ML challenge : Strong gravitational lensing

Boris Bergsma [1], Nicolas Frank [2], Alexandre Di Piazza [3]

EPFL

Abstract

With the new generation of terrestrial and spatial telescope on the horizon, fast data analysis is a growing concern. Machine Learning has improved to offer powerful tools such as Neural Networks to perform classification tasks, that become relevant in the automatisation of such data analysis. This paper will cover the particular task of analysing massive data for strong gravitational lenses using a Convolutional Neural Network. This information can then be used to determine the amount of Dark matter present in the objects at the core of the gravitational lens, generating some relevant data for modern day research.

[1] *Second year Physics Master, EPFL, Lausanne, Switzerland ,* ***boris.bergsma@epfl.ch***
[2] *First year Physics Master, EPFL, Lausanne, Switzerland ,* ***nicolas.frank@epfl.ch***
[3] *First year of Data Science Master, EPFL, Lausanne, Switzerland ,* ***alexandre.dipiazza@epfl.ch***

Contents

1. Introduction

Modern research focuses nowadays on phenomenons that can happen only with low probability. To observe such phenomenon, it is required to create or observe the underlying process an enormous amount of time, which means to collect a constantly increasing amount of data. Analysing these data becomes a major concern, it is not possible to store every acquired data hence why it is required to keep only the relevant ones, while throwing away the rest. In 2022, the ESA will launch its EUCLID telescope, which will observe a large part of the sky in order to find gravitational lenses [1]. A large quantity of data will be taken everyday, and it is required to be able to automatise their treatment.

The goal of this project is to build a Convolutional Neural Network (CNN) able to classify pictures of galaxies as "containing lenses" or "not containing lenses" in the most accurate way possible, with a dataset provided by the Bologna Lens Factory, Gravitational Lens Finding challenge [2]. Some Machine Learning processes such as data augmentation and data transformation will be applied to improve the accuracy of our Neural Network and different architectures will be tested. In a matter of comparison, non-neural network related techniques will also be tested to perform the classification

2. Theory

2.1 Strong Gravitational Lensing

Einstein's General Relativity has proven that light gets deflected by mass, this effect allows us to see distant faint objects. Indeed, if a dense mass distribution lays between us and this distant objects, its light will get deflected by the mass distribution toward us, magnifying, but also multiplying and distorting the original image. Observing a gravitational lens gives us an idea of the total mass that deflected the light, i.e. the mass of the galaxy/cluster of galaxies that was on light's way, which allows us to study the dark matter in the latter. The EUCLID telescope tends to picture many such gravitational lenses, the objective of this project is to classify the images as containing or not a lens using Neural Networks.

2.2 Data

For this project, the data set consists of 100 000 images given by the Bologna Lens Factory, Gravitational Lens Finding challenge [2], in the visible spectrum and in three bands of the infrared spectrum (H, J and Y) for a total of 400 000 images. Each image is either containing a lens, labelled "1" or not containing a lens, labelled "0". Figure 1 shows the example of two extreme cases, in the visible spectrum :

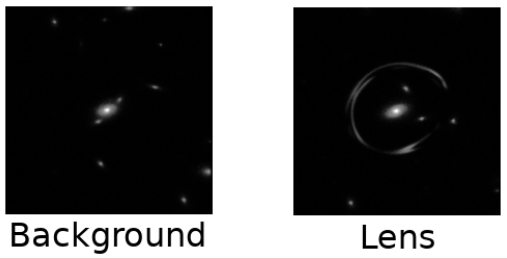

Figure 1. Comparison of the two possible cases in the training set in the visible spectrum. The images are stretched manually to help unveil fainter details.

The resolution of the visible and infrared images is not the same, the visible images contain three times more pixels, hence why the infrared images need to be interpolated in order to be combined to the visible ones. Each pixel in the images has a very low value, of the order of 10^{-13} which forces the weights and the filters of our network to be really big, a good way to improve the learning efficiency is then to normalise our data set before feeding it to the NN. Another possibility was to apply a logarithmic stretch to the data, to ensure the network would find the features of the set with more ease. Figure 2 shows the difference between the different steps of our pre-processing pipeline :

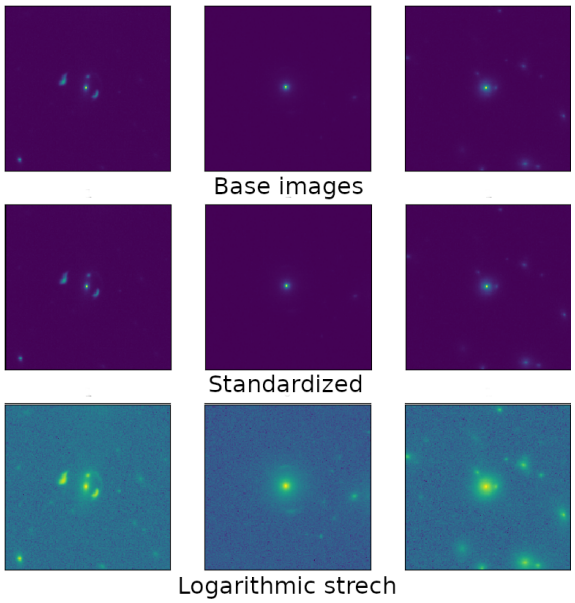

Figure 2. Demonstration of the pre-processing pipeline. While the first step seems trivial, it is really important to bring very low values (in the 10^{-13} range) to a more usable one. The logarithmic scale was brought in to improve the visibility of dim features. Indeed the lens on the first image appears brighter after applying a logstretch transformation.

One of the main problem of the data set is that it contains 90% of lenses and 10% of non lenses, a good way to learn is to have about the same amount of ones and zeros. The simplest way to get rid of the problem was to simply load equal amounts of both labels. Other more advanced techniques could have been used, but the large data set given for the challenge was large enough to have a meaningfull training set. Alongside the data was given a catalog referencing the informations about the images that were given. One of the most relevant information for machine learning was the size of the lenses. This is where the complexity of the predictions rises, there is a large quantity of very small lenses in the data set. Figure 3 shows the histogram of the lenses present in the data set.

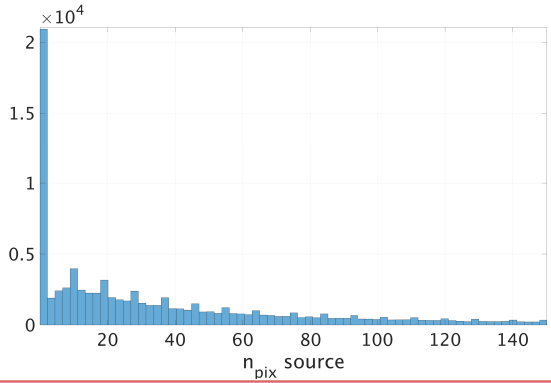

Figure 3. Distribution of the size of the lenses in the training set. The large peak at zero corresponds to the background images. The difficulty of this data sets lies in the small objects to detect. If a lens represents 20 pixels of the image, it corresponds to 0.05 % of the total pixels.

Figure 3 highlights the fact that the size of the features to detect in the data set are most of the time really small. The smallest lenses have a number of pixels equal to 2, and are not a minority. The obvious lenses that where shown in Figure 1 are rare, about 8 % and have a huge number of pixel lensed, larger than 300. This information was very relevant for the rest of the training. It was found that when lenses are below 70 pixels, it is almost impossible to detect with the human eye. Adding the noise on the image, the training was often stuck at low accuracy, being unable to detect the smallest lenses. To give a comparison, Figure 4 show lenses with decreasing pixel size :

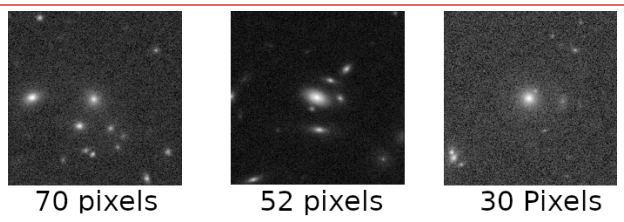

Figure 4. Demonstration of really hard data to detect, with the number of pixels lensed. A lot of companion galaxies are present, and at a low pixel size, the lenses start to blend with the background noise.

Figure 4, and the fact that the final challenge is graded with the f1 score, lead to an approach starting with the detection of large lenses. This has also a scientific reason behind it, false positive are unwanted in these kind of detection, while false negative are not so problematic. Indeed, if we do not detect a certain lens, it was probably small, and therefore less interesting for research.

3. Old techniques

The first thing that comes to mind in our recent times when talking about image recognition are CNNs. However, before the popularity increase of NNs, other machine learning technique where used. As seen in Figure 3, the complexity of the data set will probably not be favorable to these old techniques, but to have a complete overview of the problem they were nevertheless tested.

3.1 Kernel Logistic Regression / SVM

We tried to implement Kernel logistic regression and SVM (Support Vector Machine). The idea is to use the kernel trick to make the data linearly separable and apply the classifiers. The key was to find the right transformation to make the data linearly separable. We therefore tested different kernels, such as the polynomial Kernel, the RBF Kernel and the hyperbolic tangent Kernel. The best results were obtained using an RBF Kernel by Monte Carlo approximation from the library Scikit-Learn. However, the results were far from the results obtained by the neural networks. Even if we tried to optimise the parameters such as gamma and the number of dimensions of the RBF approximation, we would never reach the performance of CNN.

3.2 KPCA

We also tried to use Kernel PCA (Principal Component Analysis). It consists of increasing the dimensionality and then finding a lower sub-space to reduce the number of features (using principal directions with maximum variance). We again tried different kernels and different numbers of principal components to reduce dimensionality. Again, the best results were achieved using and RBF Kernel. However, the dimensionality reduction failed to have an impact. Indeed, it seemed that only the first eigenvalue carried significant information. Therefore, it seems to fail in finding clear characteristics to describe the data (properties that strongly differ between images that are gravitational lenses, and those which are not).

3.3 Main problems encountered and results

The largest problem encountered in the methods discussed above was to find the right Kernel function with the appropriate parameters in order to make the data linearly separable. Being not a priority in the project, the programs were only run on a laptop with 16Gb of ram and an intel i7-6700HQ processor, restraining even further the succes rate. To sum up these methods, Table 1 shows the results obtained.

Methods	Kernel Logistic Regression/SVM	KPCA
Accuracy	< 50 %	< 50 %

Table 1. Results obtained with kPCA and SVM on 1000 images (balanced).

4. Convolutional Neural Nets

4.1 First small CNN and preprocessing choices

As an introduction to the method, and to prevent the use of complex models if possible, a very simple convolutional Net was defined. It is composed of 5 Convolutional layers, 3 MaxPooling Layers, and 2 Dense layers. Dropout were added to prevent overfitting, and a BatchNormalisation was added to improve convergence. The complete scheme can be seen in Figure 7 in the Appendix. The goal here was to train the network on large lenses with more than 150 pixels, to then experiment with it. This network lead to some impressive results, with a small training session already leading to results above 80 % with a moderate amount of training (37 epochs and 5000 images as train set). Another advantage of a rather simple network (with 800 thousand parameters), is that it was very handy to check which Data set to feed the networks (there are 4 channels), and to get a better idea of what the final preprocessing should be. First, a Data set had to be chosen. The main choice was between the IR and Visible Data sets. The visible image set is mandatory, because of its higher resolution and quality. The IR data, even if it is of a mediocre quality, is still relevant for a physical reason : the universe is in constant expansion, and the lensed object are often far away. This means that their wavelength gets redshifted (according to Hubble's law) proportional to their distance. If the shift in wavelength is too extreme, the object might not appear in the Visible spectrum. The idea was to at least include the largest wavelength band (IR_ H) to improve the sensibility of far objects. The IR bands images are of size (66,66) so they had to be upscaled to the size of the visible images (of size (200,200)). This meant that some artefact appeared with the interpolation so the final idea was to give the network IR images composed of $0.15 \cdot IR_Y + 0.35 \cdot IR_J + 0.5 \cdot IR_H$, to help smooth out the problems of the upscaling, and keeping a bias towards to farthest IR band. The other decision that had to be taken was on the preprocessing pipeline. The standardisation was almost certainly a good choice to avoid the extreme values in the weights of the network and to help have a uniform dataset. The logarithmic strech is on one hand rather interesting to help increase the features in the image, but also comes with a lot of added noise. The IR band integration also come with a drawback, they add a lot to the computation time, the tensors are twice as big. This was all tested by training the small CNN with all the

different preprocessing possible, Table 2 shows the improvement made by each of them.

Method added	Standardization	Log scale	IR bands
Improvement	3 %	3.08 %	4 %

Table 2. Improvement (in accuracy) made with the neural network (presented on Figure 7) for different preprocessing methods (compared to base result). The lenses had more than 150 pixels lensed, trained on 10000 images.

As expected, for the reasons cited above, the standardisation and the IR bands give a significant improvement on the final result. The only debate lies in the logarithmic scale step. In fact, applying the logaritmic stretch and the standardization only improved the total result by 0.08% compared to just applying one of them. Finally combining all the possible improvement, this simple neural net reached a precision of 88 % on very large lenses with a number of pixels lensed above 150.

4.2 Transfer learning on EfficientNet

Transfer learning is an interesting subject in deep learning, and helps exactly with the problems encountered when learning massive networks. It consists in taking a shortcut in the training phase, by taking a network that was already trained. Then, the goal is to adapt the network to the required problem. The source followed as initiation was [3]. This technique is possible because if the CNN is trained to detect a variety of different objects, the shapes detected in the convolutional base are "arbitrary" and can be adapted to any new objects. The classifier has to be trained again to then make prediction in our dataset. In our case, the known and recent EfficientNet [4] was used, to its capability and relative simplicity. To adapt for the efficient net 3 channels input, the network was fed : (Visible data/mean(IR_Y, IR_J)/IR_H). The network defined to compete with the others is presented in Figure 8. The number of pixels in the lenses was set to 150, trained on a laptop, giving the following learning curve on Figure 5 :

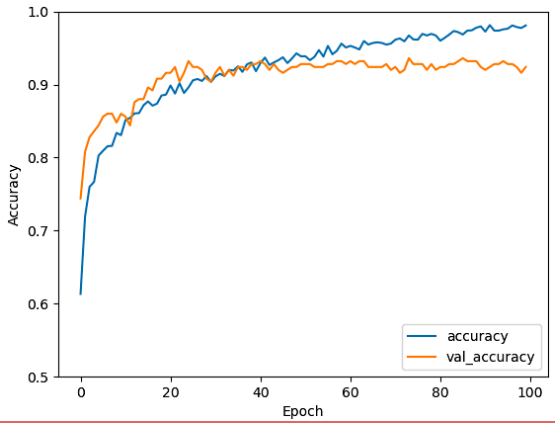

Figure 5. Learning curve of the EfficientNet CNN, on large lenses (> 150 pixels) on a laptop (gtx 970m).

Figure 5 highlights the quick results on the learning and the efficiency of the network to detect large lenses. A run was then finished to detect large sized (> 150 pixels) lenses, and the final result is given in Figure 6 :

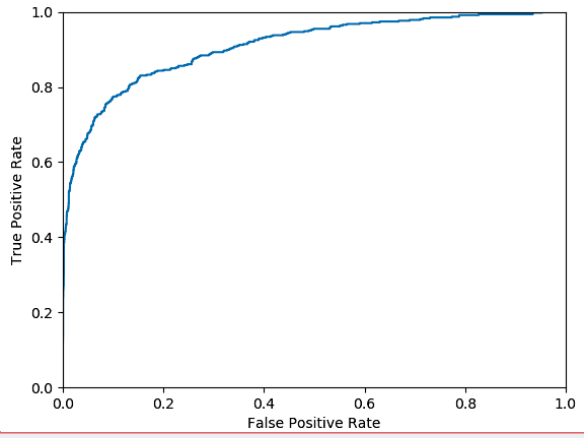

Figure 6. ROC AUC curve of the EfficientNet, for lenses above 150 pixels. The final AUC score is 0.90899.

5. Conclusion

All in all, this project proved to be a perfect training for machine learning methods and their shortcomings. To sum up the most representative results are presented in Table 3.

Methods	Old	Small Cnn	EffNet
Accuracy	< 50 %	88 %	91.733 %

Table 3. Final results obtained with the most relevant methods tested. Old stands for old methods, and was estimated on a 1000 sample data set. As for the two neural nets, the test was conducted on a 10000 image with lenses above 150 pixels set, that was never trained on to keep it as a fresh test.

Table 3 highlights the difficulty for techniques not involving Neural Nets to compete. Then with a score of 88 % , a relatively simple neural network was able to achieve descent results, for an acceptable computation time. When trying to improve the predictions, the resources needed to train the models were greatly increased. A simple laptop was often stuck, due to the low batch size and slow performance. The pre-trained EfficientNet achieved good results on large lenses, with a final result of 91.733 %. This project is still in progress, what will be done in the future, (to compete for the online challenge) is :

- Decrease the size of detected lenses.
- Train on the whole dataset, once the test set is realeased.
- Get a more complicated classifier on the top of the Efficient Net architecture.
- Get a more advanced version of Efficient Net with more parameters as the convolutions base.

A. Appendix

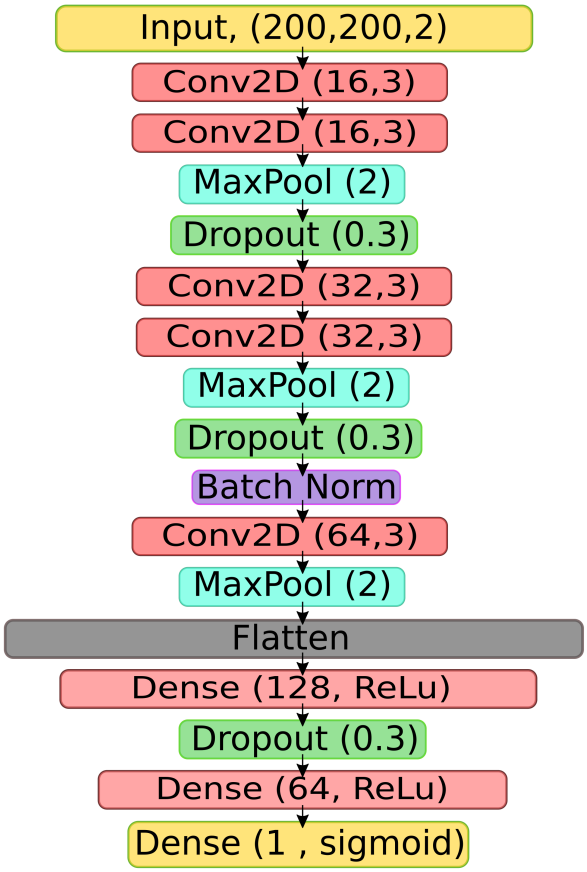

Figure 7. Small Cnn described in section 4.1. The goal was to have a small contender to see if the problem really required a larger number of parameter and Convolutional layer to get good results. It was also used to assess the efficiency of the different preprocessing methods used.

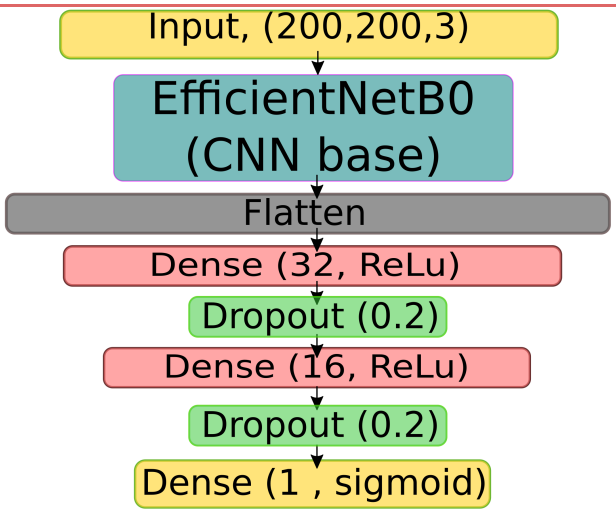

Figure 8. Architecture used for transfer learning. It is simply composed of the convolutional base of the EfficienNetB0 [4], with a simple defined classifier on top. The fact that the Convolutional base was already pre-trained, means that there can be a big advantage in computation time, if the convolutional layers are trained to look for general things.

For the neural nets, the parameters of each layers is given here. The parameters on the Conv2D layers correspond to (filter number, filter size). The parameters on the MaxPool layers are the area of sampling, with always the axis taken as equal. Here a MaxPool of 2 means we reduce the size of the image by two, sampling it on a 2*2 kernel. The parameter of the dropout layers are simply the dropout rate. Finally the Dense layers have as parameter (number of neurons , Activation function).

References

[1] EUCLID mission. https://sci.esa.int/web/euclid/

[2] Bologna Lens Factory, Gravitational Lens Finding challenge. http://metcalf1.difa.unibo.it/blf-portal/gg_challenge.html

[3] Transfer Leraning source https://www.dlology.com/blog/transfer-learning-with-efficientnet/

[4] EfficientNet for Tensorflow / Keras https://github.com/Tony607/efficientnet_keras_transfer_learning