Machine Learning
University of La Sapienza
Fall 2023
Rome, Italy

Homework 2

Alessia Pontiggia 1892079

1 Introduction

The task of this second homework is a multiclass (5 classes) classification problem on a car racing images dataset. The methodology and solutions used are explained following a sequential order, corresponding also to the code order (code available at Colab and Kaggle notebook).
The workflow followed is:

1. Data Preprocessing
2. Models
3. Conclusions

NB: For each architecture, the model is tested and evaluated. Models are tested more than once (also with different parameters) and since the splitting process is random, thus there might occur some inaccuracies from the report's results and those on the notebook. Due to GPU usage limitations, two notebook (one on Google Colab and the other on Kaggle) have been used.

2 Data Preprocessing

The preprocess of both datasets is indeed a crucial aspect to consider. Let's analyize in detail the operations done at this step.

- **Dataset loading**: in order to use both datasets, they need to be downloaded in a proper format.
- **Dataset normalization**: when the range of values on which data is distributed is too big, then there might be instability in the training, so it is a good practice normalizing data.
- **Dataset balance**: typically we want the training process not to be affected by the number of samples belonging to one class wrt another. (E.g. if the dataset has few samples for a class, so few information available, then the learning process for that label can be incomplete and can produce some errors in test step).
- **Dataset splitting**: As to perform training, validation and test steps, the separation of the dataset into training and test sets in a random way is done.

2.1 Datasets

In this section there will be discussed the preprocessing steps listed in the previous page.

2.1.1 Loading

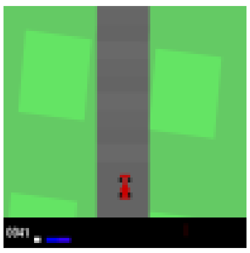
After the download of the dataset we can distinguish between:

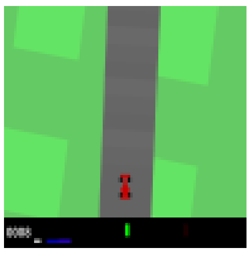
- **Training set**: containing 6347 images
- **Test set**: containing 2047 images

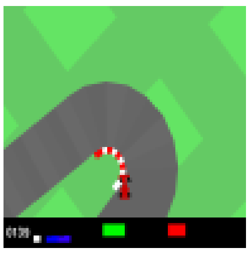
Each image has size (96,96,3), where the first two dimensions represent height and width of the image, while the third one represents the number of channels. In this case, the dataset contains RGB colored images, so they have 3 channels. Each image is attached to a label, an integer going from 0 to 4, representing the action the car should take in that particular frame:

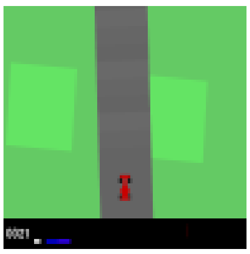
- **0**: do nothing;
- **1**: steer left;
- **2**: steer right;
- **3**: gas;
- **4**: brake;

In order to analyze the dataset, let's represent a random sample for each label:


(a) Label 0: nothing


(b) Label 1: steer left


(c) Label 2: steer right


(d) Label 3: gas

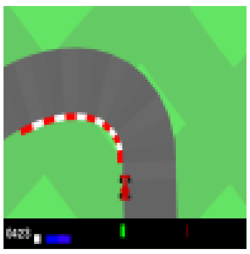

(e) Label 4: brake

Figure 1: Samples

As the figures show, the relevant features are the red car, the grey road and the green environment around. It is obvious that random samples from class 0, 1 and 3 are really similar to each other and the same can be said for class 2 and 4. This consideration makes think of an harder learning process.

2.1.2 Normalization

As normalization technique, each pixel value is scaled by 1/255 in order to have pixels in range [-1,1]. In this case, the normalization is done with ImageDataGenerator function from TensorFlow library.

2.1.3 Balance

As the figures suggest, test set is surely unbalanced, while training set is slightly unbalanced.

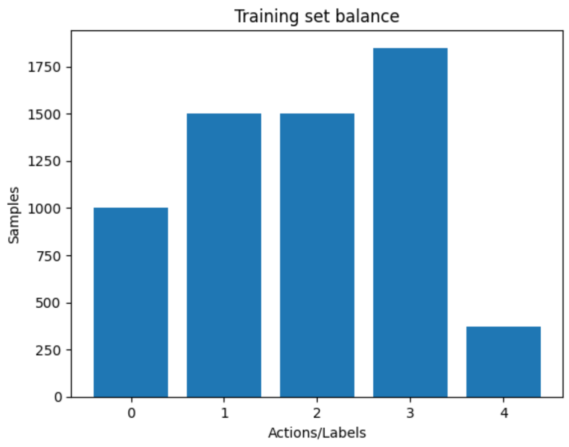


Figure 2: Train

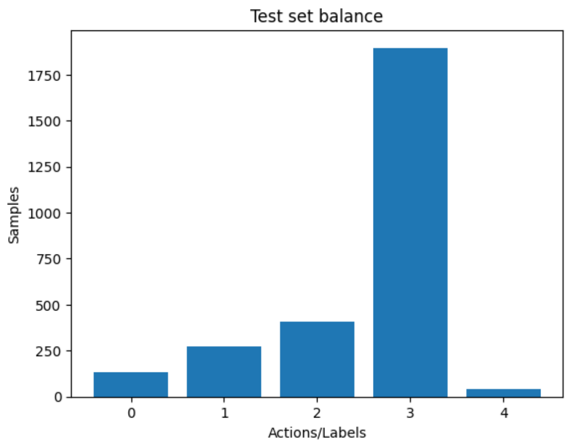


Figure 3: Test

At this step, no data augmentation is done, as the model will be first tested with the original dataset. Later on, the augmentation will be performed.

2.1.4 Splitting

By using the ImageDataGenerator tool, it is possible to set the ratio between training and validation sets (the first one will have 80% of the starting dataset, the second one the remaining 20%).

3 Models

To face this multiclass classification problem, some models have been used and tested, differing in some approaches, but relying on the same main component: convolutions. CNN are used because of their good properties with images.

3.1 Workflow Description

In each architecture the hyperparameters used after a phase of sperimental attempts, reported in brackets, are:

- **Epochs** = 30 (from 5 to 30);
- **Learning Rate** = 0.001 (from 0.01 to 0.0001);
- **Optimizer** = Adam (SGD);
- **Training batch size** = 256 (64, 128), Shuffle = True;
- **Validation batch size** = 64 (128), Shuffle = False;
- **Test batch size** = 1, Shuffle = False.

3.1.1 Different approaches

In total, 4 models have been tested: two of them share the same architecture but different regularization terms; the third model uses a shallower architecture, and the fourth one benefits from data augmentation technique. Here, there are listed the models differing in these aspects:

1. **Regularization terms**: Two models implemented differ with respect to the regularization term used, but rely on the same following architecture:

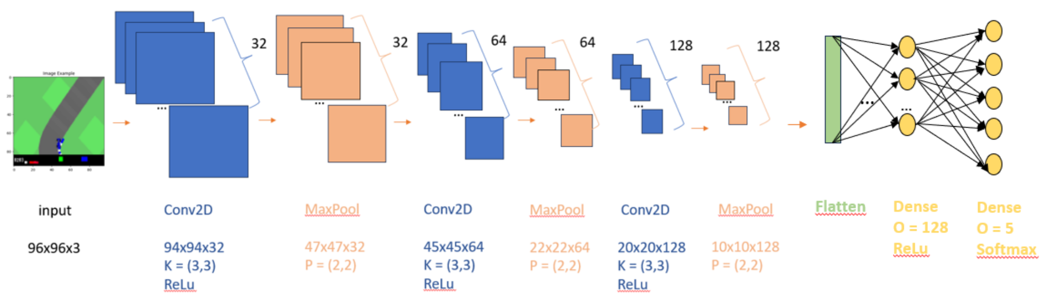


Figure 4: Architecure Design

Drop Out: the first model only uses the drop out technique between the two dense layers with a percentage of "turned off" neurons of 0.5;
Early Stopping: the second model uses both DropOut and Early Stopping techniques. The last one monitors the validation loss and stops the model when the

patience (= 3) is reached, if so the best weights are stored, else it continues the iterations until the maximum number of epochs is reached.

2. **Shallower Architecture**: In this case, the architecture is lighter with respect to the one above, since the number of convolutional layers is 2 (instead of 3). The design of the resulting CNN is:

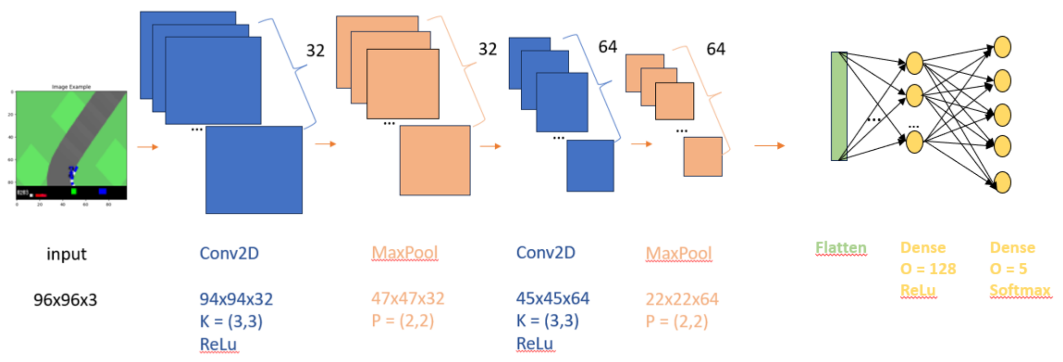


Figure 5: Architecure Design

3. **Data Preprocessing: Augmentation**: In the previous points there wasn't applied any preprocessing on data (excpet for normalization). But in this approach, the augmented dataset is used with the techniques and the model choices resulting to be the best so far.

3.1.2 Evaluation

For each model the evaluation consists in analyzing some classification metrics:

- **Loss** and **Accuracy** flow in training and validation sets
- **Loss** and **Accuracy** on test set
- **Confusion matrix**
- **Classification report**

3.2 Model 1: Drop Out

This technique is used because it prevents overfitting by turning off some neurons of the dense layers allowing the model not to "learn" their information, since they don't provide it.

- **Loss and Accuracy** (training and validation):

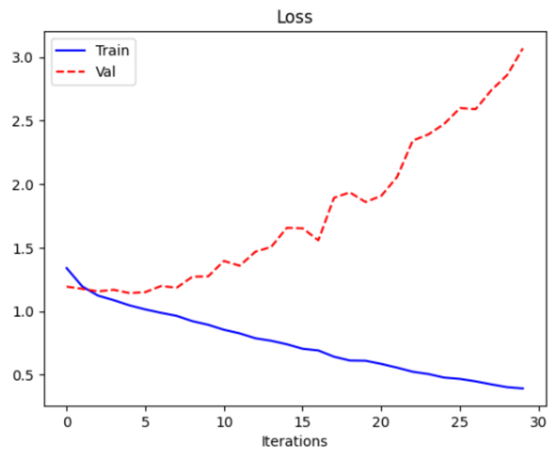


Figure 6: Loss

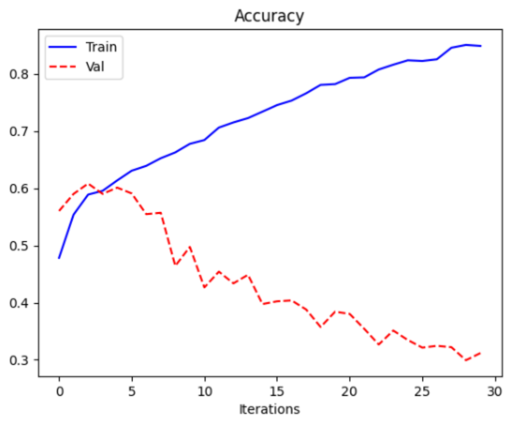


Figure 7: Accuracy

  With this approach it is clear that by increasing the number of epochs, training loss converges to zero but validation loss increases after few epochs. Also the accuracy on training gets good results but the one on the validation set gets bad. At this point, it is plausible to think that the model overfits, even if the drop put technique is used.

- **Loss on test set**: 2.55. This value is quite larger than the loss got on the training set. This confirm the intuition made before.

- **Accuracy on test set**: 48.93%. With only this metric it's not possible to judge how the classification works, since the task has 5 classes. So, it is required to analyze the entire classification with confusion matrix and classification report.

- **Confusion Matrix and Classification Report**:

  Both confusion matrix and classification report underline better results in predicting the class 3 (gas). But, at the same time, the majority of samples are misclassified with class 3. This can be caused by the fact that the majority of samples belongs to this label both on training and test sets. So the model has learnt and extracted features in a better way since it has more data available. The same happens with class 4 (brake), but in the opposite sense, since the test set contains only 39 samples and the misclassifications occur in a high percentage due to the low number of samples. With this in mind, it is useful to analyze the **weighted average** of precision, recall and f1-score are respectively: 69.2%, 48.9%, 53%. This means that, by taking into account class imbalances, the percentages of these 3 metrics increase.

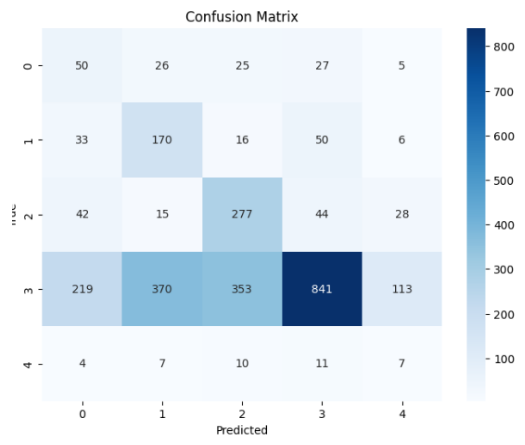


Figure 8: Confusion Matrix

Precision	Recall	F1-score
0.144	0.376	0.208
0.289	0.618	0.394
0.407	0.682	0.510
0.864	0.444	0.586
0.044	0.179	0.071

Figure 9: Classification report

3.3 Model 2: Drop Out and Early Stopping

Since the increasing number of epochs was a problem in the previous approach as the risk of overfitting occurred, with Early Stopping the model stops learning when the validation loss doesn't decrease for 3 (patience) times its value. With this approach the number of iterations reached is 10 over 30.

- **Loss and Accuracy** (on training and validation):

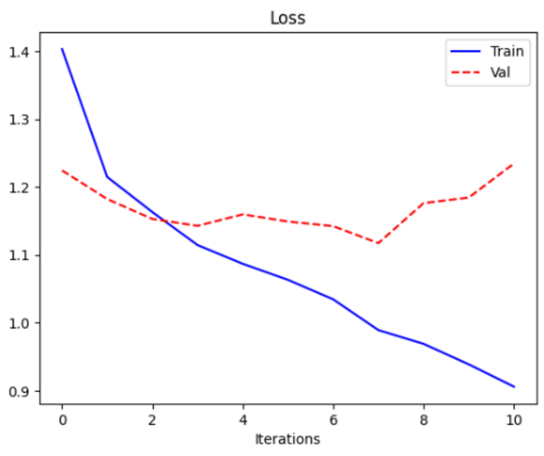


Figure 10: Loss

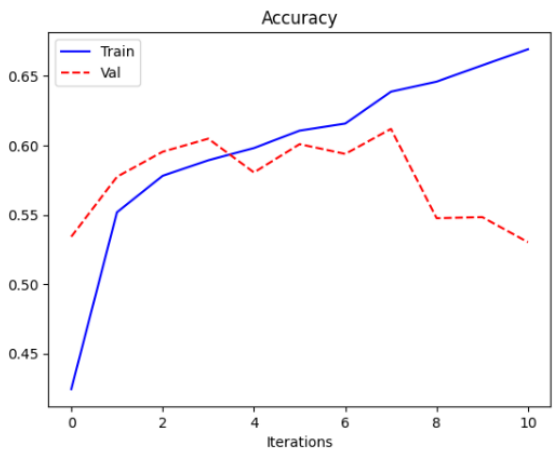


Figure 11: Accuracy

As predictable, since we are "cutting off" the epochs where the model would have started to overfit, the flow of both training and validation loss decreases and accuracy increases over time, even without reaching optimal results. With early stopping we ensure that the best weights, the one producing the minimum value of loss, are stored. So the best weights are used to compute the predictions on the test set.

- **Loss on test set**: 0.99. It has decreased significantly. It lies between the one on training and validation.
- **Accuracy on test set**: 65.81%. This result is much better than before, but again, we need to check both confusion matrix and classification report.

- **Confusion Matrix and Classification Report**:

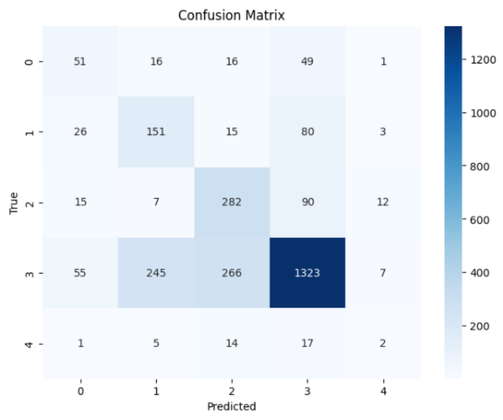


Figure 12: Confusion Matrix

Precision	Recall	F1-score
0.345	0.383	0.363
0.356	0.549	0.432
0.476	0.695	0.565
0.894	0.698	0.766
0.080	0.051	0.062

Figure 13: Classification report

Here, confusion matrix and classification report confirm the behavior seen before. The class predicted better is the third one. In this case the class 4 has even lower correct classifications. In this case, the **weighted average** of precision, recall and f1-score are respectively: 70.9%, 65.8%, 67.3%. So recall and f1-score have significantly increased: the model is better at classifying positive instances.

3.4 Model 3: Shallower Architecture

Another type of overfitting avoidance can be implementing a shallower architecture. Since the last model (the one with Drop Out and Early Stopping both) got better result with respect to the first one, then also this approach uses these regularization techniques.

- **Loss and Accuracy** (on training and validation):

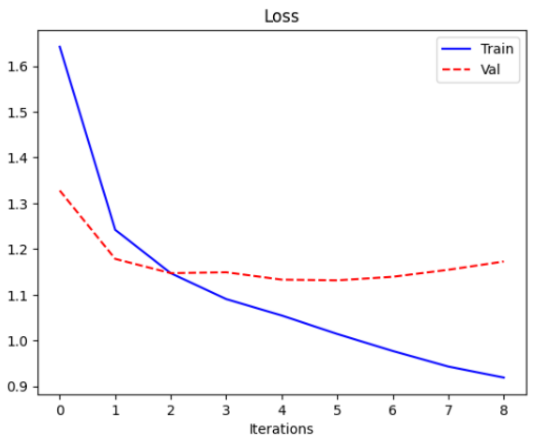


Figure 14: Loss

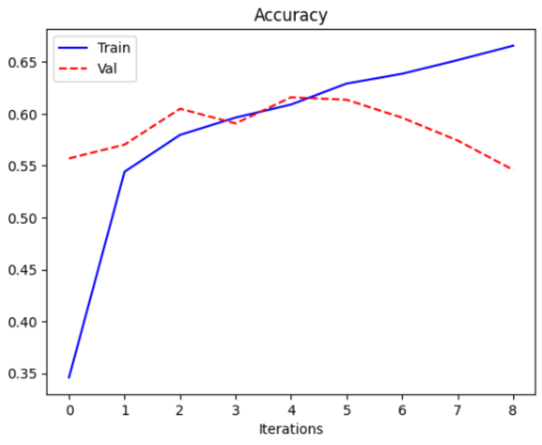


Figure 15: Accuracy

Also in this case, the flow of training loss decreases. The one on validation remains almost constant. Accuracy increases over time, even without reaching optimal results.

- **Loss on test set**: 1.06.

- **Accuracy on test set**: 63.22% Combining both loss and accuracy and the execution time, this model can be seen as the best of the combinations tried up to now.
- **Confusion Matrix and Classification Report**:

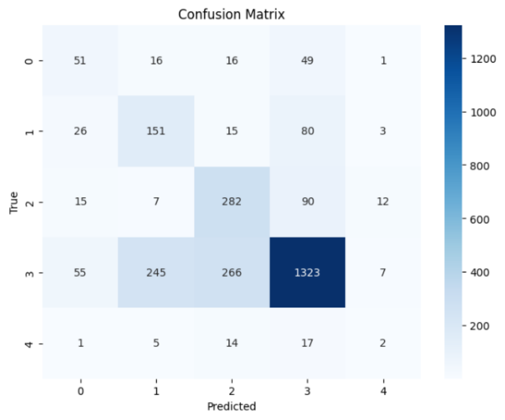


Figure 16: Confusion Matrix

Precision	Recall	F1-score
0.314	0.444	0.368
0.343	0.578	0.430
0.449	0.732	0.557
0.865	0.643	0.738
0.115	0.077	0.092

Figure 17: Classification report

Same considerations done in the previous model can be extended here. In this case, the **weighted average** of precision, recall and f1-score are respectively: 71.4%, 63.2%, 65.3%.

3.5 Model 4: Data Augmentation, Drop Out, Early Stopping and Shallower Architecture

In this section, the best combinations of regularization terms and architectures will be used in the following approach. It's used the shallower architecture because it's the fastest among all the others and the flow of accuracy and loss over time is more stable.
Moreover, data augmentation is performed over the training set in order to generalize the dataset with respect to some variations that can occur in real test samples.
The techniques used are grouped up according to the model where they have been applied:

1. Model 1:
   - Brightness in range [0.75, 1.25]
   - Zoom range: 0.6
2. Model 2:
   - Brightness in range [0.75, 1.25]
   - Vertical flip
   - Horizontal flip

3.5.1 Brightness, Zoom

The techniques used just want to adjust and vary the colour brightness and the zoom visualization of the images. This may help the models to have a focus on the car and its orientation with respect to the road.

- **Loss and Accuracy** (on training and validation):

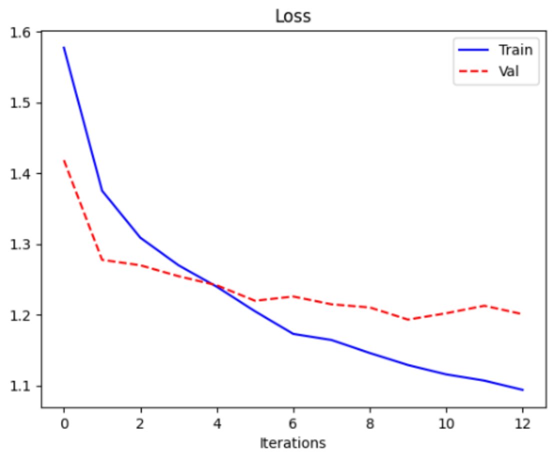


Figure 18: Loss

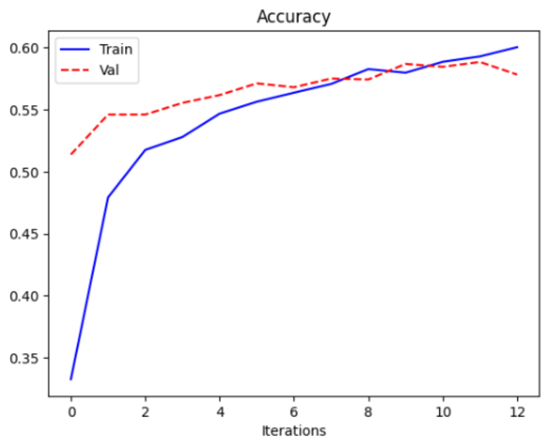
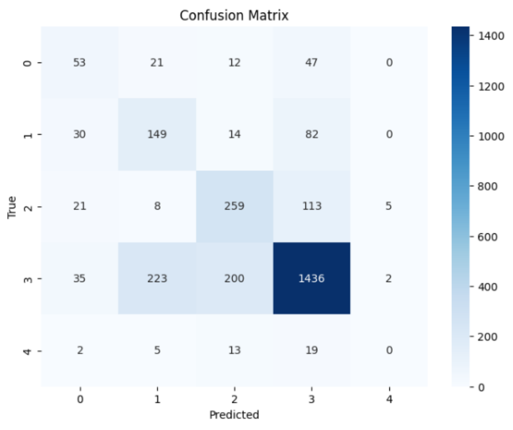


Figure 19: Accuracy

The flow of training and validation loss decreases. Accuracy increases over time, even without reaching optimal results.

- **Loss on test set**: 1.04.
- **Accuracy on test set**: 67.84%.
- **Confusion Matrix and Classification Report**:



True \ Predicted	0	1	2	3	4
0	53	21	12	47	0
1	30	149	14	82	0
2	21	8	259	113	5
3	35	223	200	1436	2
4	2	5	13	19	0

Figure 20: Confusion Matrix

Precision	Recall	F1-score
0.380	0.368	0.374
0.356	0.520	0.422
0.491	0.658	0.562
0.840	0.742	0.788
0.000	0.000	0.000

Figure 21: Classification report

In this case, the **weighted average** of precision, recall and f1-score are respectively: 70.6%, 67.8%, 68.7%.

3.5.2 Brightness, Vertical Flip and Horizontal Flip

Here, the intuition is extending the samples of each class to be as similar as possible to the ones of other classes. Let's think of a car taking a curve to the left: if this image is rotated of a proper angle (180 degrees), it seems the car has to curve to right. It's an extreme approach that may lead to better generalization.

- **Loss and Accuracy** (on training and validation):

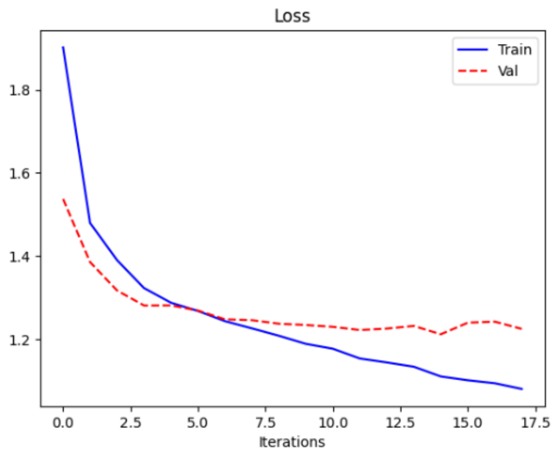


Figure 22: Loss

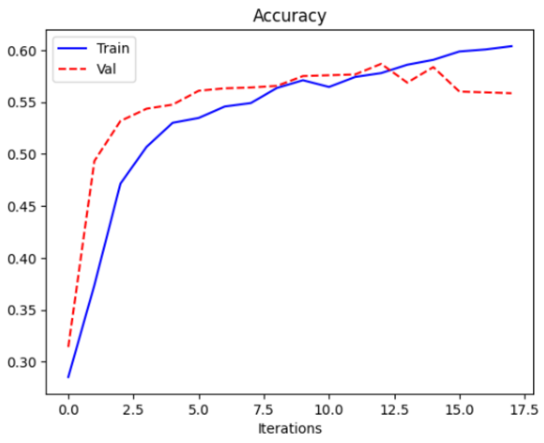
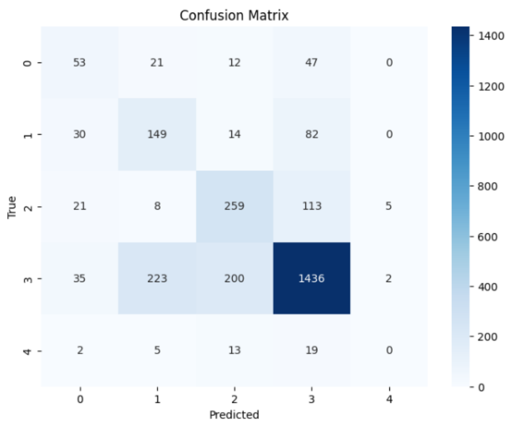


Figure 23: Accuracy

Also in this case, the flow of training loss decreases. The one on validation remains almost constant. Accuracy increases over time, up to 70%.

- **Loss on test set**: 1.03.
- **Accuracy on test set:** 69.01%. Accuracy reaches its peak.
- **Confusion Matrix and Classification Report**:



True \ Predicted	0	1	2	3	4
0	53	21	12	47	0
1	30	149	14	82	0
2	21	8	259	113	5
3	35	223	200	1436	2
4	2	5	13	19	0

Figure 24: Confusion Matrix

Precision	Recall	F1-score
0.376	0.398	0.387
0.367	0.542	0.438
0.520	0.638	0.573
0.846	0.757	0.799
0.000	0.000	0.000

Figure 25: Classification report

Here the class 4 has no correct classifications. In this case, the **weighted average** of precision, recall and f1-score are respectively: 71.5%, 69%, 69.8%. This is the best result got so far.

4 Conclusions

The models proposed above show the difficulties a task can present. In this dataset, the main struggle was the similarity of samples belonging to different classes. As the Figure 1 shows, it is really demanding extracting significant and different features for them all. This can cause misclassifications between all classes because all of them can present the same characteristics.
Given this big drawback, the models implemented tried to show an overview of the possible techniques to reach good performances, despite the difficulties. The choice of the architecture, hyperparameters, regularization terms and data augmentation, have been crucial to get discrete results, starting from <50% up to 70% of accuracy.