TRIMBLE TEST TASK

Two-class classifier using Deep Learning algirithms

Produced by
Karim HOCINE

26 mars 2023

1 Introduction

The purpose of this document is to present our methodology of resolving a two-class classifier problem using deep learning algorithms. To do so, we used `Tensorflow`.

2 Methodology

There is two major aspects to consider in order to solve an image classification problem using a deep learning algorithm : The first one is the data we are using and its quality, and the second aspect is the appropriate neural network architecture with enough representation power that can be used to form a good model to resolve our problem of classification with enough precision.
The methodology and process we used to resolve this problem is not much different from any other way of resolving a classification problem using neural networks of any other Machine Learning algorithm. This methodology consists of three steps :

1. Analyzing and processing the Data.
2. Choosing the right neural network architecture for this classification.
3. Fine-tuning our model in order to get the best performances possible.

3 The Dataset

Our task is to implement a Two-class classifier of Road-Fields images. For that, we have at our disposal three folders of images : `"roads"`, `"fields"` and `"test_images"`.
The folder `"roads"` contains **108 images** of roads in `"jpg"` and `"jpeg"` format. The folder `"fields"` contains **43 images** of fields and **2 image** of a road in `"jpg"`. The content of these two folders will be used to build our database for training the algorithm afterward. For the last folder `"test_images"`, it contains **6 images** of roads and **4 images** of field. These images will be used for testing the performance of our model.

So, we have 116 images for the first class `road` and 47 images for the second class field. In total we have a training dataset of **163 images**, and a validation dataset of **10 images** which represents a ratio of **94%/6%** for the train/test datasets. We can conclude that we are facing a classification problem with a small dataset which is unbalanced.

The major problem with unbalanced datasets is that they can lead your model to overfit, which can result in a biased model that poorly generalizes the predictions on different databases. The fact that we have few data too.
To balance the data we can : **Subsample** the majority class, **oversample** the minority one, or use a **weighted** loss function during the training. We will try every solution during training to try to get the best performance possible.

4 The Model

In Deep Learning, the question of which architecture corresponds to which data is really hard to answer. But the scientific community agreed, based on their experiments, that some algorithms and architecture works really well with a certain type of data. Our problem is to classify images, and to do so, the Convolutionnel Neural Networks are the most effective architectures to achieve this task.

There is a lot of reliable architectures that proved in the past their capacity of achieving very good performance in image classification. One of them is the Resnet architecture. So, to achieve our task we decided to go with the Resnet50 architecture that we trained with the different solution mentioned before for balancing our data.

5 The results

First, we trained a Resnet50 four times with a Cross-entropy loss function, each time using : the original dataset, the subsampling of the majority class (Road), the oversampling of the minority class (field) or a weighted loss function. The models were trained over 100 epochs, with a batch size of 16

and a decreasing learning rate from 10^{-1} to 10^{-6} controlled by a callback function of `Tensorflow`, `ReduceLROnPlateau()`. For the optimizer, we choose the SGD.
In a second time, we retrained the same configuration using `ImageDataGenerator()` to feed our model more images since our dataset is really limited in term of samples.

The results of the different tests we did using the Resnet50 are represented in the figures (1, 2). In the first figure, we can observe that all the methods used to balance the data performed as training the original dataset. The validation curves shows that all the techniques used performs poorly with a F1-score under 0.5.
When we use the `ImageDataGenerato()` the subsampling and the oversamplings methods perform best during training and validation. Which confirms that Having a balanced dataset helps generalizing relatively better and can help avoiding the overfitting. But the performs are still poor even after feeding our model new data with `ImageDataGenerato()`.
To confirm our observations and be sure that the results are not biased by the use of a certain kind of architecture, we trained two other architectures : a smaller one (Lenet5), and a middle-sized one (DenseNet). And the observations are the same. We concluded that the performance of the model was affected by the lack of training data.

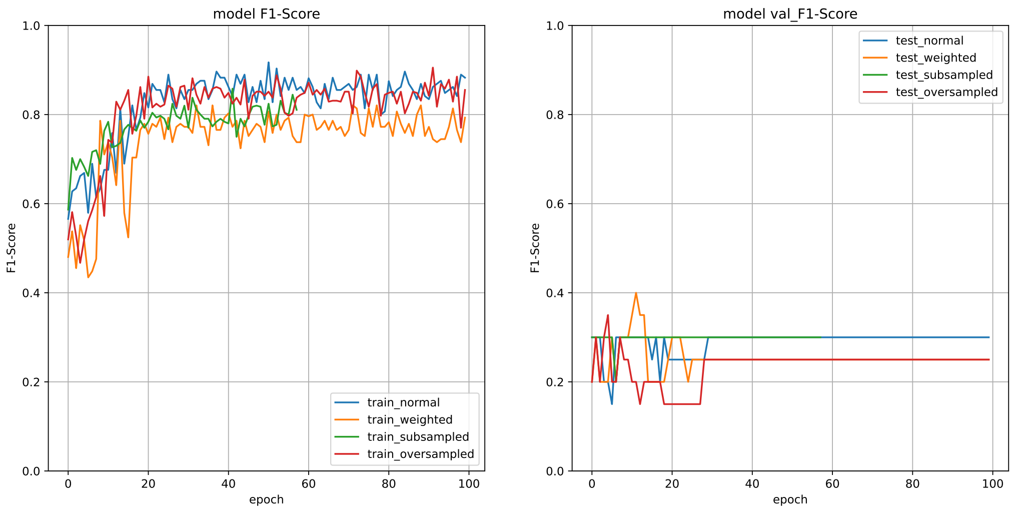

Figure 1 – Evolution of the F1-score during training : Train base (Left), Validation Base (right).

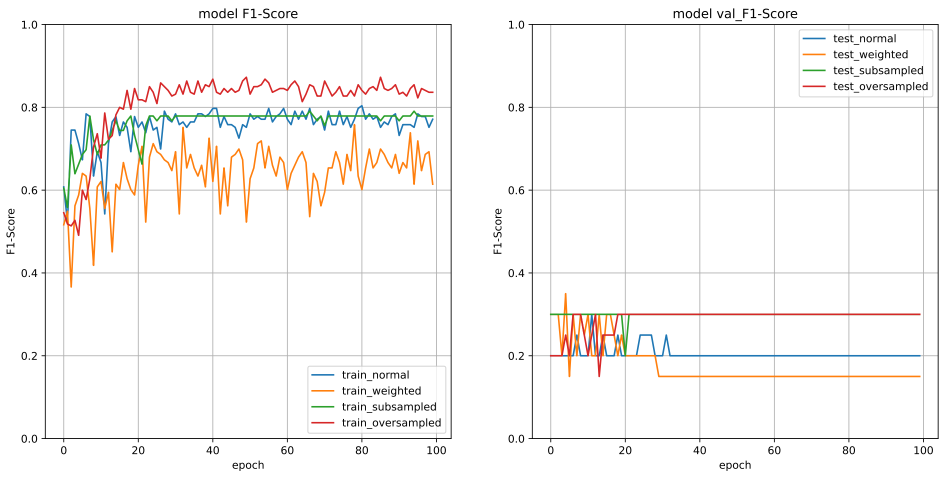

Figure 2 – Evolution of the F1-score during training using ImageDataGenerator : Train base (Left), Validation Base (right).

6 The conclusion

The performance of deep learning algorithms are data dependant. In order to benefit from the full potential of these algorithms a lot of good quality data are needed. Having a small unbalanced training base limited the representational power of our model which led to the appearance of training biases leading to poorer performance. There are two solutions to solve this problem. Enlarge our database or change our classification algorithm.

To enlarge the dataset we can either collect more data, which is the best solution, But sometimes not possible because of the non-availability of the latter or their non-accessibility. To avoid this problematic we can use GANs (generative adversarial networks), which are neural networks architecture that can help us generate synthetic data from an existing database. This is good alternative but time and computational consuming.
To avoid some of the problems related to the data, we recommand using simpler machine learning algorithms (KNN, SVM, Random forests, ...) since we only have a two class classifier to implement and few data to work on.

Github Link : `https://github.com/hocinekarim18/Roads-fields-classification/tree/main`