1. **Results**

As said in II.iii, many different CNN architectures were implemented. Within those configurations, also different preprocessing steps were applied. The process of reaching a final model included a vast trial and error methodology, always following relevant literature.

The first step taken in training was to implement a basic model with two convolutional layers each followed by a max pooling layer plus two final dense layers following a flatten layer. This simple architecture could give the authors an idea of which would be the starting point. In this try the authors worked only with a fraction of the available data in order to preserve

After looking at the first results the authors rapidly identified large amounts of overfitting (figure 1). Due to the amount of overfitting experienced the authors were forced to tackle that issue, having found two main ways of doing it: adding dropout layers (before the dense layers) and using data augmentation.

Chart, line chart

Description automatically generated

Figure 1 - Base Model vs Base + Dropout vs Base + Dropout + Data Augmentation

As seen in figure 1, none of the models really outperformed the others. It’s also fair to say that the worst performance belonged to the model where data augmentation was applied. Since no great improvements were achieved the authors decided that using all data for training could be one good solution.

Considering what is said in II.iii about training images series, the authors were aware that it could pose some issues regarding generalization capability of the model, therefore, the dropout layer was kept. In the first models trained using all data two models were tested in order to decide which would be the best choice for the dropout rate. The values used in this testing were 0.3 and 0.9 (Figure 2).

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Figure 2 – Base Model with Dropout Rate equal to 0.3 and 0.9

After a brief look at the results obtained using all data, the authors quickly found that using either one of the two values a significant part of the overfitting would disappear.

After getting these results, using class weights in training was tested. However, the results obtained didn’t differentiate significantly. Having that into consideration, it was decided that the weighting could be a final step to take when a final architecture was found.

Moreover, and with the objective of increasing performance on the validation set, the authors decided to test the model using one more convolutional layer.

The configuration of the model at this point is shown in Figure 3.

Table

Description automatically generated

Figure 2 – Model Architecture After Adding 1 Convolutional Layer

In addition to the third convolutional layer added, that as mentioned above was used to increase performance, also grayscale was applied. The latter was introduced as a new attempt to tackle overfitting.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated