Deep Learning Project

Free Choice - Garbage Classification

MASTER DEGREE PROGRAM IN DATA SCIENCE AND ADVANCED ANALYTICS

Group 02

Diogo Barros, 20230555

Diogo Roiçado, 20230557

Matilde Parreira, r20201587

Rodrigo Rocha, 20230593

Sofia Tátá, 20230546

April, 2024

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Contents

i

This report outlines the group's effort in the process developing a Deep Learning model to classify recyclable materials from images of garbage available on the Kaggle dataset present in both the bibliography of this report and on the main folder submitted.

1. Task Definition

Efficient and accurate garbage sorting is essential for enhancing recycling efforts and environmental sustainability. This project addresses this challenge by leveraging deep learning techniques to classify household garbage effectively. A novel dataset comprising 15515 images across 12 distinct classes of household waste is utilised, covering materials such as paper, cardboard, metals, plastics, glass, clothes, shoes, batteries, and general trash. Recognizing the limitations of existing datasets, our work aims to refine waste sorting granularity, thereby improving recyclability.

This system aims to tackle the critical environmental problem of waste management and recycling. Efficient sorting of garbage is a fundamental step in the recycling process, enhancing the recyclability of materials and significantly reducing the volume of waste that ends up in landfills. However, manual sorting of garbage is labour-intensive, time-consuming, and potentially hazardous to workers' health. Furthermore, incorrect sorting can lead to contamination of recyclable materials, rendering them unusable and defeating the purpose of recycling.

2. Evaluation Measure

In addressing our multiclass problem encompassing 12 labels, our initial metric of choice was accuracy, and minimising loss. However, upon deeper analysis of our severely unbalanced data, we recognized the importance of prioritising the F1 score for a more nuanced evaluation. To achieve this, it was necessary to create a class F1 score, given that *Keras* lacks such a metric and the one offered by scikit-learn has compatible issues with *Keras*/*Tensorflow* workflow. Additionally, to enhance the performance assessment of our model, we have implemented class weights on some models to ensure a more accurate reflection of its capabilities. Furthermore, alongside traditional metrics, we have placed significant emphasis on the efficiency and speed of our model. Given the scale and diversity of trash categories we aim to address, efficiency is paramount. Our model's ability to swiftly and accurately sort through large quantities of trash is integral to achieving our project objectives.

3. Approach

3.1 Capturing the Phenomena

Accurately differentiating visually identical items is essential to the deep learning model's efficacy in the waste image recognition domain. Images that the dataset includes like plastics, water bottles, and white glasses, for example, might be difficult to work with because of their similarities in terms of hue and transparency. It is necessary to comprehend and recognize the distinct visual phenomena—such as texture, shape, and size—to properly classify these objects.

To address these challenges, Convolutional Neural Networks (CNNs) are utilised, leveraging their ability to detect and differentiate subtle differences in spatial features. Preprocessing techniques such as image segmentation and edge detection are particularly valuable, helping to accentuate distinguishing features that might otherwise be overlooked. Furthermore, employing data augmentation techniques like rotations, scaling, and brightness adjustments ensures that the model can generalise well across different lighting conditions and object orientations, enhancing its ability to differentiate between items.

3.2 Data Collection and Curation

Our dataset was meticulously compiled from Kaggle, offering a diverse array of images illustrating various types of waste. We sourced some images from *Kaggle* repositories such as

Kaggle_clothes and Kaggle_9classes, while others were obtained through web scraping methods. It is important to note that it was considered the terms of service of the websites where web scraping images were taken, ensuring, in this way, that every legal aspect was respected.

3.3 Data Preprocessing

As a first approach to the exploration and to understand what should and could be the next steps, we looked at how the data is spread out *(Figure 1 - Number of images in Each Class)*. We noticed that some categories have a lot more pictures than others. For instance, clothes have a huge 5325 images, while green-glass only has 629. This tells us our data isn't exactly balanced so we have created a class weights variable to address the weight of each class.

We then delved into the statistics regarding the size of images across our dataset *(Figure 2 - Width and Height Analysis)* and examined their distribution *(Figure 3 - Distribution of image Width and Height)*. Initially, our plan was to resize all images to 244x244, a common size among peers that seemed suitable for our dataset. However, we soon discovered that our computers couldn't feasibly handle images of that size as it would be too time-consuming. Accordingly, we adjusted our strategy, settling on resizing them to 50x50 and 128x128 to assess the models' performance at these different sizes.

Regarding the use of RGB channels, while some image recognition problems utilise only one channel, we encounter situations, such as identifying types of glass, where colour plays a crucial role. Despite the simplicity of reducing the number of channels, we opt to utilise all three to ensure the development of the most effective model possible.

We encountered some compatibility problems, which are going to be mentioned with more detail further, when performing cross validation, so, following that, we performed a *train_test_split*, allocating 80% of each class to the training set, resulting in a total of 12,412 images. Subsequently, we further divided the remaining test data into validation and test sets using a 60-40 split. This resulted in 1,862 images for validation and 1,241 images for testing, where each class represented the same percentage in all the sets according to the percentage they represented in the whole dataset *(Figure 4 - Testing Set Class Distribution (the percentage is the same for each class in different sets))*.

Considering all the challenges mentioned, we've chosen to implement data augmentation techniques to enhance our training dataset. Data augmentation involves applying a range of transformations to existing images, thereby boosting the diversity and variability of the training data. These transformations, such as random rotations in the range (0-45 degrees), randomly shift images horizontally and vertically (10% of width and height, respectively), 0.1 shear transformations, randomly zoom images in and out (20%), horizontal flips, randomly changes in brightness (80%-120%) introduce a wider spectrum of image variations in the train data that the model may encounter during inference. At the same time, normalisation is integrated into our data augmentation process, both in train and in validation data. This involves rescaling pixel values to the range of [0, 1], reducing computational complexity, and ensuring stability during training. Despite this rescaling, the valuable insights embedded in each image are preserved, contributing to the model's ability to learn effectively from the augmented data. As illustrated in *(Figure 5 - Image Transformations)* these transformations significantly enrich the dataset, aiding the model's learning process by looking at a different augmented image at each epoch.

3.4 Modelling and Fine-Tuning

Our approach to model creation involved dividing our work into several notebooks, each dedicated to specific types of data and models. We had one notebook for pre-trained models, and two for custom models—one for 50x50 pixel size images and another for 128x128. We then consolidated the top-performing models from each category into a notebook named 'Best Models'. Finally, we conducted a manual grid search in a separate notebook to further optimize the best-performing models.

3.4.1 Pre-Trained Models

We started experimenting with different pre-trained models, such as *ResNet-18*, *ResNet-34*, *VGGNet-16*, *VGGNet-19*, *DenseNet-121*, and *DenseNet-169*. We changed batch size, added pre-trained weights, and used data augmentation techniques. Following a thorough analysis, we concentrated on *DenseNet-169* and *VGGNet-19* for 128x128 images and *VGGNet-16* and *VGGNet-19* for 50x50 image sizes because of their better results in our first testing. Using deep convolutional layers, *VGGNets* are excellent at catching minute details, which is essential for differentiating between different kinds of garbage. In the meantime, *DenseNet-169*'s effective architecture improves feature propagation and reuse, which makes it perfect for higher resolution feature recognition. These models were chosen because they strike a crucial balance between computational efficiency and reliability.

3.4.2 CNN Models and Architectures

Regarding the custom models, we utilized the *Keras* library to construct neural networks, all of which followed a Convolutional Neural Network (*CNN*) architecture. Each model consisted of one or more convolutional layers, followed by max-pooling layers, and then a flattening layer. This was succeeded by a series of dense layers, with the final layer always comprising 12 neurons and utilising a *softmax* activation function to facilitate predictions across classes. We initially began with simpler models and iteratively refined them.

During our model testing, we noticed that using the class weights wasn’t getting better results, this can be explained by the utilization of data augmentation, which is already covering the unbalancing problem. Additionally, we conducted experiments with different batch sizes, finding that batch sizes of 32 and 64 yielded optimal results, while batch size 16 showed poorer performance and batch size 128 led to overfitting.

Batch normalisation and dropout techniques were utilized in the building of the more complex models. Batch normalization was utilized to stabilize and accelerate the training process by normalizing the input of each layer and reducing the sensitivity to parameter initialization. Another technique to reduce overfitting and encourage the network to learn more robust properties was dropout, which involves randomly deactivating a predetermined number of neurons during training. In addition, callbacks were also added to every model to track and enhance the learning process. We put in place a callback system that, if the model's performance did not improve after a predetermined number of epochs, dynamically modified the learning rate. Because of this adaptive strategy, the model was able to learn efficiently. Here, *(Figure 6 - Models and Characteristics)* are the models we tested to conclude which one should we conduct a grid search in.

As we thought, data augmentation was a crucial step to prevent overfitting because, even though our best F1 score (0,97) on train was achieved with a no data augmentation model, it was also the one with the biggest overfit (0,79 val F1 score). As we can see the best model was the one with the most complex architecture for the 50x50 size – Non-Linear 3 with F1 score train – 0.905 and F1 score val – 0.85. The selection of the best overall model was based on both the highest F1 score and efficiency. Since the two best results originated from the same model architecture but were applied to different image sizes, we opted for the version applied to 50x50 dimensions due to its runtime that is 12 times faster, while yielding similar results or even better results.

3.5 Results and Final Model

Lastly, regarding the manual grid search, we applied it to the best model to improve it even more, the objective was to maintain the architecture of the model while identifying the optimal combination of parameters to maximize model effectiveness, such as optimizer (*adam* or *sgd*), dropout (0.1, 0.2 or 0.3) and the use or not of class weights.

The results of our grid search indicate that the optimal configuration comprises the Adam optimizer, a dropout rate of 0.1, and the absence of class weights. With this setup, we achieved an F1-score of 0.89 for training and 0.84 for validation with a descending loss. Furthermore, the test F1-score stood at 0.85 *(Figure 7 - Evaluation Metrics for Test Set)*. Notably, this model demonstrates no signs of overfitting, as evidenced by the loss curves and F1 score curves below, exhibiting the expected shapes *(Figure 8 - Loss and F1 curves for Final Model after Manual).*

We used a confusion matrix *(Figure 9 - Confusion Matrix)* to look more closely at each class's unique classification to understand our model predictions better. Notably, our model demonstrated strong performance in some categories with high accuracy rates for clothing (98%) and green-glass (96%). On the other hand, it performed worse with biological (62%), batteries (68%), and plastic (68%).

Our analysis revealed certain patterns of misclassification. For instance, the model frequently misclassified biological items as brown-glass. Additionally, batteries were often misidentified as clothes, metal, or shoes. Similarly, plastic items were commonly misclassified as paper or white-glass. It's worth noting that shoes were occasionally classified as clothes, which, considering their similarity, may not be a significant error. Furthermore, it's expected that clothes, as the most dominant class, encompasses most of the poorly defined labels.

Additionally, we have examples of incorrectly classified images, which are depicted in *(Figure 10 - Misclassified Images)*, which can give us some clarifications about what problems should we try to address next time that maybe the model is not solving exactly as it should.

4. Critical Evaluation and Further Work

Throughout the project, we faced several challenges, with the most significant being the limitations in computer capability and storage. These constraints forced us to adopt simpler model architectures, impacting the potential quality of our results, as well as looking for reliable options to run our code. Moreover, we sought reliable alternatives to execute our code efficiently. Initially, we opted for *Google Colab*, as recommended in our classes. However, we switched to *Visual Studio Code* for our work because of the frequent GPU disconnections. This further limited us because it meant we could only rely on the processing power of our local machines. Without these constraints, we believe our outcomes could have been more favourable.

In addition to reducing model complexity, we were unable to implement cross-validation due to resource constraints, further limiting our ability to validate and optimize our models effectively. Furthermore, we investigated the possibility of fine-tuning data augmentation methods to improve our best-performing model, especially for classes with lower performance. However, our lack of resources and incompatibility between *ImageDataGenerator* and *GridSearchCV* made it impractical to execute such improvements**.**

Further we will seek to address the problems mentioned as at the same time investigate the relation between the efficiency of class weights and data augmentation implementation, address the cross-validation problem that we couldn't overcome, try different splits for the models to learn more about the minority classes and try different images sizes with a more computational efficiency available.

5. Conclusion

To sum up, this research addresses the crucial issue of accurate and efficient garbage classification with the goal of improving recycling efforts and environmental sustainability. Our goal was to increase recyclability and waste sorting granularity by applying deep learning techniques and a novel dataset with 15515 images from 12 different classes of waste. Our evaluation prioritised the F1 Score for accuracy while focusing on efficiency as well as speed to address the significantly unbalanced data. While not all included in the final model, strategies like class weights were taken into consideration. Convolutional Neural Networks (CNNs) caught minor visual differences among apparently identical things (like the differently coloured glasses), enhancing model resilience along with a meticulous preprocessing approach.

Regarding the limitation of our computational resources, our strategy achieved promising findings. With an F1 score of 0.84 for validation and 0.85 for testing, the final model—which was chosen by manual grid search—showed strong performance without significant overfitting. Given the model's optimal balance between efficiency, precision, and accuracy, we advise its implementation. It provides a good answer for trash categorization problems, with a runtime significantly faster when compared to competing models tested by the group.

6. Bibliography

"Garbage Classification (12 Classes)." *Www.kaggle.com*, www.kaggle.com/datasets/mostafaabla/garbage-classification

7 Best Image Classification Models You Should Know in 2023 - Jonas Cleveland. (2023, July 14). Retrieved from https://jonascleveland.com/best-image-classification-models/

Corominas, Ò., Riera, I., Aditya, S., & Singh, S. (n.d.). *Image Classification with Classic and Deep Learning Techniques*. Retrieved from https://arxiv.org/pdf/2105.04895

Dang, K. (2023, February 2). Deep learning: Computer vision using transfer learning (ResNet-18) in Pytorch — Skin cancer…. Retrieved from Medium website: https://medium.com/@kirudang/deep-learning-computer-vision-using-transfer-learning-resnet-18-in-pytorch-skin-cancer-8d5b158893c5

Data Augmentation: A Class Imbalance Mitigative Measure. (2022, November 3). Retrieved April 28, 2024, from Paperspace Blog website: https://blog.paperspace.com/data-augmentation-a-class-imbalance-mitigative-measure/

How to Deal with Imbalanced Datasets in Computer Vision — Picsellia. (n.d.). Retrieved April 28, 2024, from www.picsellia.com website: https://www.picsellia.com/post/improve-imbalanced-datasets-in-computer-vision

Keras. (n.d.-a). Keras documentation: DenseNet. Retrieved from keras.io website: https://keras.io/api/applications/densenet/

Keras. (n.d.-b). Keras documentation: Learning to Resize in Computer Vision. Retrieved from keras.io website: https://keras.io/examples/vision/learnable_resizer/

Keras. (n.d.-c). Keras documentation: VGG16 and VGG19. Retrieved from keras.io website: https://keras.io/api/applications/vgg/

Sharma, D. (2021, January 11). CNN For Image Classification | Image Classification Using CNN. Retrieved from Analytics Vidhya website: https://www.analyticsvidhya.com/blog/2021/01/image-classification-using-convolutional-neural-networks-a-step-by-step-guide/

VGG-16 | CNN model. (2020, February 26). Retrieved from GeeksforGeeks website: https://www.geeksforgeeks.org/vgg-16-cnn-model/

7. Annexes

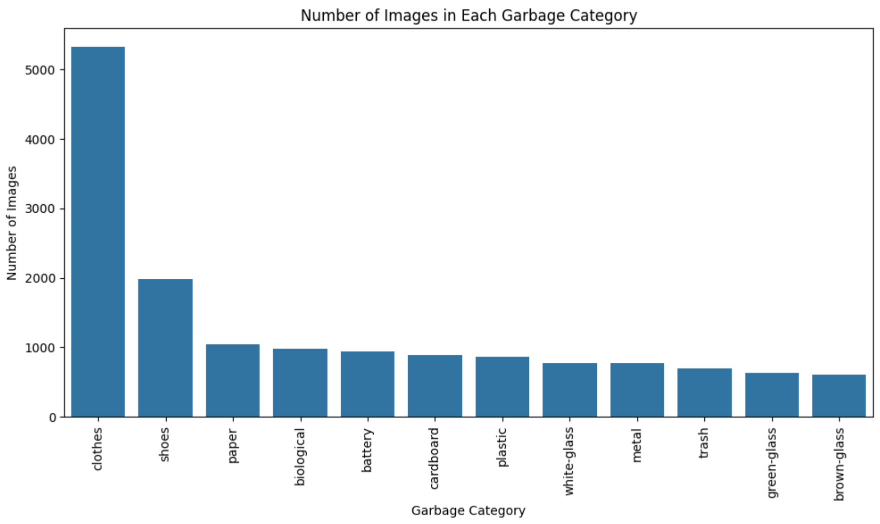
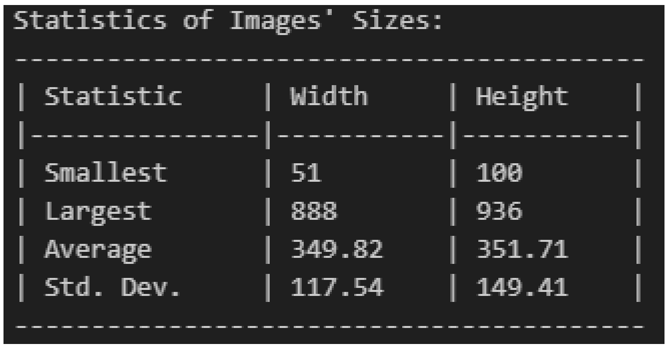


Figure 1 - Number of images in Each Class



```
Statistics of Images' Sizes:
-----------------------------------------
| Statistic      | Width      | Height     |
|---------------|-----------|-----------|
| Smallest       | 51         | 100        |
| Largest        | 888        | 936        |
| Average        | 349.82     | 351.71     |
| Std. Dev.      | 117.54     | 149.41     |
-----------------------------------------
```

Figure 2 - Width and Height Analysis

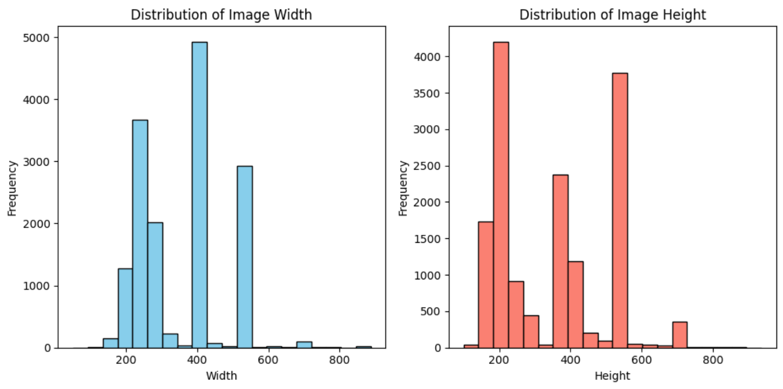


Figure 3 - Distribution of image Width and Height

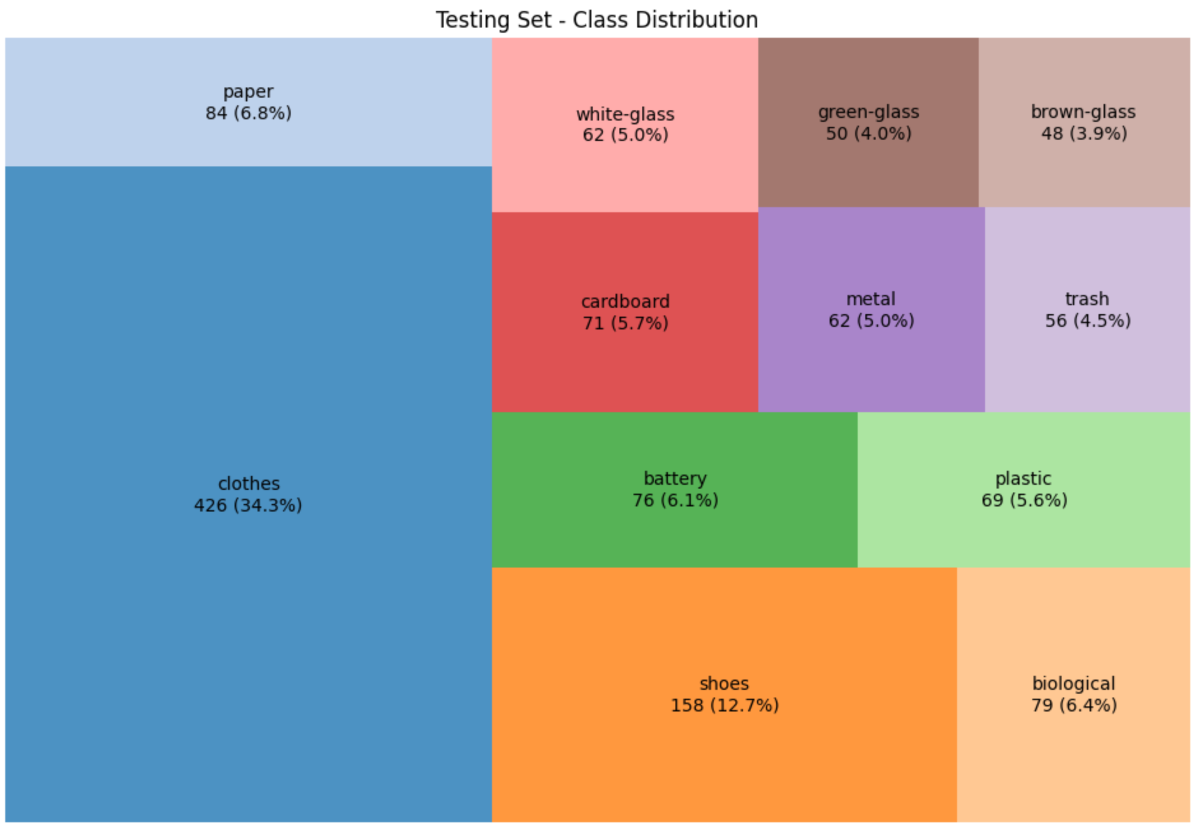


Figure 4 - Testing Set Class Distribution (the percentage is the same for each class in different sets)

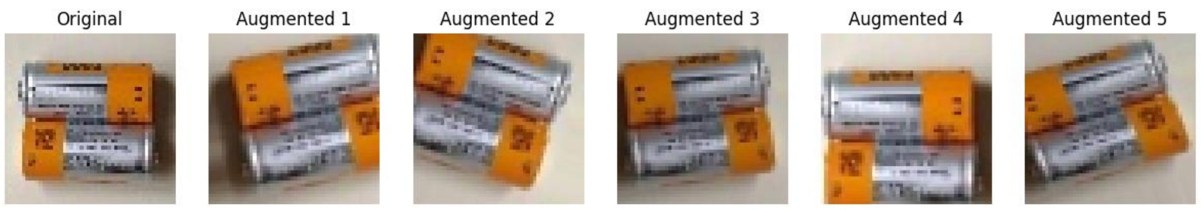
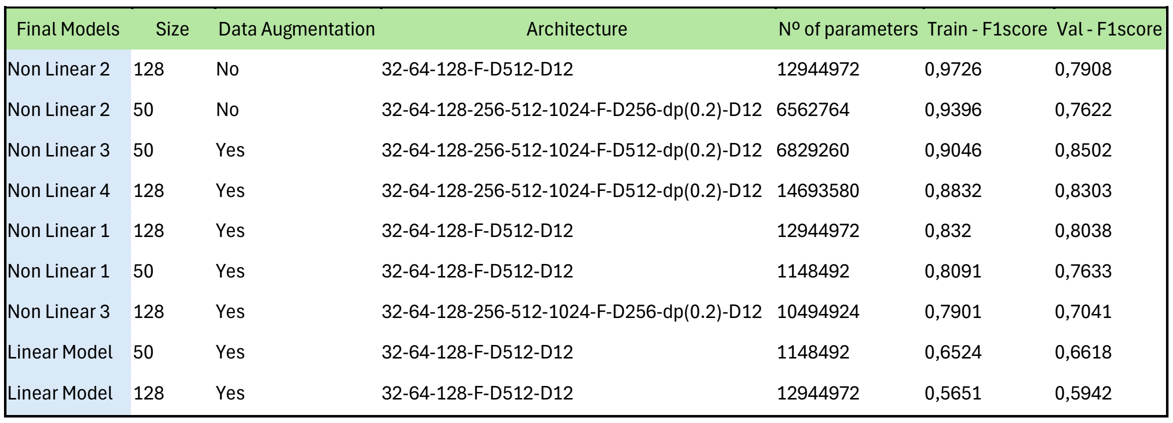
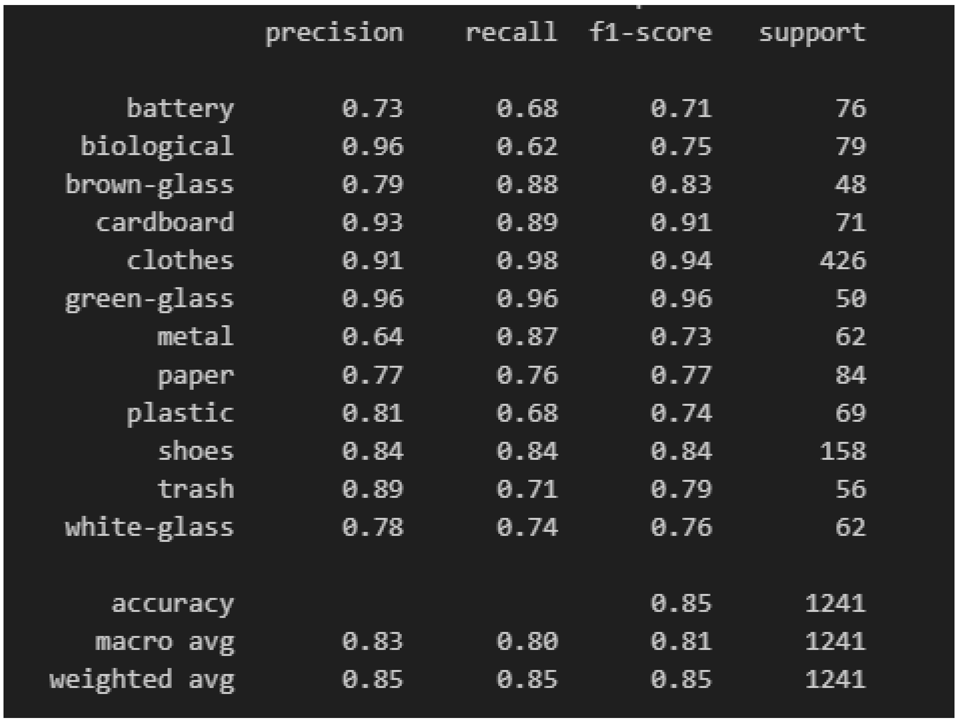


Figure 5 - Image Transformations



Final Models	Size	Data Augmentation	Architecture	Nº of parameters	Train - F1score	Val - F1score
Non Linear 2	128	No	32-64-128-F-D512-D12	12944972	0,9726	0,7908
Non Linear 2	50	No	32-64-128-256-512-1024-F-D256-dp(0.2)-D12	6562764	0,9396	0,7622
Non Linear 3	50	Yes	32-64-128-256-512-1024-F-D512-dp(0.2)-D12	6829260	0,9046	0,8502
Non Linear 4	128	Yes	32-64-128-256-512-1024-F-D512-dp(0.2)-D12	14693580	0,8832	0,8303
Non Linear 1	128	Yes	32-64-128-F-D512-D12	12944972	0,832	0,8038
Non Linear 1	50	Yes	32-64-128-F-D512-D12	1148492	0,8091	0,7633
Non Linear 3	128	Yes	32-64-128-256-512-1024-F-D256-dp(0.2)-D12	10494924	0,7901	0,7041
Linear Model	50	Yes	32-64-128-F-D512-D12	1148492	0,6524	0,6618
Linear Model	128	Yes	32-64-128-F-D512-D12	12944972	0,5651	0,5942

Figure 6 - Models and Characteristics



```
               precision    recall  f1-score   support

      battery       0.73      0.68      0.71        76
   biological       0.96      0.62      0.75        79
  brown-glass       0.79      0.88      0.83        48
    cardboard       0.93      0.89      0.91        71
      clothes       0.91      0.98      0.94       426
  green-glass       0.96      0.96      0.96        50
        metal       0.64      0.87      0.73        62
        paper       0.77      0.76      0.77        84
      plastic       0.81      0.68      0.74        69
        shoes       0.84      0.84      0.84       158
        trash       0.89      0.71      0.79        56
  white-glass       0.78      0.74      0.76        62

     accuracy                           0.85      1241
    macro avg       0.83      0.80      0.81      1241
 weighted avg       0.85      0.85      0.85      1241
```

Figure 7 - Evaluation Metrics for Test Set

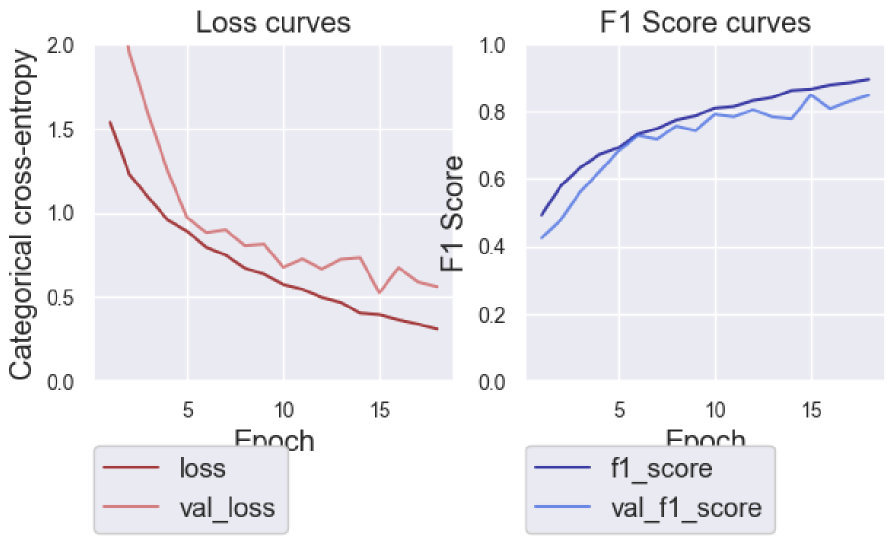
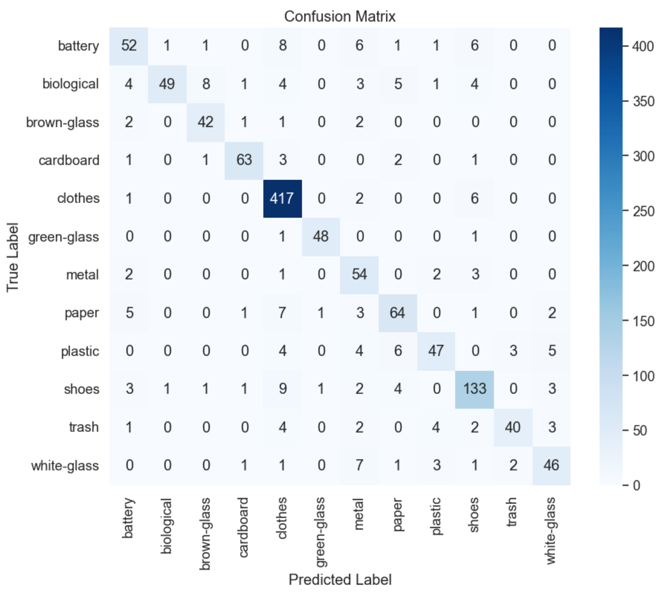


Figure 8 - Loss and F1 curves for Final Model after Manual Grid Search



True Label \ Predicted Label	battery	biological	brown-glass	cardboard	clothes	green-glass	metal	paper	plastic	shoes	trash	white-glass
battery	52	1	1	0	8	0	6	1	1	6	0	0
biological	4	49	8	1	4	0	3	5	1	4	0	0
brown-glass	2	0	42	1	1	0	2	0	0	0	0	0
cardboard	1	0	1	63	3	0	0	2	0	1	0	0
clothes	1	0	0	0	417	0	2	0	0	6	0	0
green-glass	0	0	0	0	1	48	0	0	0	1	0	0
metal	2	0	0	0	1	0	54	0	2	3	0	0
paper	5	0	0	1	7	1	3	64	0	1	0	2
plastic	0	0	0	0	4	0	4	6	47	0	3	5
shoes	3	1	1	1	9	1	2	4	0	133	0	3
trash	1	0	0	0	4	0	2	0	4	2	40	3
white-glass	0	0	0	1	1	0	7	1	3	1	2	46

Figure 9 - Confusion Matrix

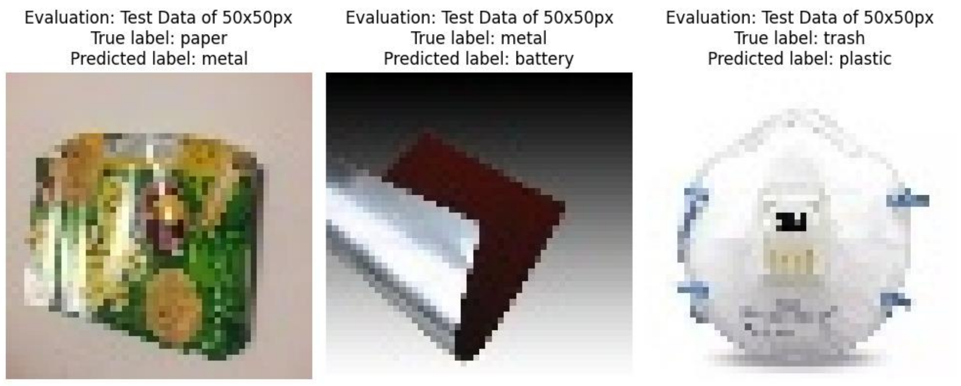


Figure 10 - Misclassified Images