"Introduction to Machine Learning" project Report

Tommaso Manzana
University of Trento
Via Sommarive, 9
tommaso.manzana@studenti.unitn.it

1. Introduction

The objective of the project was to develop an algorithm able to classify galaxies in 10 different classes, provided an RGB image of them. Example of possible classes: Edge-on with Bulge, Barred Spiral, Round Smooth, ...
The dataset consisted of 17736 images, of which 12415 used for the training set, whereas 5321 belonged to the test set. Training set images were already supplied with their respective labels. Since the ground-truth for the test data was unknown the accuracy was estimated on the validation data, which was obtained through a split of the training one.
Two metrics were used to asses the performance:

Sample wise accuracy

$$A_{sw} = \frac{1}{N}\sum_{n=0}^{N} 1(y_n == p_n)$$

where $1()$ is an indicator operator, y_n is the ground truth labels and p_n is the label predicted by the algorithm.

Class wise accuracy

$$A_{cw} = \frac{1}{|C|}\sum_{c\in C} A_c$$

$$Ac = \frac{TP_c}{TP_c + FN_c}$$

2. Proposed Method

The provided images where 256 x 256 matrices of RGB values. Given the high number of features some traditional machine learning models such as kNN or SVM would suffer greatly on accuracy, incur in over-fitting and slow training time. The most favorable technique were neural networks, more specifically during the course we were encouraged to implement Transfer learning, meaning we would use pre-trained models like ResNet or VGG in order to solve our Galaxy Classification task. A major hurdle was the amount of data to work with, NNs perform greatly when provided with abounding data samples whereas our dataset was narrow; a higher accuracy would most definitely been achieved if the dataset were to be substantially bigger.
Lastly the dataset, as for a real world example, was imbalanced; to mitigate this issue the classes' weights were inversely proportionate to their frequency of appearance.

3. Results

In order to better tackle the given task a simpler approach was tested at first with a k-NN model, the complexity of the whole project was then gradually increased.

3.1. Nearest Neighbors

As premised k-NN wasn't a good candidate for such an assignment, several hyper-parameters were tested but the A_{sw} hovered around 0.4. In order to achieve a higher accuracy further time and space complexity was need but it wasn't worth the trade.

3.2. ResNet-18

First and foremost the smallest ResNet architecture was implemented: ResNet-18, an 18 layers deep convolutional neural network composed of 1 7x7 convolution , 16 3x3 convolution and one linear layer. The train dataset needed to be split into train and validation sets, 9700 and 2715 respectively. The validation dataset was extrapolated from the training one since the groud-truth labels of the test images was unknown, in order to evaluate the proposed model before submitting the complete classification done on the test dataset. The train dataset was divided in minibatches of size 32 and Adam used as optimization function, after a couple of runs it was switched to stochastic gradient descent. Cross-Entropy Loss was implemented as the loss function given its performance for the classification of images. In Figure 1 the first results of the model with the best A_{sw} at 0.72, already significantly higher than the kNN approach.

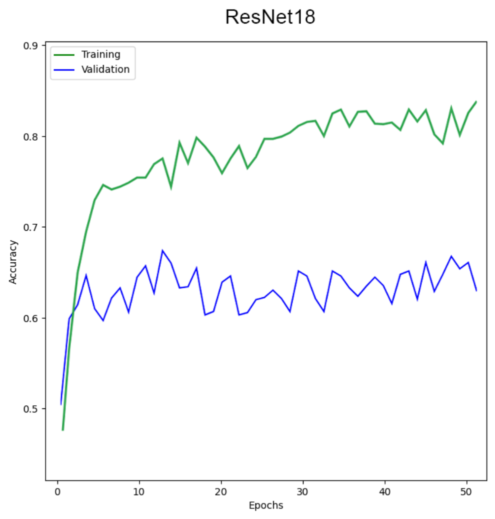

Figure 1. Training and Validation A_{sw} ResNet18

3.3. ResNet-50

As another attempt to increase the accuracy ResNet50 was imported. The main difference from ResNet18 is the amount of layer of the NN. The switch itself brought limited improvements so a series of tuning tests [3] were performed: different batches' sizes 16, 32, 64, different learning rates 0.01, 0.001, 0.0001, as well as changing SGD momentum and weight decay to mitigate over-fitting. Figure 2 shows the different tweaks performed.

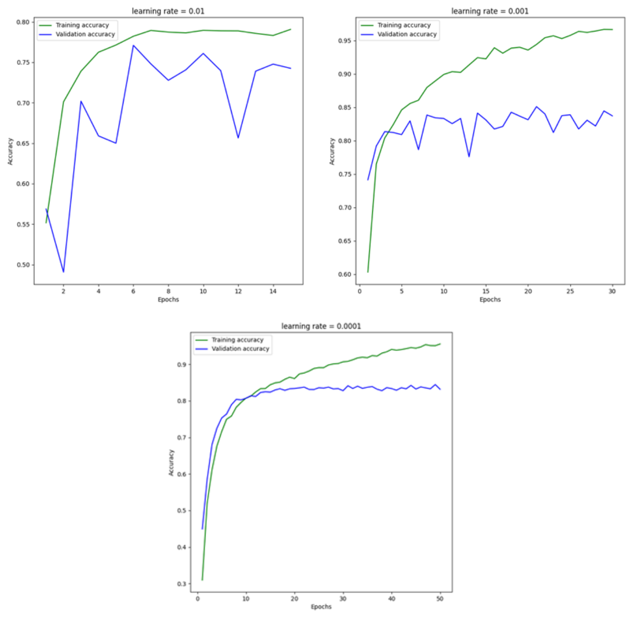

Figure 2. Training and Validation A_{sw} with different lr

Further improvements were achieved by artificially generating more samples for the image features set with the use of mirroring and rotation [2]. Although greyscaling was used at first it was quickly dropped. Consequently a learning rate scheduler was implemented, after trying to use a simple ReduceLROnPlateau, CosineAnnealing [1] was later introduced, as well as the "linear warm up" variation of the latter. As can be seen in Figure 3 there was a great improvement in performance.

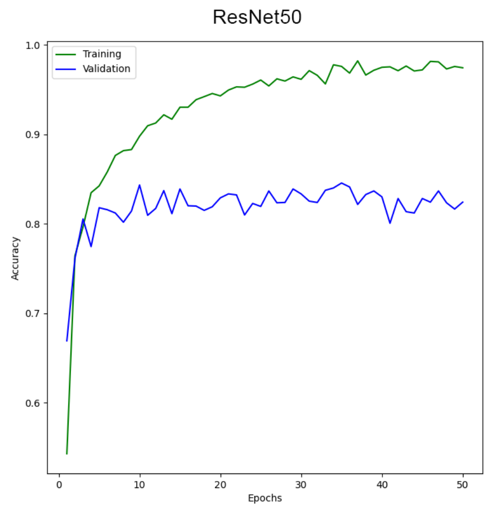

Figure 3. Training and Validation A_{sw} ResNet50

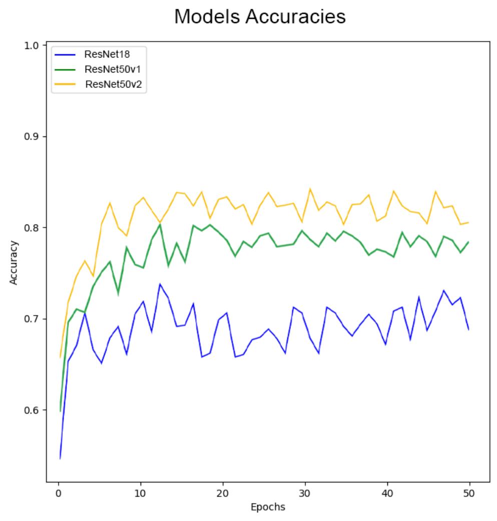

Figure 4. A_{sw} value on the validation, all models

The best performing model was saved after each training batch, the best one so far implemented all the mentioned features and achieved an A_{sw} of 84.6 and an A_{cw} of 82.3 on the test set. Figure 4 shows the improvements in accuracies comparing the best model of each version tested. The use of stratification to split the data might bring some improvements to the accuracy, but more test are needed to verify this.

References

[1] G. Garg. Exploring learning rates to improve model performance in keras. 2019.

[2] J. T. Jia-Ming Dai. Galaxy morphology classification with deep convolutional neural networks. 2018.

[3] S. konda. Improving the performance of resnet50 graffiti image classifier with hyperparameter tuning in keras. 2020.