (2018). "Garbage Classification" [Dataset]. Kaggle. https://doi.org/10.34740/KAGGLE/DS/81794

[6] Jonghong Kim, O. Sangjun, Yoonnyun Kim, Minho Lee. (2016). "Convolutional Neural Network with Biologically Inspired Retinal Structure". Procedia Computer Science, 88, pp. 145-154. https://www.sciencedirect.com/science/article/pii/S187705091631674X

[7] Nwankpa, Chigozie & Ijomah, Winifred & Gachagan, Anthony & Marshall, Stephen. (2020). "Activation Functions: Comparison of trends in Practice and Research for Deep Learning". https://arxiv.org/pdf/1811.03378.pdf

[8] Pandey, Swarnima. (2020). "How to choose the size of the convolution filter or Kernel size for CNN?". Medium. https://medium.com/analytics-vidhya/how-to-choose-the-size-of-the-convolution-filter-or-kernel-size-for-cnn-86a55a1e2d15

[9] Takimoglu, Aysegul. (2021). "What is Data Augmentation? Techniques, Benefit & Examples". AI multiple. https://research.aimultiple.com/data-augmentation/

[10] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). "Dropout: a simple way to prevent neural networks from overfitting". The journal of machine learning research, 15(1), pp. 1930

[11] Keras. (2021). "Dropout layer". https://keras.io/api/layers/regularization_layers/dropout/

[12] PICO. (2021). "Overfitting, Variance, Bias and Model Complexity in Machine Learning". https://www.pico.net/kb/overfitting-variance-bias-and-model-complexity-in-machine-learning/

[13] Daoust, Mark et al. (2021). "Convolutional Neural Network (CNN)", GitHub repository, https://github.com/tensorflow/docs/blob/master/site/en/tutorials/images/cnn.ipynb