Machine Learning project Report

Luca De Menego
University of Trento
luca.demenego@studenti.unitn.it

Source Code in Colab Notebook

1. Introduction

The objective of the project was to develop an algorithm able to identify, given an RGB image of a galaxy as input, the morphological class it belongs to. In details, we had 10 possible classes: *disturbed*, *merging*, *round smooth*, *in-between round smooth*, *cigar round smooth*, *barred spiral*, *unbarred tight spiral*, *unbarred loose spiral*, *edge-on with bulge* and *edge-on without bulge*. The dataset provided consisted of 17.736 images of size 256×256, divided in train set (12.415 labelled examples) and test set (5.321 non-labelled examples). However, 20% of the training set has been used as validation set, and below accuracies were calculated based on it.

We had two main metrics used to estimate the performance of our algorithms:

- **sample-wise accuracy**: correct predictions with respect to the total number of samples;

$$A_{sample-wise} = 1/N \sum_{n=0}^{N} 1(y_n == p_n) \quad (1)$$

- **class-wise accuracy**: average of the accuracies on each category.

$$A_{class-wise} = 1/|C| \sum_{c \in C} (TP_c)/(TP_c + FN_c) \quad (2)$$

  TP_c: number of images which have been correctly assigned to category c;

  FN_c: number of images which have been incorrectly assigned to category c.

2. Proposed Method

The provided images are basically composed by $256 \times 256 \times 3 = 196.608$ features, so the use of shallow methods is obviously discouraged without a valid pre-made feature extraction. Both pre-trained torchvision models and color histogram have been tested to extract features, but they lead to low sample-wise accuracy (see. Fig.3), probably due to the fact that:

1. the provided torchvision models were trained on the classification of classes that were too much different from galaxies, so the extracted features were not accurate enough;

2. a simple distribution of colors can't accurately classify the types of galaxies we had, as there are other important features that in this way we don't consider (e.g. the shape of the galaxies).

This is why the chosen approach were neural networks. In particular, the one used in our case was ResNet, a CNN with a peculiar feature: it uses *shortcut connections*, primarily to avoid the vanishing gradient problem and perform better with more layers [1].

When dealing with NNs, we should have a lot of data to work with, and a valid balancing of examples per class. To address these lacks, the proposed solution uses data augmentation, with a pre-defined set of transformations each image can have, and a weighted sampler. The final ResNet50 model scored with test set data $A_{sample-wise} = 86.4\%$ and $A_{class-wise} = 84.8\%$ (Fig.1).

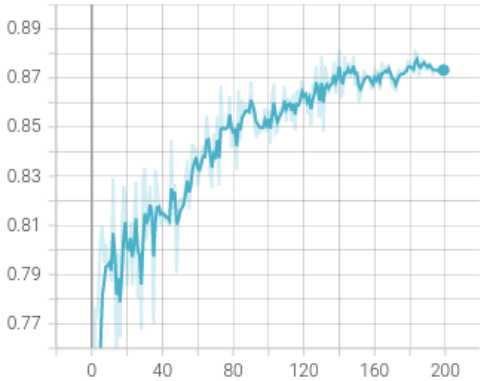

Figure 1. ResNet50 sample-wise accuracy on validation data.

3. Results

The following methods have been tested: *shallow methods* (Support Vector Machines, k-NN, Random Forests) with different types of feature extractors and *deep methods*, using ResNet18 and ResNet50 with an extensive tuning of the hyperparameters.

3.1. Shallow Methods

3.1.1 Resized Images

Given the fact that the provided images were too big for shallow methods, the first thing that has been tested was to manually reduce the number of features of each example with a simple resize. In particular, each image has been cropped in the center ($150 \times 150px$) and then resized to $30 \times 30px$. In fact, in the majority of our examples, the galaxies were in the middle of the image. This lead us to the following results:

- **k-NN**: $A_{sample-wise} = 50.9\%$, with $k = 5$;
- **Random Forests**: $A_{sample-wise} = 59.1\%$, with $n.trees = 100$ and no set max depth;
- **SVM**: $A_{sample-wise} = 54.1\%$, with a regularization parameter of 1.0.

3.1.2 Color Histogram

The features got from a color histogram alone lead to pretty low results:

- **k-NN**: $A_{sample-wise} = 23.2\%$, with $k = 5$;
- **Random Forests**: $A_{sample-wise} = 34.3\%$, with $n.trees = 100$ and no set max depth;
- **SVM**: $A_{sample-wise} = 24.2\%$, with a regularization parameter of 1.0.

The distribution of colors, without any other type of feature, can't be enough to accurately classify the types of galaxies. That's why new methods have been tried, without wasting too much time with this kind of solution.

3.1.3 Pre-trained models as feature extractors

A pre-trained ResNet18 provided by torchvision has been tried in this phase. The last fully connected layer was removed, leading to an extraction of features that gave as result 512 features per example. The results were the following:

- **k-NN**: $A_{sample-wise} = 34.3\%$;
- **Random Forests**: $A_{sample-wise} = 39.0\%$;
- **SVM**: $A_{sample-wise} = 48.2\%$.

The problem was how this model was trained: it was trained on classes too much different from galaxies, so the extracted features were not enough representative of the classes we actually had. In fact, when using our ResNet18 model trained on our classes as feature extractor, the performance of these shallow methods was comparable to the performance obtained by the Neural Network itself (e.g. with SVM: $A_{sample-wise} = 84.7\%$ - Fig.2).

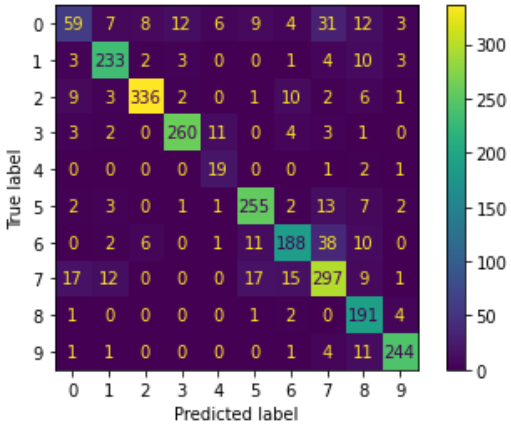

Figure 2. SVM confusion matrix using our ResNet18 model as feature extractor. We can see low results in *Disturbed* and *Cigar Shaped Smooth* classes, which had few examples in the train set.

3.2. Neural Networks

As stated before, the proposed method dealt with class imbalance with a weighted sampler, and used data augmentation to make the model able to generalize more. In particular, the following transformations were applied:

- random crop ($224 \times 224px$);
- random rotations or flips;
- random affine;
- random perspective;
- gaussian blur;
- change colors jitter;
- normalization.

Moreover, the weight of each class i was calculated as:

$$ClassesWeights[i] = 1.0/ClassesCounts[i] \quad (3)$$

where ClassesCounts is an array containing the number of examples each class had.

After some tests, Adam optimizer seemed to be the best choice, as well as CrossEntropyLoss as loss function and LinearWarmupCosineAnnealing as scheduler.[2][3] The results below were obtained with 200 epochs and batches of size 128. In details:

- Adam Optimizer started with a learning rate of 0.001;
- LinearWarmupCosineAnnealing was initialized with 50 warmup epochs. During these epochs, the scheduler started with a learning rate of 0, increasing it in a linear way. Then, it followed a cosine annealing schedule between the values 0 and 0.001.

In order to get better results in less training time, pre-trained models have been used. However, no layers have been freezed for this transfer learning technique: the data these models have been trained on is still too different from ours, and freezing the first layers would have brought us to a lower score.

3.2.1 ResNet18

ResNet18, trained with the parameters specified before performed:

- $A_{sample-wise} = 86\%$
- $A_{class-wise} = 85.4\%$

3.2.2 ResNet50

ResNet50, trained with the parameters specified before performed:

- $A_{sample-wise} = 87\%$
- $A_{class-wise} = 85.8\%$

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition.

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization.

[3] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts.

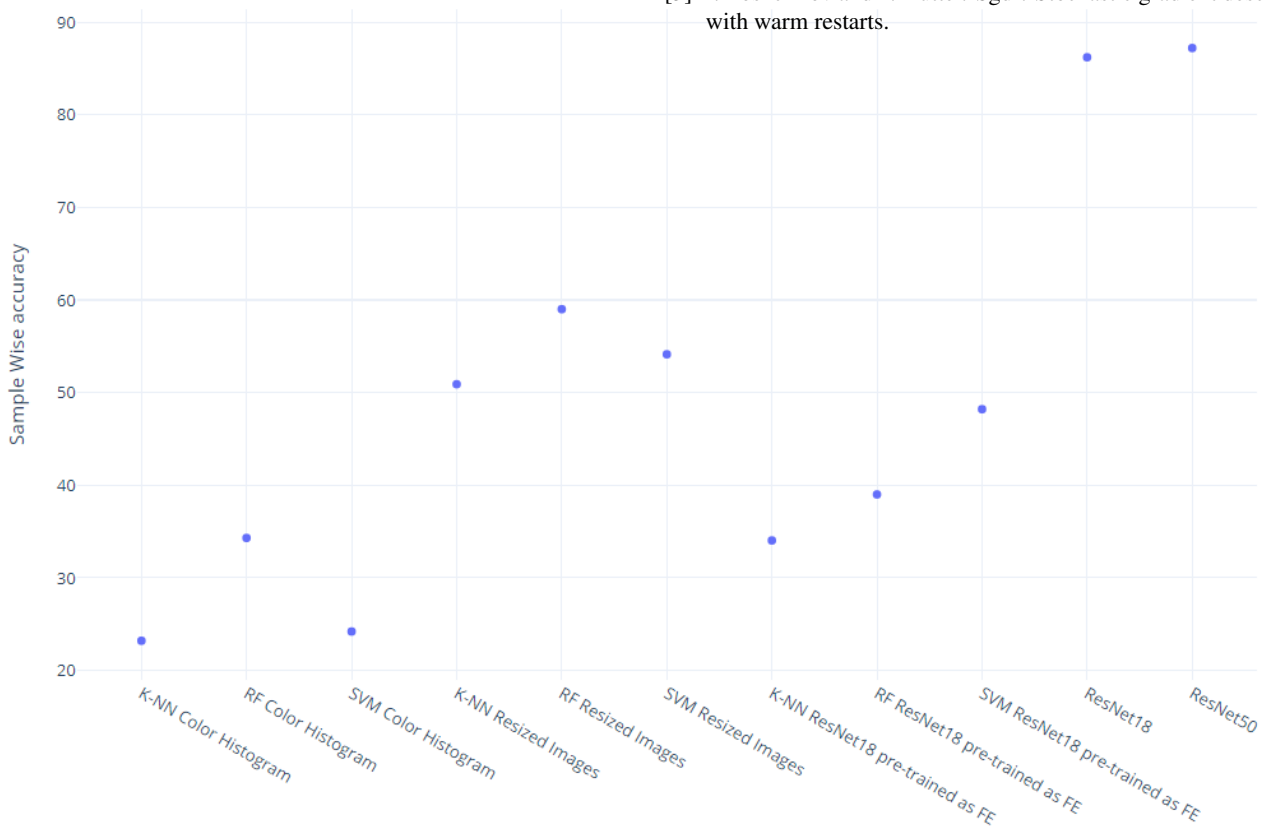

Figure 3. Sample-wise accuracies of different methods.