Machine Learning: Gravitational Lens Finding

Y. Saied, L. Syvis, V. Crettenand
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—The Euclid telescope, due for launch in 2021, will perform an imaging and slit-less spectroscopy survey over half the sky. In such imaging surveys finding strong gravitational lenses by human inspection is difficult. For this reason a Convolutional Neural Network (CNN) was developed that can be used to identify images containing strong gravitational lenses. CNNs have already been used for image and digit classification as well as being used in astronomy for star-galaxy classification. In this paper, the CNN is trained and tested on Euclid-like observations in the VIS, and NISP J, Y and H bands. Obatining an area under the ROC curve of up to 0.94.**

I. Introduction

Gravitational lens is a phenomenon that occurs when a massive object noticeably alters the direction of light passing nearby[1]. The effect is analogous to that produced by a lens. A photon's geodesic is affected by massive objects' gravitational pull, and consequently light passing a distant galaxy will be noticeably deflected as a result. Since the first find in 1979, numerous such gravitational lenses have been discovered. In addition to providing tests of Einstein's theory of general relativity, gravitational lenses have proved to be valuable tools. Notably, one can determine the mass distribution of the object that is bending the light — including the mass of the still-enigmatic dark matter, which does not emit or absorb light and can only be detected via its gravitational effects. The lens also magnifies the light source, acting as a "natural telescope" that allows astronomers a more detailed look at distant galaxies than is normally possible.

So far, less than a thousand lenses have been found in total across many heterogeneous data sets. Projects like *The Square Kilometer Array*[2], *the Large Synoptic Survey Telescope*[3] and the *Euclid space telescope*[4] are expected to increase the number of potential lenses by orders of magnitude. For example, it is estimated that there will be approximately 200 000 observable galaxy-galaxy lenses in the Euclid data set among tens of billions of potential objects [1][2]. These surveys will bring a new era for strong lensing where large relatively well defined samples of lenses will be possible. It will also require handling much larger quantities of data than has been customary in this field. The aim of this paper is to make use of CNN to identify strong gravitational lenses in an Euclid-like observation data-set.

[1]https://www.britannica.com/science/gravitational-lens
[2]http://skatelescope.org
[3]https://www.lsst.org
[4]https://www.euclid-ec.org

II. Dataset

The data used in this paper comes from the "Gravitational Lens Finding Challenge" organized by *Bologna Lens Factory*[5]. The dataset was designed to mimic ground based data with multiple bands, roughly modeled on the Kilo-Degree Survey (KiDS) [6]. The simulated images were not meant to precisely mock these surveys, but the surveys were used as guides to set noise levels, pixel sizes, sensitivities, and other parameters. These images were Euclid-like observations in the VIS, and NISP J, Y and H bands. The pixels sizes were 0.1" for VIS and 0.3" for J, Y and H. The training and the test sets consisted of 100,000 images in each band. Each data point contains a gray scale 200x200 pixel image in the visible spectrum as well as three 60x60 images in other spectrum. Each set of four pictures has an associated label specifying if the image contains a gravitational lens or not. The goal of this paper will be to predict this label.

III. Models and Methods

The following section explains what deep learning techniques were used and why they were used.

A. Transfer Learning

Transfer Learning is a Machine Learning technique in which a model originally trained and developed for a given task is used for related task. It refers to the situation where what has been learnt in one setting is exploited to improve optimisation in another setting [7]. [10] proposes a system which uses a model named "efficient net" which was developed to classify images from ImageNet. The model architecture, as well as the pre-trained weights, were used as a starting point for this paper. There are two usual ways of doing transfer learning. One can either modify only the last few layers in which case it is not necessary to retrain the weights of the layers that were imported or freely modify any layer in which case it is necessary to retrain all layers and weights. As the dimensions of the data, number of channels, that was used was different from the input dimensions of the efficient net model, the second option was used. The initial layer was modified to fit the dimensions of our data and the last layer: the classifier layer, was modified to make binary class predictions hot encoded in one output.

[5]http://metcalf1.difa.unibo.it/blf-portal/gg_challenge.html

B. Data augmentation

In order to improve the convolutional neural network (CNN), the data augmentation technique was used. This technique essentially helps to overcome the problem of limited quantity and limited diversity of testing data by generating additional data from the existing data. Having more data-points makes it harder for the model to over-fit the data. Without data augmentation the model might in part memorize the images instead of learning the characteristic features of gravitational lenses. Data augmentation helps reduce this effect.
In this project it was decided to augment the training set by using the square's symmetry group ($\mathbf{D}_4$) elements. Namely, the four mirroring symmetries and the four rotating symmetries of a square. The possibility of doing data augmentation by adding noise to the images was not investigated.

C. Adaptive learning rate and automatic stopping

Adaptive learning rate was used to dynamically change the learning rate during the learning phase. The principle of adaptive learning rate is to use a large learning rate at the beginning of the learning to approach the minimum of the loss function quicker. As the minimum of the loss function is approached, the learning rate is gradually decreased so as to avoid overshooting the minimum. An automatic stopping condition was implemented which stops the simulation if the difference between the maximum reached AUROC (area under the ROC curve) and the current epoch's AUROC passes a certain threshold. The reason for implementing this is that it was noticed that the loss sometime worsens significantly during training and doesn't come back to its best value for several epochs. When this happens, the weights that achieve the lowest value of the loss function are lost. The fact that the loss function sometimes worsens is due to the random component in the descent path brought about by stochastic gradient descent. The stopping condition allows to stop training in a favorable state.

D. Balanced batch sampler

A batch sampler that balances the number of data-points containing gravitational lenses with those not contain gravitational lenses was used. The reason for using such a batch sampler is that about 90% of the data-points contain gravitational lenses. This means the model could have learned to always predict gravitational lenses and reached an accuracy of 90% in only a few learning steps. The balanced batch sampler was used to remove this biased tendency of the model to always predicting gravitational lenses.

E. Regularisation

Regularisation was used to reduce the chance that the model would over-fit the data. The chosen regularisation was L2 norm. The regularisation coefficient used was 1e-4.

F. Accuracy criteria

In order to evaluate the results, traditional criteria were chosen. The most known method for evaluating a classification algorithm is using the receiver operating characteristic (ROC) curve. It is the plot of the True Positive Rate (TPR) against the False Positive Rate (FPR), where TPR and FPR are defined as such:

$$\mathrm{TPR} = \frac{\text{number of true lenses classified as lenses}}{\text{number of true lenses}} \tag{1}$$

$$\mathrm{FPR} = \frac{\text{number of non-lenses classified as lenses}}{\text{number of non-lenses}} \tag{2}$$

If the classifier made random guesses, then the ratio of lenses to non-lenses would be the same as the ratio of the number of cases classified as lens to the number of cases classified as non-lenses and so TPR = FPR. So, the plot with FPR on x axis and TPR on y axis would be a diagonal line from the origin to the top right corner. The better a classifier is, the smaller the FPR and the larger the TPR. Hence, the further away ROC curve gets from the diagonal line and the closer it is to the top left corner.

Since gravitational lenses are rare events, small false positive rates are desired, hence the ROC curve was plotted with the FPR in log scale.

A common figure of merit for a classifier is the area under the ROC curve (AUROC). This evaluates the overall ability of a classifier to distinguish between cases. It is also a scalar, so it can be simply compared between different algorithms.

IV. Results and Discussion

To make the comparison of the different methods of training the model easier, the data was split in a training (80 percent of dataset) and a testing set (20 percent of dataset) which were used for all of the following results. The best model according to AUROC was model 8 (as shown in Table 1). [add why]

Before starting this project we had a concern that current methods would be too slow or require too much human intervention to handle large data sets.However, the CNN codes take some time to train, but once trained they are very fast in classifying objects.

A. *Results*

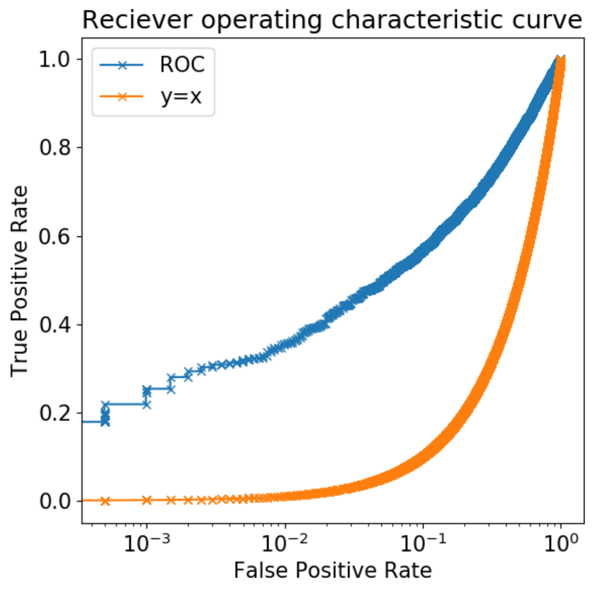

Figure 1. ROC curve for model 7

B. *Training*

Two methods of training worked well. **Method 1** is to reinitialize the batch normalization layer and retrain all the model's weights. Re-initializing the batch normalization layers deletes the mean and standard deviation memorized from training the model on efficient net. This makes sense because our data should be normalized differently from the images of efficient net since they are of a different type (space pictures some of which are not even in the visible spectrum vs every day life pictures).

Method 2 is not to reinitialize the batch normalization layers but keep the model in training mode when testing it. In this method too the weights of the model are retrain. It was observed that the model's predictive power on the testing data was poor when the model was run in testing mode. It was tried to use the model to predict the classes of the testing data while keeping the model in training mode and this yielded much better results.

Let's look at what difference it makes to run the model in testing or training mode. The only difference between the two modes has to do with the batch normalisation layers which normalize there input using the mean and the standard deviation. For practical reasons, the mean and standard deviation are computed over the current mini-batch in training mode. In testing mode the model should be able to provide predictions even when there is only one input data point. This poses a problem because standard deviation of a single data point is zero and in the process of normalizing the data, every feature is divided by their standard deviation. This problem is usually solved by using the mean and the variance of the features of the whole population of data points available in testing. The batch normalization layers records the mean and the standard deviation of every batch during training and calculate a global mean and standard deviation.

Now that the difference between training and testing mode is understood let's discuss why the results were better when making predictions in training mode with batches of 8 data points. A presumed reason for having better results when testing in training mode is that the mean and the standard deviation learned by the batch norm layers are not robust. Another possible explanation is that the mean and/or variance of the feature itself have such a large variance that for some outlying batches cannot be appropriately normalized by the mean and variance of the whole population. The second method yields an overall better predictive power than the first method because the AUROC is larger. Unfortunately it has the disadvantage that predictions can only be made with batches of size strictly greater than one. Some ideas of how to further investigate why the model yielded good results when the model was kept in training mode when testing would be:

1) Compare the behaviour of the second method to the behaviour of the first method when keeping its model in training mode too. It might be the case that better predictions can always be made when the testing data is bundled in mini-batches and the model is kept in training mode.
2) Plot the AUROC achieved with the second method as a function of the testing batch size. Since the batch normalisation layer parameters (mean and variance) are calculated only on the batch, looking at such a plot would allow to verify what the effect of batch normalization is with means and variances calculated on samples of different sizes.

The effectiveness of batch normalization though clearly experienced in practice is not well understood. It is debated in the deep learning community whether it improves results because it reduces the internal covariate shift [8] or because it smooths the loss function [9]. Because of the lack of understanding of why batch normalization improves the results at all, it is difficult to further discuss why it seems to perform poorly when the mean and standard deviation statistics are on some

1) Attempt to train the model with only 3 channels: An attempt was made to do transfer learning by replacing only the classifier layer of efficient net. In order to do this, only the 3 non-visible spectrum channels of the data were used. The visible channel was simply ignored. Only the weights of the last layer (the replaced layer) was trained. This method worked but it was dropped because it didn't yield as good results as the other method, with AUROC converging to

Model	Data augmentation	Batch sampling	Batch size	Number of epochs	coeff of regularisation	Highest AUROC	Method #
1			128	10		0.7	1
2		✓	128	10		0.7	1
3		✓	8	5		0.74	1
4		✓	4	5		0.74	1
5		✓	128	4	0.0001	0.64	1
6			8	1		0.72	1
7	✓	✓	8	10	0.0001	0.83	1
8	✓	✓	8	10		0.94	2

Table I
THE ACCURACY OBTAINED USING THE 6 DIFFERENT METHODS

0.71. This result however unimpressive compared to the best reached AUROC values is worth mentioning as a demonstration of the power of transfer learning. Indeed, an AUROC of 0.71 was reached by a method that requires a lot less computational power.

C. *Instance normalization*

Instance normalisation was tried in the hope to overcome the problems encountered with batch normalisation but it was noted that it works worst than batch normalisation. It yields less than 0.7 AUROC values and was therefor not used.

1) Choosing the batch size and the learning rate: Several factors affect the choice of batch size. Even though the used loss function is a convex function of its parameters, because of the non-linear activation functions used in the network, non-convexity is introduced. Indeed, the loss function is a non-convex function of the weights of the model. This means that a regular gradient descent with a small enough learning rate could get stuck in a local minima which is far from the global minima. Stochastic gradient descent introduced some noise in the descent path since only part of the data is considered to approximate the gradient. This noise enable to get out of local minima. This is therefor a wanted effect. The noise introduced by stochastic gradient descent can controlled by changing the learning rate. This is why the two hyper-parameters should be chosen jointly. Another aspect that plays a role in the choice of the batch size is that in order to use the GPUs efficiently, they have to be loaded to close to 100% of their capacity. The batch number that was found to do this is 128. A strategy would be to always keep the batch size to 128 and adjust the learning rate. This was tried and it was found that the learning rate sometimes had to be increased so much to have sufficient noise that the model didn't train anymore. The approach that was retained was to start with a small batch size of 8 and a large learning rate of 0.01 and gradually increase the batch size to 128 (which is the maximum imposed by the GPU's limited memory) and lower the learning rate as the model was trained. This tuning of parameters is done manually by looking at the AUROC (area under the rock curve). When the AUROC oscillates a around a value for a whole epoch, the batch size and the learning rate are lowered. At the end of the training of the model, typical values are a learning rate of 1e-9 and a batch size of 128.

V. SUMMARY

The aim of this paper was to preform binary classification on a Euler-like data-set. Classification was achieved using transfer learning on a CNN model call efficient net. An AUROC of 0.7 was reached. These results were improved by augmenting the data, using a balanced batch sampler, dynamically changing the batch size and learning rate during the simulation as well as implementing automatic stop and using regularisation. The best result reached using all of these techniques was an AUROC of 0.83. If the constraint to always use the model on bundles of 8 images is acceptable than method 2 can be used in which case the AUROC climbs to 0.94. A follow up to this project would be to use ensemble averaging to further improve the results. This was not done in this project because training a large number of models is too computationally heavy to be done in the time available.

REFERENCES

[1] Oguri, M., Marshall, P. J. 2010. Gravitationally lensed quasars and supernovae in future wide-field optical imaging surveys, *Monthly Notices of the Royal Astronomical Society*, Volume 405, Issue 4, pp. 2579-2593

[2] Collett, T. E. 2015. The Population of Galaxy-Galaxy Strong Lenses in Forthcoming Optical Imaging Surveys, *The Astrophysical Journal*, Volume 811, Issue 1, article id. 20, pp. 10

[3] Jackson, N. 2008. Gravitational lenses and lens candidates identified from the COSMOS field, *Monthly Notices of the Royal Astronomical Society*, Volume 389, Issue 3, pp. 1311-1318

[4] Pawase, R. S., Courbin, F., Faure, C., Kokotanekova, R., Meylan, G. 2014. A 7 deg^2 survey for galaxy-scale gravitational lenses with the HST imaging archive, *Monthly Notices of the Royal Astronomical Society*, Volume 439, Issue 4, pp. 3392-3404

[5] Geach, J. E., More, A., Verma, A., et al. 2015. The Red Radio Ring: a gravitationally lensed hyperluminous infrared radio galaxy at z = 2.553 discovered through the citizen science project SPACE WARPS, *Monthly Notices of the Royal Astronomical Society*, Volume 452, Issue 1, pp. 502-510

[6] de Jong, J. T. A., Verdoes, Kleijn G., A., Kuijken K. H., Valentijn E. A., 2013. The Kilo-Degree Survey, *Experimental Astronomy*, Volume 35, Issue 1-2, pp. 25-44

[7] Gao, Y., Mosalam, K., 2018. Deep Transfer Learning for Image-Based Structural Damage Recognition, *Computer-Aided Civil and Infrastructure Engineering*

[8] Ioffe, Sergey, Szegedy, Christian, 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[9] Santurkar, Shibani, Tsipras, Dimitris, Ilyas, Andrew, Madry, Aleksander, 2018. How Does Batch Normalization Help Optimization?

[10] Mingxing Tan, Quoc V. Le, 2019, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Appendix

The cross entropy loss function was used to train the model. This loss function is related to the negative log likelihood. The last layer of a classification network usually has as many outputs as the number of different classes. Applying an activation function to these outputs ensures they satisfy the properties of probabilities. One can interpret these probabilities as a discrete probability density function of a random variable over the classes. The likelihood at hand is the probability of this random variable being the same as the empirical prediction and is given by:

$$-\sum_i q_i \log p_i \tag{3}$$

where the q_i represent the empirical label and the p_i represent the predicted label. q_i is 1 if the sample is part of the ith class and 0 otherwise. p_i is the predicted probability for the input to be in class i.

In the present project there are two classes which will be hot encoded in the output y with the convention that $y = 1$ if there is a gravitational lens and $y = 0$ if there is no gravitational lens. Using the convention that $y = 1$ corresponds to $\vec{q} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $y = -1$ corresponds to $\vec{q} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ the original form of the negative log likelihood reduces to:

$$-\log p_{\frac{y+1}{2}} = \begin{cases} \log\left(\frac{e^{x_0}+e^{x_1}}{e^{x_0}}\right), & \text{if } y = -1 \\ \log\left(\frac{e^{x_0}+e^{x_1}}{e^{x_1}}\right), & \text{if } y = 1 \end{cases} = \log(1+e^{-x_1 y}) \tag{4}$$

where the definition of the softmax activation function was used. It is chosen that $x_0 = 0$ and $x_1 = x$ to simplify the notation. x is the single hot encoded output before going through the softmax activation function. This is the cross entropy loss function for two classes hot encoded in one output.