Deep Learning Garbage Classification

Pedro Monteiro, 97484
Work Load: 50%
Machine Learning Fundamentals
Course Instructor: Pétia Georgieva
DETI, Aveiro University
Aveiro, Portugal
pmapm@ua.pt

Eduardo Fernandes 98512
Work Load: 50%
Machine Learning Fundamentals
Course Instructor: Pétia Georgieva
DETI, Aveiro University
Aveiro, Portugal
eduardofernandes@ua.pt

***Abstract*—The recognition of objects will be the main emphasis of this article, with a focus on the recognition of different types of garbage such as metal, glass, paper and plastic using deep learning techniques. All data utilized comes from a kaggle dataset [1]. Deep learning algorithms were implemented and compared to each other, verifying the best for this problem. Later on, a full comparison will be made of the classifiers implemented in the first project and this project, as well as those implemented by the authors of other articles/works.**

***Keywords*—machine learning, deep learning dataset, garbage, logistic regression, svm, decision tree, neural networks, random forest, algorithm**

I. Introduction

This article comes within the scope of the second FAA [2] project, where the objective consists in the application of Deep Learning techniques, either developed during class or self-taught, in the solving of one of several problems.

Deep learning is a powerful tool in the field of machine learning, and it has been applied to a wide range of applications such as image and speech recognition, natural language processing and computer vision.

In recent years, it has also been used to tackle the problem of garbage classification.

In this article, we will explore how deep learning techniques, specifically convolutional neural networks (CNNs), can be used for garbage classification, and how it can improve the efficiency of the process.

All the code developed can be found in the notebook that is in our github repository [3].

II. State of the Art

Along with the dataset, available on Kaggle [1], it is possible to find works developed by several authors, implementing machine learning and deep learning techniques that aim to develop a good model that classifies the types of garbage in a precise way.

Doing a little research on the notebooks and exploring the most popular ones, the notebook of Jabullae [4] stands out. Keras library was used to configure the learning process of the neural network, being used as cost function 'categorical_crossentropy', which is a common loss function for multi-class classification problems, the optimization function 'adam', which is a popular optimization algorithm that adaptively adjusts the learning rate during training and accuracy was used as a metric.

In the end, it's obtained a final accuracy of around 90% for the training data and between 40% and 50% for the test data, which is a relatively low value.

Another relevant notebook was Marcelo Moreno's [5].

Similar to the previous work, a CNN was also implemented, also using the same optimizer, adam, and the same metric, accuracy. In this case, was used another cost function, sparse_categorical_crossentropy. This cost function is common for multi-class classification problems with integer labels. It calculates the cross-entropy loss between the predicted output and true output.

The final accuracy was higher than the previous work, reaching around 93% in the training case and close to 90% for the test data.

Outside of Kaggle, several articles were analyzed and the most relevant ones explored.

One of the articles explored was found in the IEEE repository, with the authors Shanshan Meng and Wei-Ta Chu [6].

In this article a study about garbage classification using Convolutional Neural Networks is done. The initial dataset contained 2527 images, and the authors, through flipping and rotation, increased it to 10108 images. The authors created a model of ResNet50, where they first used conv. layer and a pooling layer to get the rough features of images. After the normal conv. block, the model uses totally 16 residual blocks with an increasing dimension of features. The residual blocks also use ReLU as activation function to make the most of its advantages. The same as the simple CNN architecture, ResNet50 also uses softmax as the activation function in the last layer.

In terms of final accuracy, was obtained a value very close to 90%, both in training data and in test data, with the train cost and validation cost also decreasing over the number of epochs.

In addition to this article published in IEEE, we also explore the work done by Zhongxue Yang, Yiqin Bao, Yuan Liu, Qiang Zhao, Hao Zheng and YuLu Bao titled Research on deep learning garbage classification system based on fusion of image classification and object detection classification [7].

In a similar way, the ResNet model was implemented, and in this case the final accuracy is 99%, in training data, which represents an extremely good performance, even though 41 hours of training were used, and 81.5% in test data.

III. Dataset Preparation

A. Dataset Description

The Garbage Classification dataset [1], available on Kaggle, was the dataset chosen to perform the task of identifying different garbage objects, such as a can of coke, a cardboard box or a bottle of water.

This dataset is the same used in project 1, and will now be used to apply deep learning techniques.

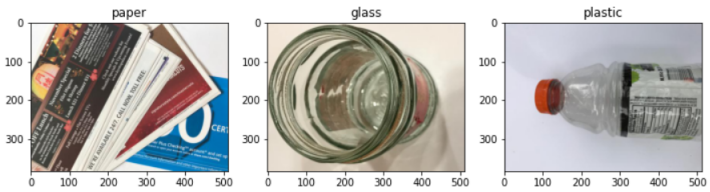

Fig. 1. Dataset Images Example

The dataset is organized into folders, each containing images with a single object relating to a type of garbage, indicating which type each image represents.

The dataset contains a total of 6 different classes, which are:

- Paper
- Cardboard
- Glass
- Plastic
- Trash
- Metal

For the application of the models we selected only three, those that we considered most relevant, paper, plastic and glass, as can be seen in figure 1:

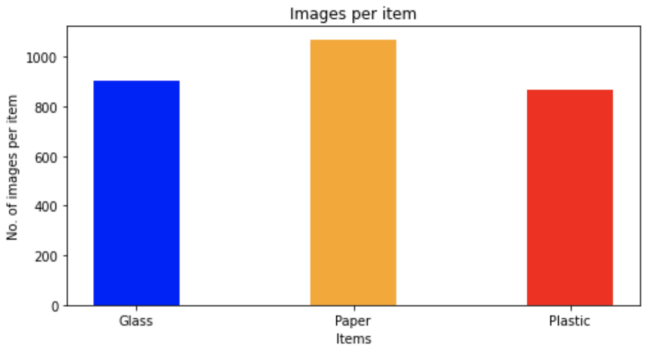

Fig. 2. Unbalanced Dataset Images per Item

With this it's easy to verify that the dataset is unbalanced, that is, there is a very high different between the number of items of each class. In other words, classes are not represented equally. There are about 1100 images for paper whereas for plastic there are only 868.

B. Balancing

The problem described above, dataset imbalance, can lead to overfitting and outliers in the data, so there is a need to correct it. For this, the data need to be normalized, generating similar proportions for each of the classes.

In order to eliminate possible outliers in the problem, all classes now have the same number of examples, 868, as can be seen in figure 2. Thus, the dataset is balanced, contributing to the models being more accurate.

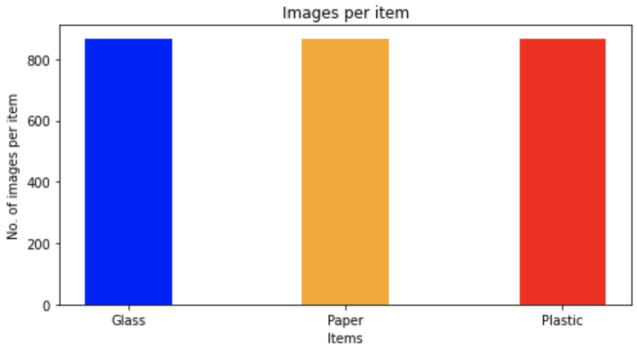

Fig. 3. Balanced Dataset Images per Item

C. Underfitting

Also, we found that the accuracy had relatively low values, around 50%, due to the fact that the training examples were few, so was necessary to correct this problem, adding more items to train.

Initially, little data was being used for training, about 20% of the dataset. As the dataset wasn't very big, 20% was a bit too short for what was needed, so we added more data, which led to a significant increase in accuracy.

D. Scaling

Another problem we faced was that the images had very large sizes, 512x384, which made the algorithms significantly expensive and time consuming. There was then a need to change the size of the images, each one having 4096 pixels, 64x64.

E. One-hot encoding

It was also necessary convert the target label vector into a binary class matrix, using a utility function from Keras called 'to_categorical', also known as one-hot encoding.

One-hot encoding is a way of representing categorical variables as binary vectors. Each element in the input vector is encoded as a binary vector of the same length, with a 1 in the position corresponding to the index of the element and 0s in all other positions.

In the case of this code, the y variable is being converted into a one-hot encoded format, which is useful for training neural networks with categorical variables, as it ensures that

the model will not assume any ordinal relationship between the classes. It also allows the model to output a probability distribution over the classes, as the output of the final layer with softmax activation will be a probability distribution, as will be discussed below.

After applying all this pre-processing, the dataset was divided into test data and training data, with about 80% of the data used to train the model and 20% for testing.

Figure 4 represents the data for each of the classes (Glass, Paper, Plastic), and as the dataset is balanced, the number of train and test data is equal in each class.

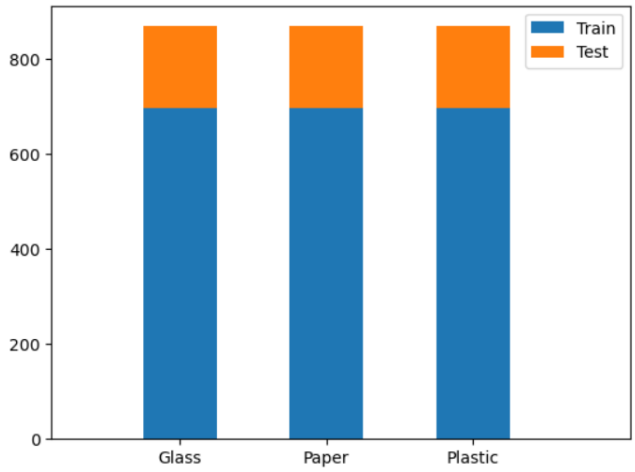

Fig. 4. Train and Test Data

IV. Models

We tried to implement the models that best fit and that prove to be more efficient in the facing the image classification problem, according to the work demonstrated in the papers and articles found.

So after analyzing and exploring the various articles already mentioned we decided to implement a Convolutional Neural Network (CNN) and make another implementation that follows the ResNet50 architecture, as developed by most of the authors of the articles.

A. CNN

The first model we implemented was a convolutional neural network (CNN) [8] using the Sequential model from the Keras library [9].

A CNN is a type of neural network that is particularly effective at processing image data.

The model has a series of convolutional layers (Conv2D), activation layers (Activation), pooling layers (MaxPooling2D), and dropout layers (Dropout) followed by a fully connected dense layer (Dense) and a final output layer (also Dense) with 3 nodes and a 'softmax' activation.

The convolutional layers (Conv2D) are responsible for extracting features from the input image. Each convolutional layer applies a set of filters to the input, where each filter is a small matrix that is convolved with the input to produce a feature map.

Activation layers (Activation) are used to introduce non-linearity to the model. The activation function used is ReLU (Rectified Linear Unit), which is a common choice for CNNs. The ReLU function replaces all negative input values with zero, allowing the model to learn more complex decision boundaries.

Pooling layers (MaxPooling2D) are used to reduce the spatial dimension of the output from the previous layer by a factor of 2.

Dropout layers (Dropout) are used to randomly drop out a portion of the input units during training, which can help to prevent overfitting.

The final layers of the model are a dense layer (Dense) with 150 nodes and a final output layer with 3 nodes and a 'softmax' activation.

The final output layer uses the softmax activation function, which produces output values that represent probability distributions over the possible classes.

The model is then compiled with categorical_crossentropy as the loss function, optimizer is set to 'rmsprop' and accuracy is used as the evaluation metric. Loss function is used to measure how well the model is doing in terms of training and optimizer is used to adjust the parameters of the model in order to minimize the loss function. Evaluation metric is used to evaluate the performance of the model.

In terms of the results obtained, figure 3 shows that the accuracy is significantly high, both in the training data and in the test data, with values around 95%/96%.

The accuracy obtained represents a very high value, which may mean that this model is indicated for this type of problem.

Several tests were carried out to determine the number of epochs to be used, 50 already representing a very high value and leading to overfit, the results in the training data being very good but in the test data the same was not verified, and the training time started to be high.

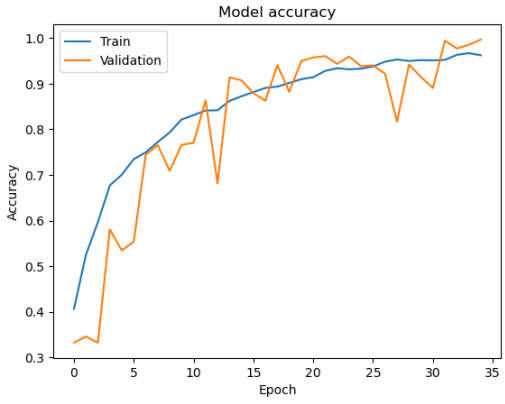

Fig. 5. CNN Accuracy Variation with Epoch

Also the values in the case of loss are very good, showing an increasingly smaller loss with the number of epochs, approaching the value 0 at the end.

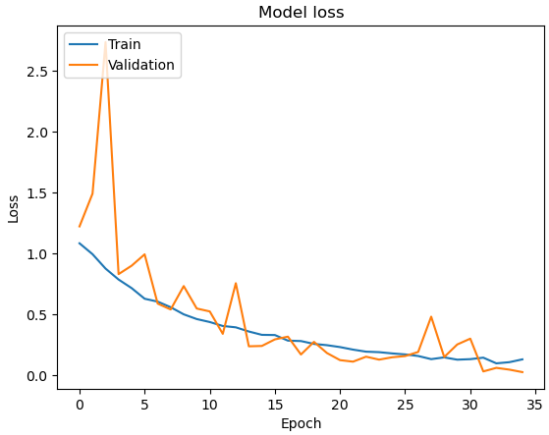

Fig. 6. CNN Loss Variation with Epoch

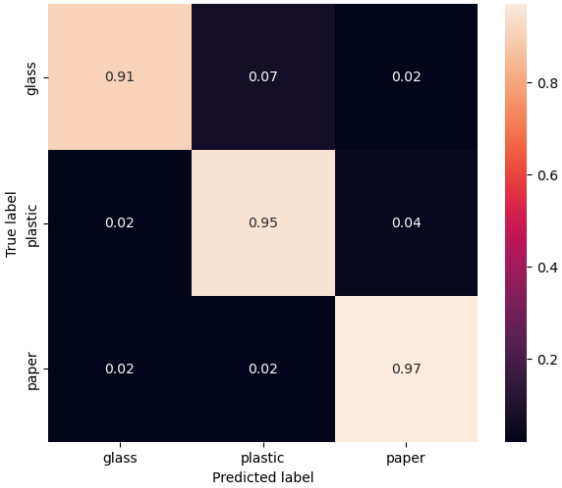

Fig. 7. CNN Confusion Matrix

Here we have the confusion matrix, which is extremely useful for measuring *recall*, *precision*, *specificity* and *accuracy*.

It is a performance measurement for machine learning classification problem where output, in our case, is 3 classes.

As it is possible to see, the classifier achieved an accuracy above 0.90 for each class, which is a relatively good result.

B. ResNet50

ResNet50 [10] is a convolutional neural network (CNN) architecture that was developed by Microsoft Research in 2015. It is a deep residual network that is trained on the ImageNet dataset, which contains more than 14 million images and 1000 classes. The architecture is designed to improve the performance of deep CNNs by addressing the problem of vanishing gradients.

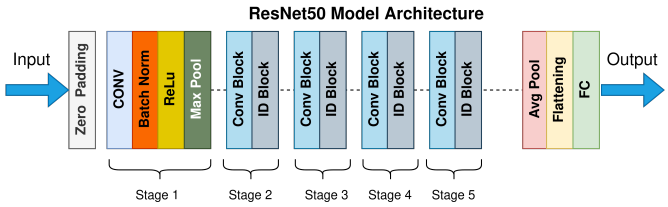

Fig. 8. Example of ResNet50 Architecture

ResNet50 has 50 layers and uses the ResNet block structure. The ResNet block structure is composed of two convolutional layers, batch normalization and a shortcut connection. The shortcut connection is used to add the input to the output of the block, this allows the flow of information through the block.

In the same way as analyzed in the previously mentioned articles, we also implemented convolutional neural network (CNN) using the ResNet50 architecture, which, as already mentioned, is a pre-trained model available in the TensorFlow Keras library [9].

The pre-trained model is then used as the input for a new model. The new model starts by adding a global spatial average pooling layer, GlobalAveragePooling2D(), to the output of the base model. The global spatial average pooling layer is used to reduce the spatial dimensions of the feature maps while maintaining the channel-wise information.

Then, the new model adds a fully connected layer Dense(1024, activation='relu') with 1024 neurons and a ReLU activation function.

Finally, the new model adds a logistic layer Dense(3, activation='softmax') with 3 neurons and a softmax activation function. The softmax function is used to output a probability distribution over the different classes.

The model is compiled, using 'adamax' as optimizer, 'categorical_crossentropy' as loss function, and the metrics used is 'accuracy'.

Finally, the model is trained on the training data for 40 epochs, with a batch size of 32.

To determine the performance of this implementation, the accuracy was plotted over the number of epochs, with normal accuracy increasing as the number of epochs increases, due to model learning.

The result of the loss function was also plotted, which, as expected, should decrease over time, that is, as the number of epochs increases.

Figure 9 represents the accuracy variation and, as you can see, it starts at around 45%, with 1 epoch, and ends at around 65%, with 40 epochs.

The expected final accuracy was higher, since this is a model quite suitable for this type of problems that also uses a cost function indicated for multiclass problems.

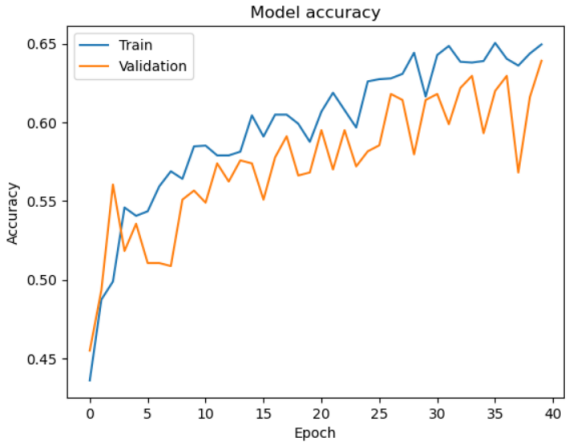

Fig. 9. ResNet50 Accuracy Variation with Epoch

As already mentioned, the loss was also plotted throughout the process, being naturally higher at the beginning, 1 epoch, and with learning over time, the loss value at the end of the 40 epochs is relatively smaller.

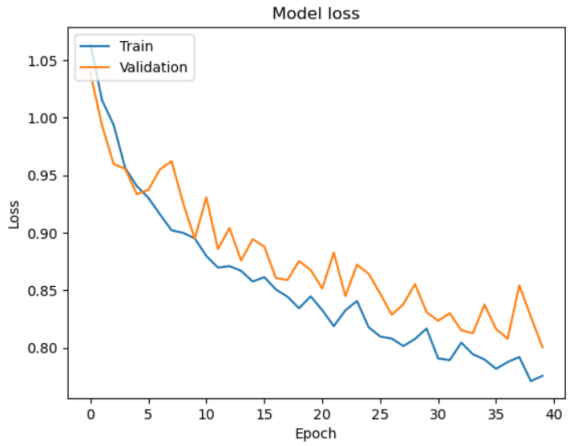

Fig. 10. ResNet50 Loss Variation with Epoch

V. Models Comparison

Regarding all models, implemented both in project 1 and in project 2, the performances of each one were represented in table I, with the objective of comparing the classifiers and to understand which one best suits to this type of classification.

As expected, CNN has a significantly high accuracy, 94%, although it is also matched by the Random Forest Classifier, developed in the previous project, 95%.

All other classifiers developed in the previous project, Logistic Regression, Decision Tree, Neural Networks and SVM have similar accuracies, between 70% and 80%, and not being very good they are not very bad either, and can be said that these are reasonable classifiers for this type of problem.

Finally, and contrary to expectations, the CNN that follows the ResNet architecture has the lowest accuracy, around 65%, which is far below what was expected, being this model considered very good in this type of problems.

TABLE I
Test Accuracy Results

Model	Base	Hypertuned	K-fold CV
Logistic Regression	84.64%	85.60%	84.64%
SVM	85.22%	90.02%	85.22%
Random Forest	95.01%	94.05%	95.20%
Decision Tree	89.64%	86.96%	89.25%
Neural Networks	91.75%	90.55%	90.59%
CNN	94%		
ResNet50	64%		

The box plot below allows an easier visualization of the accuracies of each model, being perfectly visible the one that presents better results.

CNN and Random Forest stand out at the top of the box plot, in the 95% accuracy line, with the rest of the classifiers in relatively lower positions.

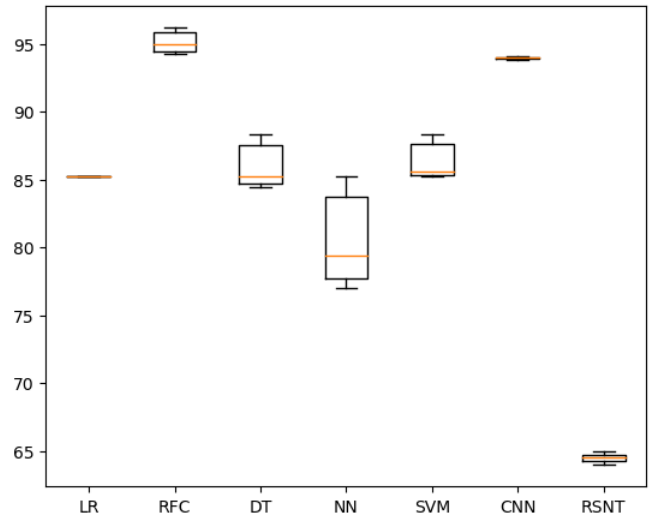

Fig. 11. Model Comparison with Box Plot

VI. Novelty and Contributions

Looking at the results obtained by the authors and comparing them with the results obtained by the ResNet implemented by us, we observe that the results in the case of the authors are relatively better, with higher accuracy.

Looking at Table II, it is noticeable that the ResNet of Zhongxue Yan and his colleagues is quite efficient, classifying garbage images with 90% accuracy. In the case of Shanshan Meng & Wei-Ta, although the accuracy in the test data is relatively lower, 81.5% can be considered a good model for classifying images.

In the case of the approximation developed in this article, the values are a little low, 65%, when compared to the work of other authors, and this model may have some difficulty in classifying certain images.

TABLE II
ACCURACY RESULTS COMPARISON

Model	Zhongxue Yan	Shan. & Wei-Ta	Our Approach
ResNet50 (train)	99%	90%	65%
ResNet50 (test)	81.5%	90%	64%

VII. CONCLUSION

Deep learning is a powerful tool that has been used to solve various problems in the field of machine learning. One of the areas where deep learning has shown great potential is in garbage classification. Garbage classification is an essential process in waste management, and it is crucial for reducing the amount of waste sent to landfills and increasing the efficiency of recycling.

Convolutional neural networks (CNNs) have been shown to be particularly effective for image-based garbage classification tasks. They have been used to classify images of waste into different categories based on their visual characteristics. The ResNet-50 architecture is one of the most popular CNN architectures used for image classification, and it has been used to achieve state-of-the-art performance on several image classification benchmarks.

In this article, we have explored how deep learning techniques, specifically CNNs, can be used for garbage classification, and how it can improve the efficiency of the process. By automating the process of garbage classification, deep learning can help to reduce human error and increase the accuracy of the classification. With the increasing amount of waste generated every day, the use of deep learning in garbage classification can play a crucial role in improving waste management and reducing the environmental impact of waste.

WORKLOAD

Each student worked 50% of the project.

ACKNOWLEDGMENT

We would like to thank Professor Pétia Georgieva, regent of FAA course, for being available to answer questions and for the flexibility to easily schedule meetings to discuss the project.

REFERENCES

[1] Dataset used https://www.kaggle.com/datasets/asdasdasasdas/garbage-classification/code
[2] Machine Learning Fundamentals https://www.ua.pt/pt/uc/15262
[3] Github Repository https://github.com/pedromonteiro01/garbage_classification
[4] Jabullae 's notebook https://www.kaggle.com/code/jabullae/garbage-classification-cnn
[5] Marcelo Moreno's notebook https://www.kaggle.com/code/martxelo/cnn-classifier-with-keras
[6] Shanshan Meng and Wei-Ta Chu' article https://ieeexplore.ieee.org/document/9181311/authors#authors
[7] Research on deep learning garbage classification system based on fusion of image classification and object detection classification https://www.aimspress.com/article/doi/10.3934/mbe.2023219
[8] CNN https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
[9] Tensorflow Keras Library https://www.tensorflow.org/api_docs/python/tf/keras
[10] ResNet50 https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758