Classifying errors from strong lens modelling with neural networks

Hoggett Emma
Supervised by : Pr. Courbin Frédéric, Dr. Peel Austin, PhD. Galan Aymeric
Laboratory of Astrophysics (LASTRO), EPF Lausanne, Switzerland

***Abstract*—Strong gravitational lens modelling is a powerful method to study an object's unknown astrophysical and cosmological properties and enables a quantization of dark matter in the universe. Nevertheless, systematic modelling errors often riddle generated models that need to be spot manually. Due to the significant increase in observed strong lensing systems expected with future telescopes, examining residual maps by eye will become unpractical. This project uses neural networks to classify residual maps as a key component in an automated lens modelling pipeline. The main application of such classifiers is to quickly and reliably obtain guidance for further refinement of a lens model while providing a study of degeneracies between model parameters. Our analysis shows that with Ridge regression algorithm, we reach an AUROC prediction score of 0.950.**

I. Introduction

In 1919, Einstein stated in the theory of general relativity that gravity bends light. This theory was confirmed in 1979 when a telescope observed a strong gravitational lens for the first time due to the emergence of increasingly efficient telescopes. By combining theory with astronomical observations, scientists can gauge dark matter in our universe. As dark matter does not reflect, emit light or absorb, it makes it hard to detect. However, dark matter interacts predominantly with luminous matter thanks to gravity, providing valuable information to deduce its existence. As we are more and more able to produce many strong lensing observations, due to technological advances in spatial observations, identifying errors manually in residual maps no longer seems viable. Additionally, residual maps are inclined to have degeneracies, which occur when two residual maps are similar while having different components' parameters. Errors' origins are thus difficult to spot, and advanced analysis of the residual map is crucial. This project focuses on constructing simulated residual maps' data set and automating the classification efficiently while providing an accurate classification for each residual maps.

We use the lenstronomy package to simulate instrumental and lensing effects, including the generation of imaging data and parametric models for our benchmark[4]. To simplify and speed up the process of data creation, we test the deeplenstronomy wrapper. In our analysis, we consider masses' and sources' errors first. The error percentage between the model and the experimental map is assumed to be in the range between $[0, 5]\%$. We also perform a study on the imbalances of each class as empirically, spurious models are abundant compare to accurate models. The data set then trains on convolutional neural networks, and the resulting predictions train an ensemble method.

II. Theory : Experimental approach

A. Measurements

As we obtain imaging data through charge-coupled devices (CCD), we must consider other parameters such as point spread function (PSF), resolution and noise. Those parameters highly depend on the type of observation executed.

a) Point Spread Function (PSF): The point spread function (PSF) describes the output image of a point source or point object captured by a device. The PSF definition depends on the instruments, the atmosphere and the wavelength of our telescopes. As we study spatial observation, the atmosphere has no impact on the PSF. The wavelength, on the other hand, can be determined. The PSF H acts on the imaging data and the model though convolution, such that:

$$Y = H * (G + FS) \tag{1}$$

Where G correspond to the lens light effect in the centre of the image and the FS term corresponds to the source deformed arcs. F is the mass lens effect that bends the source light distribution, and S is the true light distribution of the source galaxy.

b) Resolution: In astronomical observations, one pixel corresponds to an angle in the sky captured by the camera. This angular resolution is limited by two factors: effective resolution and theoretical resolution. The theoretical resolution δd is an absolute lower bound and depends on the measured wavelength λ and the diameter of the primary mirror D of the telescope. The theoretical resolution has, by definition, no instrumental considerations.

$$\delta d = 1.22\frac{\lambda}{D} \tag{2}$$

On the other hand, the effective resolution corresponds to the actual captured resolution and must be greater than the theoretical resolution. This phenomenon is due to technological limitations that come from CCD camera resolution and other associated factors. We can reduce the effective resolution by taking several images of the same object with different angles and combine them in a single image. Thus, the final resolution is inferior to the effective resolution and greater than the theoretical resolution. Final angular resolution is described in the lenstronomy simulator as the unit of angular measurement and is defined in the PSF.

c) Noise: The noise present in imaging data is mostly Poisson noise and Gaussian noise and comes from the measuring tools. Poisson noise derives from the discrete nature of electric charges and can be reduced by increasing the exposure time. Gaussian noise originates from several phenomenons, such as the sensor's heat. Both are present in our imaging data and our final residual maps. We define our final imaging data as following, where N describe the Gaussian noise and the Poisson noise.

$$\hat{Y} = H * (G + FS) + N \tag{3}$$

B. Residual maps construction

We build residual maps from imaging data $\hat{Y}$, model images Y and noise standard deviation σ, which is linked to the noise N in Equation (3). The final generated residual is thus in unit of noise such that :

$$R = \frac{Y - \hat{Y}}{\sigma} \tag{4}$$

When we show images of the residual map, we set a range of $[-6, 6]$ since we assume that variation of 6σ is significant compared to the noise distribution.

III. Theory: Analytic galaxy models

We depict the strong gravitational lens effect as a combination of lens mass and light effects combined with the source light effect. Lens mass F and source light S effects create deformed arcs FS in simulated images, while lens light G occurs in the centre of the arcs, such that simulated images are $G + FS$. This project focuses our analysis on arcs' errors: lens mass error and source light error. As both components are intertwined in our final simulated image, degeneracies are more likely to happen. Two profiles are selected: power-law elliptical mass density profile for the lens mass profile and Sersic light profile for the source light profile. Those profiles are not exhaustive.

On the one hand, the power-law elliptical mass density profile (PEMD) defines the lens mass profile which combines the dark matter mass and the luminous matter. It contains the following parameters: the radius of the Einstein ring (θ_E), the ellipticity (e_1, e_2), the centre (x, y) and the power-law slope (γ_s).

$$\kappa = \frac{3 - \gamma_s}{2} \left(\frac{\theta_E}{\sqrt{q e_1^2 + e_2^2 / q}} \right)^{\gamma_s - 1} \tag{5}$$

The Einstein radius θ_E depends on the size of the image, as it corresponds to the size of the ring. The ellipticity e_1 and e_2 are characterized by the axis ratio $q = \frac{a}{b}$, where a is the semi-major axis and b the semi-minor axis, and by the angle ϕ of the semi-minor axis.

$$e_1 = \frac{1-q}{1+q} \cos(2\phi), \quad e_2 = \frac{1-q}{1+q} \sin(2\phi) \tag{6}$$

On the other hand, the Sersic profile describes the source profile and satisfies the following relationship :

$$I(R) = I_e \exp\left(-b_n \left[\left(\frac{R}{R_e} \right)^{1/n} - 1 \right] \right) \tag{7}$$

Where I_e is the intensity of the galaxy and b_n is specified by the Sersic exponent n that solve the equation $\Gamma(2n) = 2\gamma(2n, b_n)$, where Γ is the Gamma function and γ is the lower incomplete Gamma function [9]. R_e is the Sersic radius that encloses half of the total light emitted by an object, while the value R is determined with the centre $[x, y]$ and the ellipticity components (e_1, e_2) (6).

Each quantity's distribution is designed to fit an 64×64 image, in Table I [8]. Our goal is to obtain simulated images where a ring of light is visible, as shown in Figure 1.

TABLE I: Model distributions : $\mathcal{N}_{log}$ is a log-normal distribution, $\mathcal{U}$ is a uniform distribution[8].

Component	Distribution
Lens: PEMD	
x-coordinate lens centre	$x_{lens} = 0$
y-coordinate lens centre	$y_{lens} = 0$
Einstein Radius	$\theta_E \sim \mathcal{N}_{log}(\mu : 0.0, \sigma : 0.1)$
Power-law slope	$\gamma_s \sim \mathcal{N}_{log}(\mu : 0.7, \sigma : 0.1)$
Axis ratio	$q \sim \mathcal{U}(0.7, 1)$
Major axis angle	$\phi \sim \mathcal{U}(0, \frac{\pi}{2})$
Source : elliptical Sersic light	
Source magnitude	$m \sim \mathcal{U}(20, 24)$
Half-light radius	$R \sim \mathcal{N}_{log}(\mu : -0.7, \sigma : 0.4)$
Sersic index	$n \sim \mathcal{N}_{log}(\mu : 0.7, \sigma : 0.4)$
x-coordinate src centre	$x_{src} \sim \mathcal{U}(-0.5, 0.5)$
y-coordinate src centre	$y_{src} \sim \mathcal{U}(-0.5, 0.5)$
Axis ratio	$q \sim \mathcal{U}(0.7, 1)$
Major axis angle	$\phi \sim \mathcal{U}(0, \frac{\pi}{2})$

IV. Lenstronomy parametrization

A. Deeplenstronomy

Deeplenstronomy is a lenstronomy wrapper that uses a configuration file instead of raw coding skills with lenstronomy, which ease the data generation.

a) Dataset: The size of the data set is established in this section as long as the output directory. The output directory corresponds to the data saving location. We set the random number generator's seed of the data generation to obtain the same data set each time.

b) Cosmology: Two parameters are essential to define the cosmology with which lenstronomy performs all calculation : the Hubble constant and the matter density parameter. The Hubble constant is a constant that link the speed v at which galaxies move away from each other according to their distance d, such that $v = H_0 d$. Several estimations of that constant are available, such that $H_0 = 73.4 \text{km} \cdot \text{s}^{-1} \cdot \text{Mpc}^{-1}$[5] and $H_0 = 67.4 \text{km} \cdot \text{s}^{-1} \cdot \text{Mpc}^{-1}$[6]. A value between both is then set, such that $H_0 = 70.4 \text{km} \cdot \text{s}^{-1} \cdot \text{Mpc}^{-1}$. The omega matter density parameter is a ratio that quantify the total amount of mass in the universe. It can be determined by observing the galaxies' rotation speed or galaxies' orbit speed in clusters of galaxies. The current measurement for the matter density parameter is $\Omega_m = 0.3$.

c) Image & Survey: We choose a spatial telescope model to perform observations, as they have better image quality than an earth telescope due to the earth's atmosphere. Two main choices are available: Euclid space telescope and Hubble

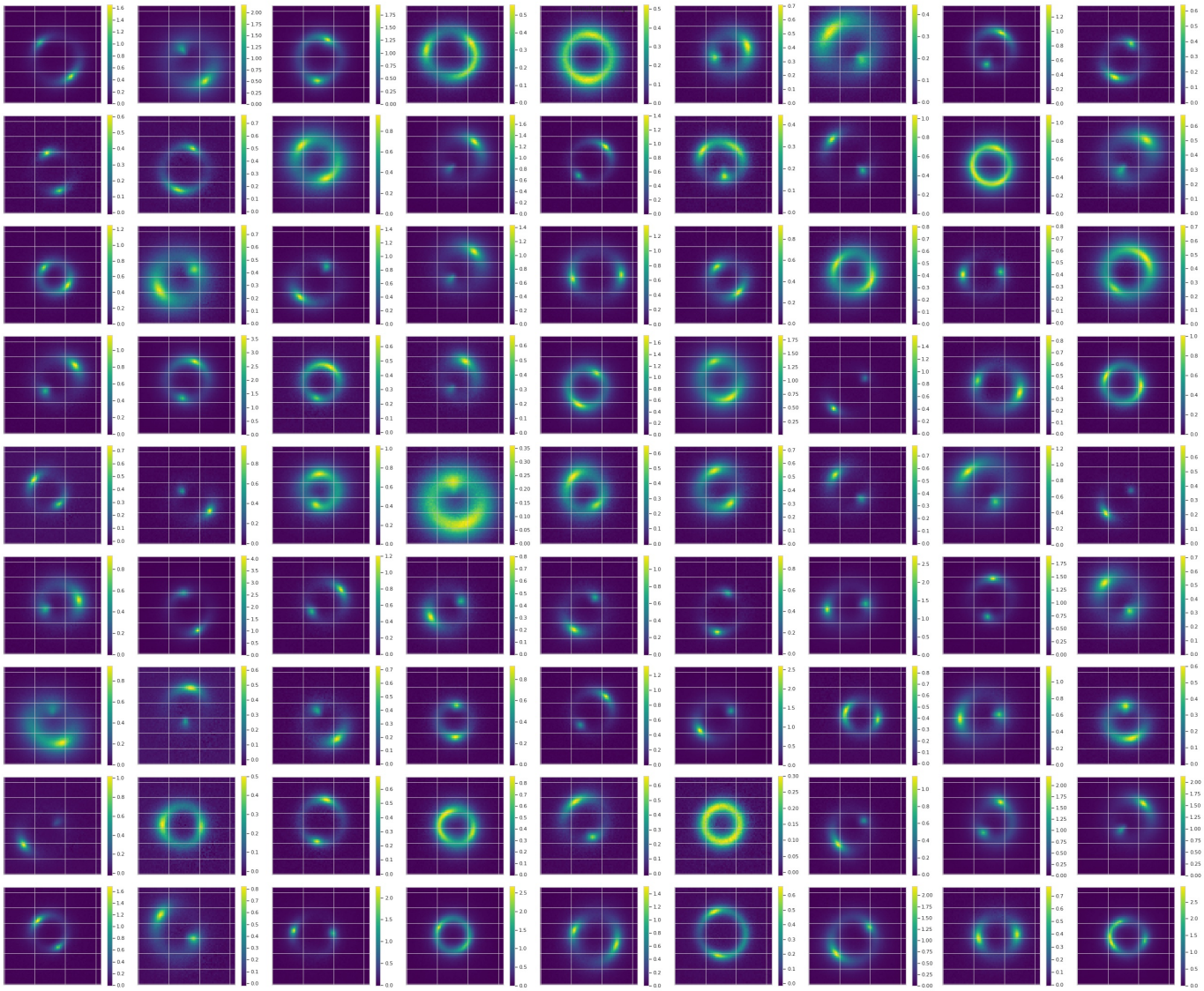


Fig. 1: Random sample of the simulated data set for the described distribution in Table I with lenstronomy

Space Telescope (HST). Due to HST versatility, HST-like observations is chosen for our benchmark [7]. Deeplenstronomy wrapper does not require a point spread function (PSF) setting. The size of images is also set in this section with the `numpix` parameter. We set the seeing to zero in the .yaml file, as deeplenstronomy is unable to handle the `None` command.

TABLE II: Parameters' for HST observation[7]

Exposure time	**Pixel scale**	**PSF type**	**Read noise**	**ccd gain**
5400	0.08	PIXEL	4	2.5
Band	**seeing**	**Magnitude zero point**	**sky brightness**	**exposure**
WFC3_F160	None	25.9463	22	1

d) Species: We can characterize three main model components: galaxies, point sources and noise. Each species can portray parameters via ranges and distributions or via constants. Galaxy objects encompass lenses and sources, with their main configuration parameters such as light profiles and mass profiles. The light profile and the mass profile describe the lens profile. The light profile is constant, while the mass profile depicts the distribution in Table I. The lens' light profile should not impact the residual maps if it is set constant for every simulated image. The light source model obeys the Sersic ellipse profile and has a distribution described in Table I. The only noise implemented in deeplenstronomy is the Poisson noise, which is problematic as imaging data includes Gaussian noise in real life.

e) Geometry: In this section, we specify images composition with the appropriate species. We can establish several configurations with a specific fraction and name. The fraction corresponds to the proportion of the dataset that has this particular configuration. The number should be between $[0, 1]$

and the sum of fractions is equal to one. Each configuration contains a combination of planes in which we must specify a species name with its redshift. Redshift is an increase of wavelength due to the expansion of the universe, which indicates the position of distant objects.

B. Lenstronomy

a) Observations type: Lenstronomy provides a camera and observational settings to set the observation type. In our case, the HST model with Time Delay Lens Modelling Challenge (`TDLMC_F160W_band_obs`) is chosen [7]. This configuration needs a point spread function (PSF) definition separately due to the pixel PSF type. We consider a drizzled point spread function in Rung 0 used in the Time Delay Lens Modelling Challenge [7]. This configuration corresponds to observations obtained with the WFC3 camera in the F160W filter.

b) Model: The model consists of a lens mass profile and a source light profile. With lenstronomy, the lens light profile is optional, so the lens light profile is not set at first. With the distributions described in the Table I, the Figure 1 is obtained with a bent light ring.

V. Data generator

A. Simulated images

In Figures 1 and 22, we remark that rings are more obvious in the lenstronomy parametrization compare to deeplenstronomy simulated images. Nevertheless, both have the same model's configuration, proving that we must select additional parameters such as cosmology, image, survey and redshifts wisely in deeplenstronomy. In Figure 22, a light dot is present in the centre of the picture, which corresponds to the lens light profile that we define in the deeplenstronomy file.

B. Residual maps

Due to different simulated image generation, we expect that deeplenstronomy and lenstronomy provide different image rendering. In deeplentronomy, different models and simulated images are compared without any fixed relative error, while in lenstronomy, we compare models and simulated images with a fixed error percentage. In Figures 2 and 23, we observe that the noise generation is different between deeplentronomy and lenstronomy. Images generated with deeplenstronomy have a smoother noise definition, whereas images generated with lenstronomy contain granularity. This phenomenon is due to Gaussian noise consideration since Gaussian noise is not implemented in the deeplenstronomy wrapper.

As we expected in Section IV-A, the fixed lens light profile does not show in the lenstronomy's residual maps as we subtract equal quantities (Figure 23). The resulting images' contain profiles with obvious errors for the mass error when source errors and source and mass errors have subtle amplitude variation. In the deeplenstronomy configuration, users have difficulty differentiating between source errors and mass and source errors.

In lenstronomy's residuals, the mass error and the source and mass error are easy to spot, although the source errors are often not discernible. With lenstronomy configuration, users have difficulty differentiating between mass errors and mass and source errors.

C. Conclusion

For generating noise on the data set, the wrapper deeplenstronomy is unpractical, as we can observe in Figure 23. The parameters of the images need to be set to produce noise, and adding a configuration of noise does not involve the same parameters as with lenstronomy. Moreover, the configuration is less flexible, and building models with 10% of error is trickier than using the lenstronomy simulator. This simulator can be interesting in the future for a more advanced model, as the difficulty to differentiate between source errors and mass and source errors may improve our final classification.

VI. Building errors

At the end of this process, we obtain residual maps corresponding to the subtraction of the noiseless model and the imaging data containing noise, as in Equation (4). An imaging data and a model with a fixed error percentage are subtracted and normalized to build the false data set. We use the same approach for making the accurate data yet with the same configuration for the imaging data and the model.

A. Error type: relative error

The added error corresponds to relative error of the model $v_t = v_f + \delta|v_f|$ where v_t is a model's parameter and v_f the experiment's parameter. We test several error percentages to quantify an acceptable range of error. A relative error of 5%, as shown in the Figure 2, is easy to spot for the human eye and is unrealistic compare to real model generation. In Figure 3, the error is more insidious, which makes it more suitable as a dataset. We also observe that the error for the mass is easier to spot than the source error, which arises from the range of initial distributions and differences between the source and mass models' definition. Thus, we consider different percentages for each label combination to reach coherent results. As the perception of error is subjective, we must set a metric to give meaningful results. As the source error is more insidious than mass error, we state an error of approximately 1.5% for the source. Then iteratively, we achieve a percentage of the mass error and the mass and source error at a portion of 0.5% in Figure 4.

B. Residuals error metric

Relative errors are, however, not sufficient to define the resulting residual maps. Generated maps often show obvious errors or errors that are too similar to noise (Figure 4 and Figure 5). Data distributions between quantities are different, making the emergence of differences in the error generation, as it is a relative error. A solution to this problem is to implement an image metric to give a more homogeneous data set. Firstly, images r containing at least one pixel $r(m, n)$ with absolute

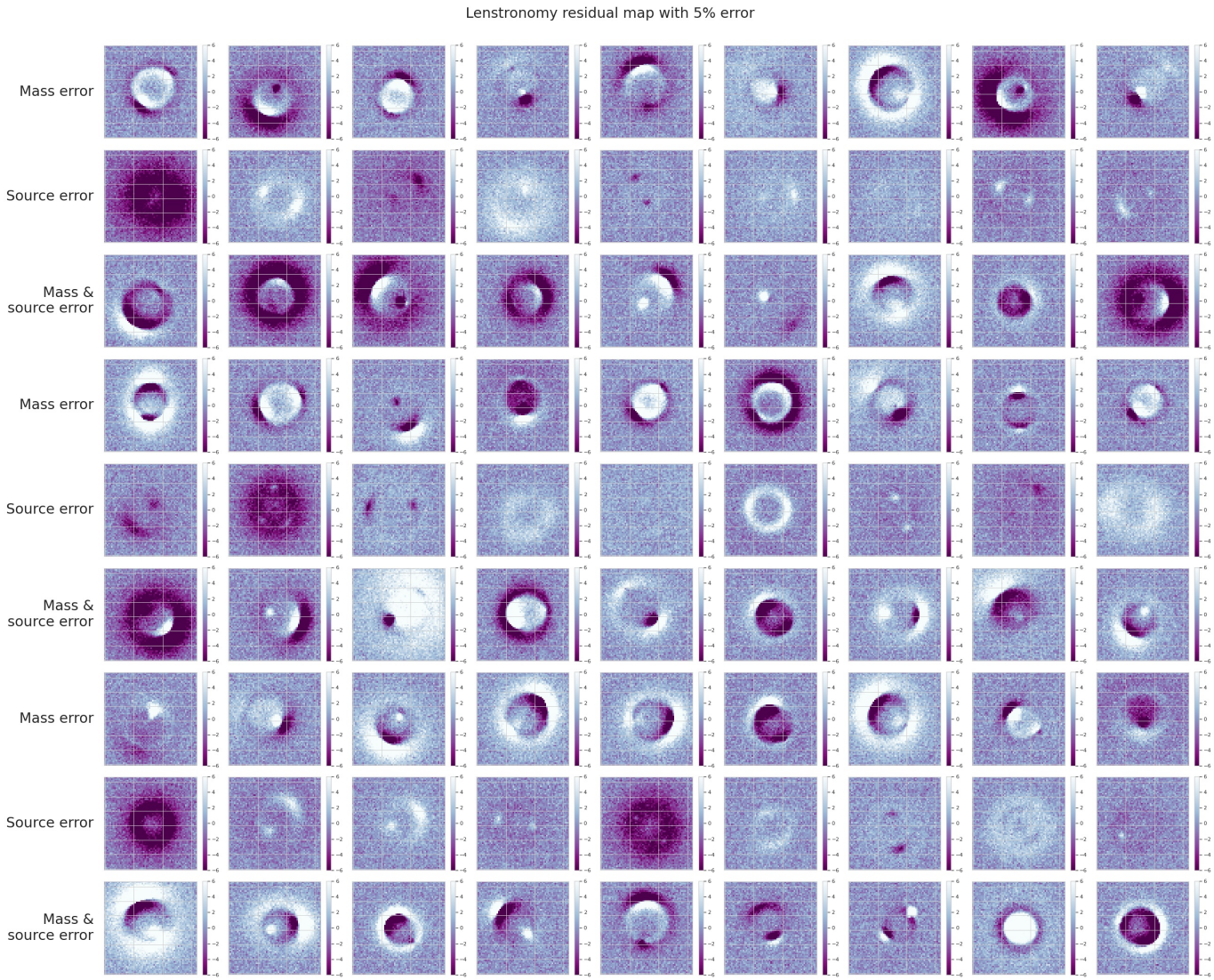


Fig. 2: Residual maps with 5% error: the percentage of error is the same for each label combination. The scale is between $[-6, 6]$. No constraints are set on the image rendering.

amplitude strictly greater than six are removed from the data set as they contain too obvious errors, such that for every residual map $|r(m,n)| < 6$. For images that we perceive as 'residual noise maps alike', the definition is more tedious. When we refer to 'residual noise maps alike', it corresponds to maps that seem to contain only noise, yet parameters have a fixed relative error compare to the imaging data parameters'. This data type occurs mostly in source residual maps, as some quantities are close to zero (Table I).

As the noise distribution is uncorrelated and has zero mean, we apply a chi-square test. For residual maps with no errors, we expect a chi-square distribution around one. While for residuals with errors, we forecast a chi-square distribution shifted to higher values.

$$\chi^2 = \frac{1}{NM} \sum_{m=0}^{M} \sum_{n=0}^{N} r(m,n)^2 \tag{8}$$

Where $r(m,n)$ correspond to the pixel at line m and column n in the residual map r. Since we aspire to remove noise-like residual maps and residual maps with no errors have a chi-square close to one, we consider maps with a chi-square strictly above one. However, this limit is not sufficient as correct residuals are never equal to one and setting the limit to one still brings noise-like images as described in Figure 7, with residual maps with no errors around zero. Several values are tested and the Figure 6 is obtained with a lower limit $\chi^2_{low} = 1.2$ as shows Figure 8 with a shifted chi-square distribution of errors and an upper absolute amplitude

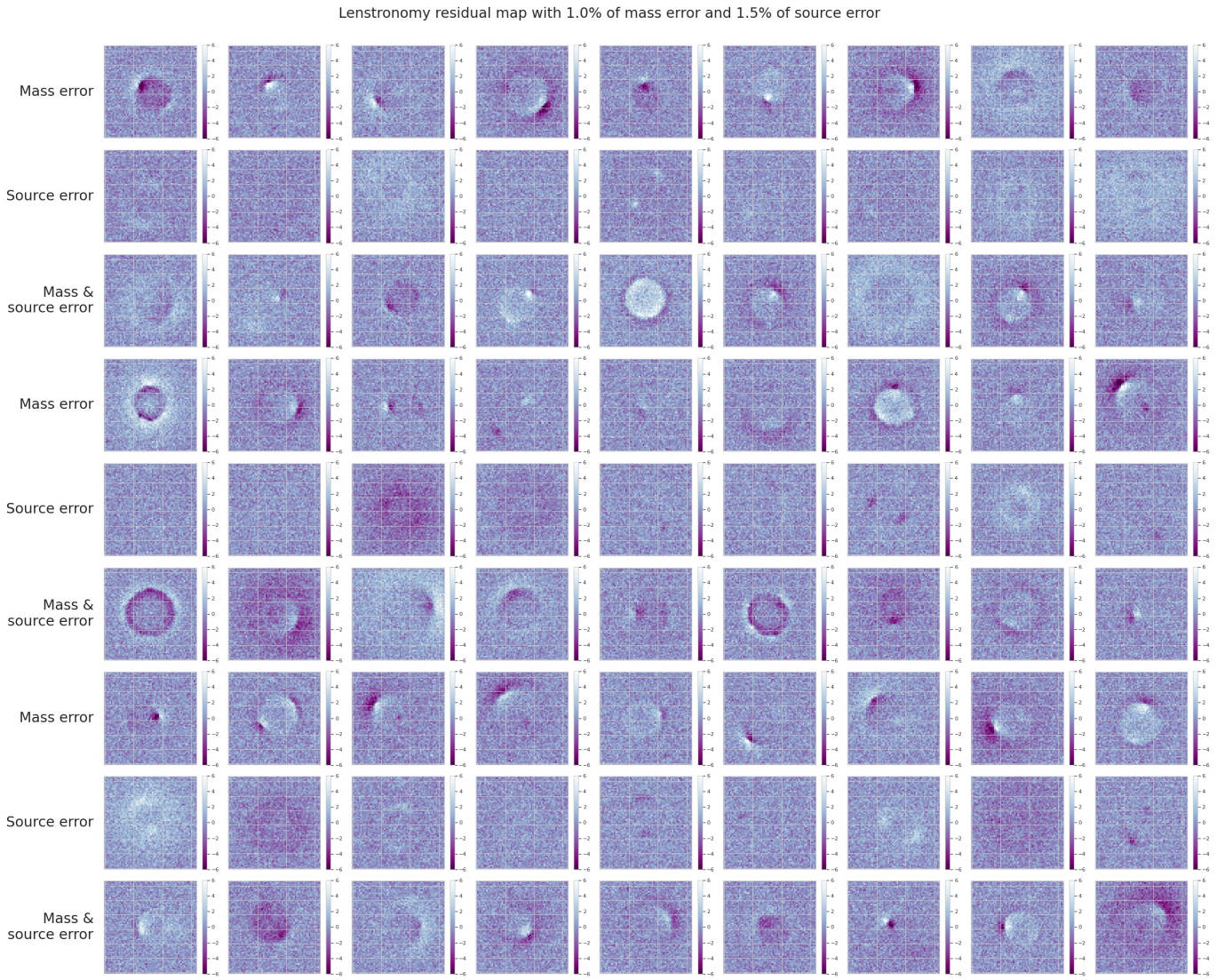


Fig. 3: Residual maps with 1% error : the percentage of error is the same for each label combination. The scale is between $[-6, 6]$. No constraints are set on the image rendering.

$|r(n, m)| < 6$. We observe one major tendency with source and mass errors: source errors are more diffused in the image, while mass errors are more concentrated around the ring. Due to this property, source errors are more inclined to have a noise like behaviour and mass error tends to have more obvious errors. It also means that the neural network is more likely to mistake a mass-source error residual map with a mass error residual map.

C. Final distribution

The distribution of the data-set in Figures 9 and 10 is statistically similar for maps with errors and maps with no errors. We note that for the Einstein radius θ_E, the power-law slope γ_s, the half-light radius R and the Sersic index n, some samples differ significantly from other samples. Those outliers come from the log-normal distribution. With the constraints, we observe in Figure 9 a shift in the distribution for the Einstein radius θ_E, and the power-law slope γ_s. The ellipticity's distribution is slightly changed; nevertheless, the change is insignificant compare to other quantities. In Figure 10, we notice that the centre x and y have more points concentrated in the image centre. Centred sources bring well-defined rings, while decentralized sources tend to have partial rings. Sersic radius R is slightly shifted to the left while the Sersic index is concentrated close to zero. Just as before, ellipticity's distribution changes are insignificant compare to other parameters. For each parameter, distribution is slightly changed, which imply a highly significant dataset.

D. Labelling

We perform the labelling with a multi-label encoded vector. This method is used in categorical classification as most

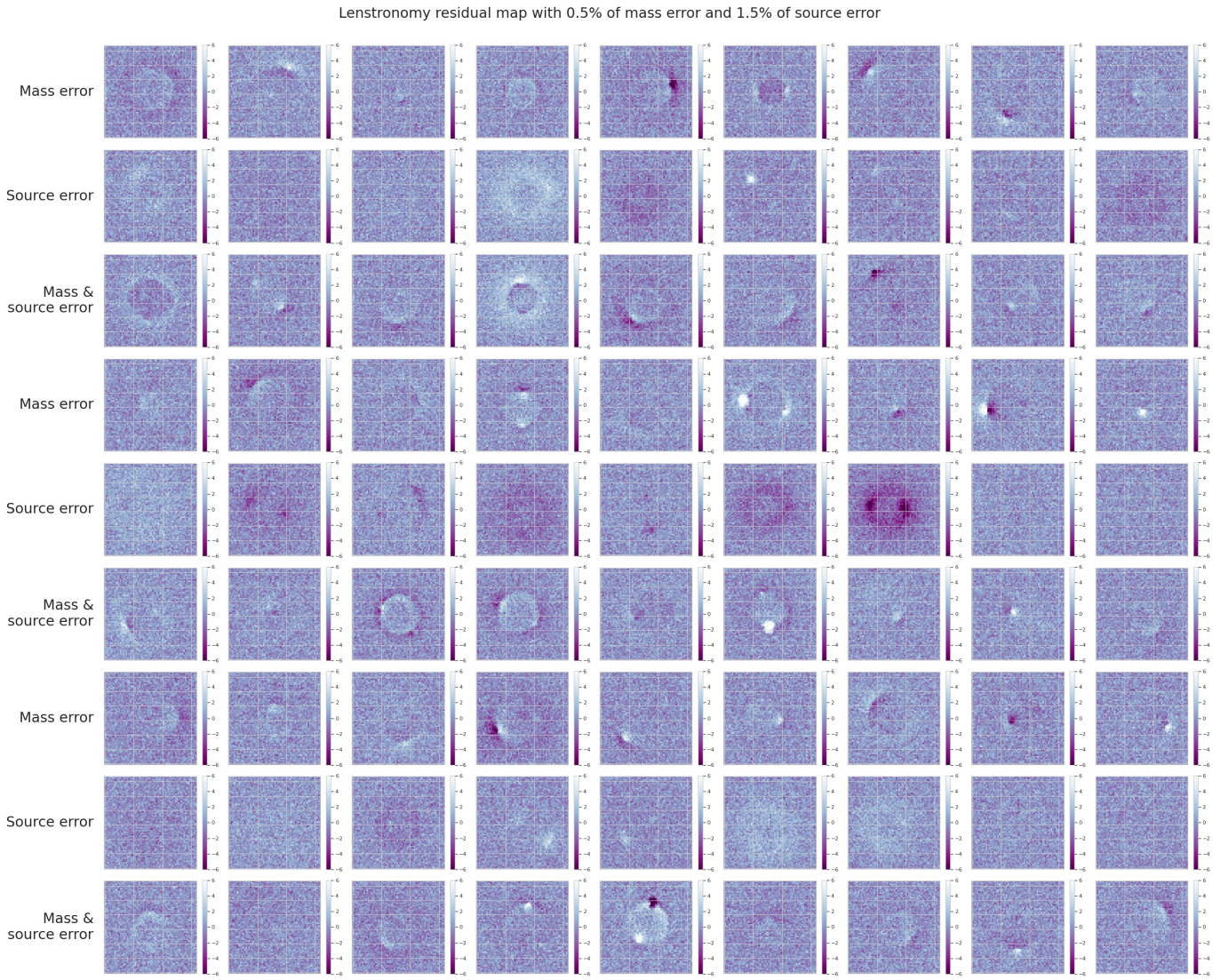


Fig. 4: Residual maps with separate error for each label combination: $p_m = 0.5\%$, $p_s = 1.5\%$ and $p_{m\&s} = 0.5\%$. The scale is between $[-6, 6]$. No constraints are set on the image rendering.

TABLE III: CNNs are trained on this residual maps parameters

Component	Value
Simulated Images	
Ratio	75 %
Percentage of error	
Mass	$p_m = 0.5\%$
Source	$p_s = 0.5\%$
Mass & Source	$p_{m\&s} = 0.5\%$
Bounds	
Mean square error	$\chi^2_{low} = 1.2$
Maximum amplitude	$\lvert r(n,m)\rvert < 6$

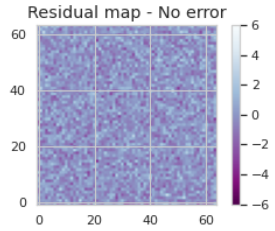


Fig. 5: Residual maps with no error. The scale is between $[-6, 6]$. Constraints have no effect on the residual maps with no error.

machine learning algorithms can not operate directly with categorical data. Thus, we define each error through probability. Naturally, mistakes can be combined, which means that our problem is a multi-labelling optimization problem. As the loss function solves a non-exclusive class problem, adding a class for the non-error case bring an additional error to our classification task as error labels have a clear correlation

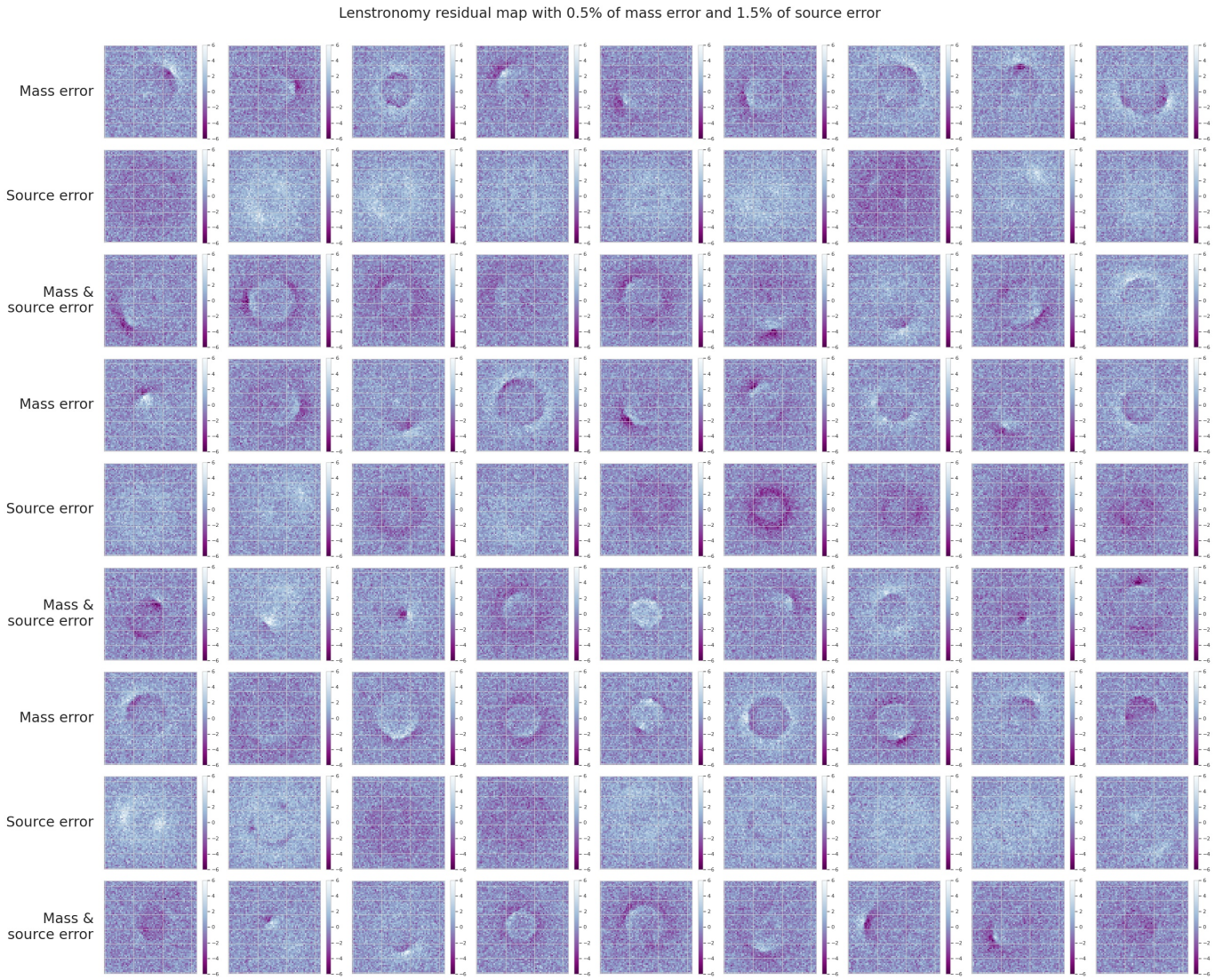


Fig. 6: Residual maps with separate error for each label combination: $p_m = 0.5\%$, $p_s = 1.5\%$ and $p_{m\&s} = 0.5\%$. The scale is between $[-6, 6]$. Obvious error maps are removed from the dataset such that the pixel's maximum absolute amplitude $|r(n, m)| < 6$. Noise-like maps are removed from the dataset such that chi-squared score of residual maps are $\chi^2 \geq 1.2$. This data set is the data-set used for our final training set.

(Section VIII). Thus, we consider two labels with mass and source errors.

VII. Data saving format

The simulator returns residual maps as numpy arrays, while the metadata is a pandas data frame [10]. This project aims to handle huge heterogeneous data sets, implying that we need to study efficient ways to store this data. Several options are available:

- Storing to disk: In this method, we save the image in a CSV file. The storage memory is therefore small compared to other methods (approximately 200[MB] for 10000). Nevertheless, the main disadvantages of this method are that the storage and the reading time is larger than other methods($\simeq 70$[s]).
- LMDB Lightning Database: It is known mainly for its efficiency. This efficiency is since it is a memory-mapped method (inferior to 10[s] for reading and storing). However, it uses much memory compared to the two previous methods (more than 400[MB] for 10000).
- HDF5 Hierarchical Data Format: This type of data storage is probably the best trade-off between disk usage (approximately 300[MB] for 10000) and storing and reading time (< 10[s] for reading and storing).

Thus, to have a good ratio between disk usage, storing and reading time, HDF5 storage is picked for the data saving

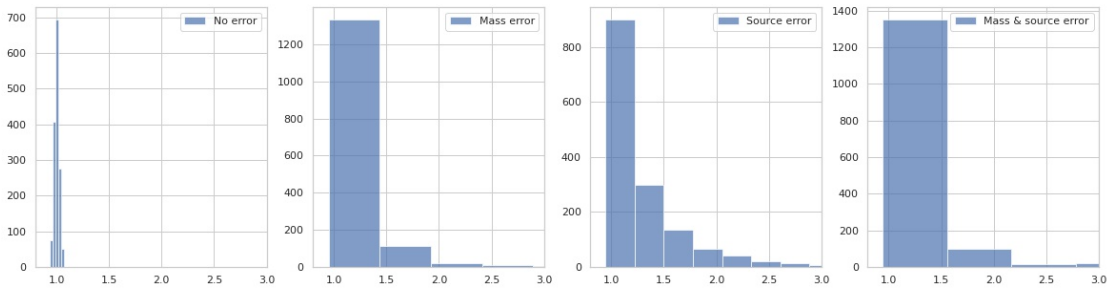


Fig. 7: Chi square test distribution of the data set. Samples : 6000 - Percentage of error : $p_m = 0.5\%$, $p_s = 1.5\%$, $p_{m\&s} = 0.5\%$. No constraints are applied

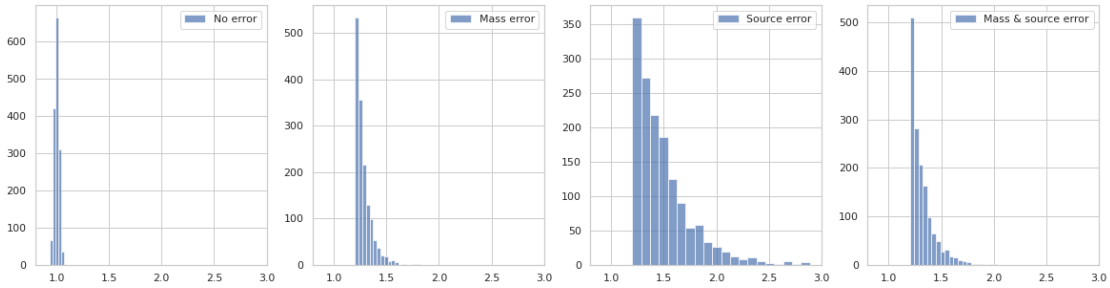


Fig. 8: Chi square test distribution of the data set. Samples : 6000 - Percentage of error : $p_m = 0.5\%$, $p_s = 1.5\%$, $p_{m\&s} = 0.5\%$. Obvious error maps are removed from the dataset such that the pixel's maximum absolute amplitude $|r(n,m)| < 6$. Noise-like maps are removed from the dataset such that chi-squared score of residual maps are $\chi^2 \geq 1.2$. This data set is the data-set used for our final training set.

format.

VIII. Neural network structure

We use large convolutional networks in image classification tasks, implying a considerable training time and a large memory usage. Therefore, neural networks are developed on the Pytorch framework to provide fast and efficient computations.

A. Loss function

In classification tasks, we often use binary cross-entropy as it provides a predicted probability on the label. The loss function combines the label y_i with the predicted value x_i. Both values are contained in $x_i, y_i \in [0, 1]$.

$$J(x,y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln(x_i) + (1-y_i)\ln(1-x_i) \quad (9)$$

Nevertheless, this definition is inadequate for multi-labelling classification, as our predictions should be in the range $x_i \in [0, 1]$ with independence between each classes' prediction. An output function must be introduced: the Sigmoid function. The sigmoid function provides to the loss function a forecast of each label independently on the last layer. Losses of each label are then summed up and represent our final loss function.

One classes loss :

$$J_l(x,y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \ln\left(\frac{1}{1+e^{-x_i}}\right) + (1-y_i)\ln\left(\frac{e^{-x_i}}{1+e^{-x_i}}\right) \quad (10)$$

Final loss :

$$J(x,y) = \frac{1}{L}\sum_{k=1}^{L} J_l(x,y) \quad (11)$$

The neural network's output is rounded for the test set such that $y_{pred} \in \{0, 1\}$, yet not for the training, as it brings poor results. We only perform the rounding during our analysis for predictions; we expect a binary answer for each label for real experiments $y_{pred} \in \{0, 1\}$.

B. Metric

The metric choice is crucial to select our final neural network. As we deal with a multi-labelling task, resulting predictions are often partially correct. Multi-label classification task solves different classification tasks, yet the tasks are related somehow [11]. Therefore, although we present several metrics in the Appendix XV-A, we only focus here on our final choice: Area Under the Receiver Operating Characteristics (AUROC).

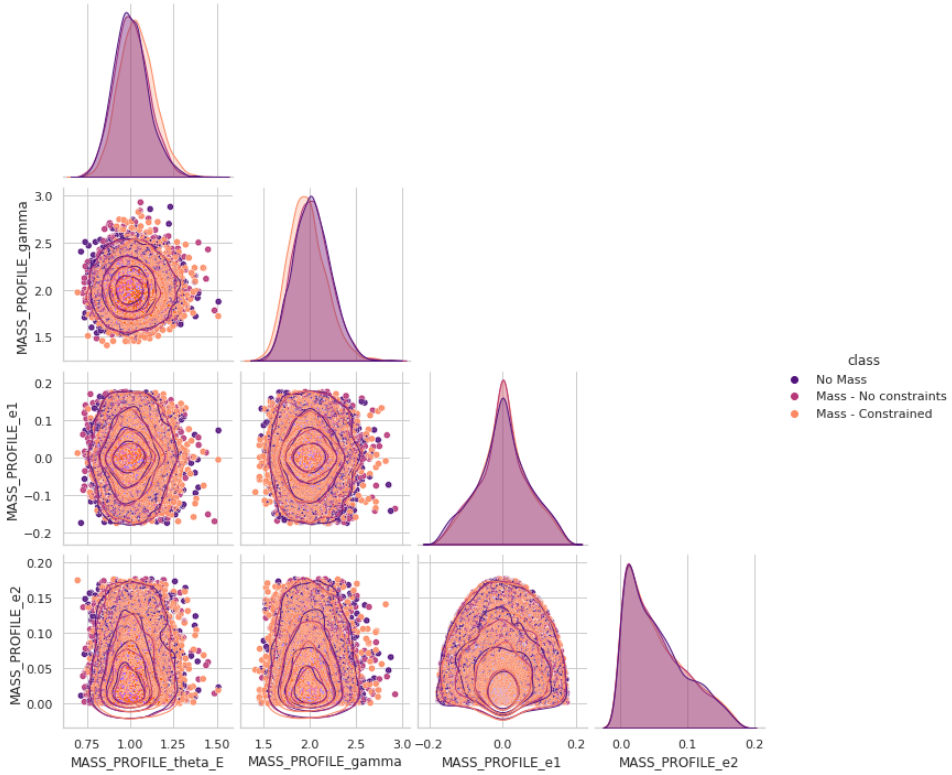


Fig. 9: Distribution of the data set over the mass parameters for the described distribution in Table I. Samples : 6000 - Percentage of error : $p_m = 0.5\%$, $p_s = 1.5\%$, $p_{m\&s} = 0.5\%$. Obvious error maps are removed from the dataset such that the pixel's maximum absolute amplitude $|r(n,m)| < 6$. Noise-like maps are removed from the dataset such that chi-squared score of residual maps are $\chi^2 \geq 1.2$. This data set is the data-set used for our final training set.

a) Area Under the Receiver Operating Characteristics (AUROC): AUCROC is a metric that corresponds to the integral of the Receiver Operating Characteristics (ROC) curve. The ROC curve is a performance measurement for classification problems. The True Positive Rate (TPR) is plotted over the False Positive Rate (FPR).

$$TPR = \frac{TP}{TP + FN} \tag{12}$$

$$FPR = \frac{FP}{TN + FP} \tag{13}$$

An AUROC close to one means that the model easily distinguishes the two classes. On the contrary, an AUROC close to 0.5 implies a model that has no discrimination capacity. It also provides another quantity: decision threshold. The decision threshold gives an indication of the differentiation between two classes for a given model. A decision threshold with small variations between $[0, 1]$ implies that the model has difficulty distinguishing between two classes, while a threshold with wide variation tends to discriminate more the two classes. Thus, it does not necessarily imply a good classification. Nevertheless, this metric performs well on a large balanced dataset.

b) Conclusion: The final metrics choice highly depends on the balance of the data set. For highly imbalanced data sets, recall, precision, or F1-score are relevant, while accuracy, hamming loss or ROC AUC are pertinent for balanced data sets. We set a ratio of 75% of residual maps with errors in our experiment, which leads to a balanced data set (Figure 11). Thus, ROC AUC is selected as our final choice since it gives a statistically consistent and more discriminating measure than other balanced metrics. In classification, a statistically consistent model implies that for a large training set, the classification is close to a fully known population distribution. Furthermore, a more discriminating model involves the minimization of cases where the classification is ambiguous. If the balance of the dataset is imbalanced, we advise a metric such as the F1-score.

C. Cross validation

Cross-validation is a model validation technique, where one section is used as training while the remaining section is used for testing. Thus, it is performed several times with the same overall dataset, yet with a different repartition in the training and validation set. As a result, it gives a more accurate estimation of the model performance as the resulting predictions highly depend on the data distribution. However,

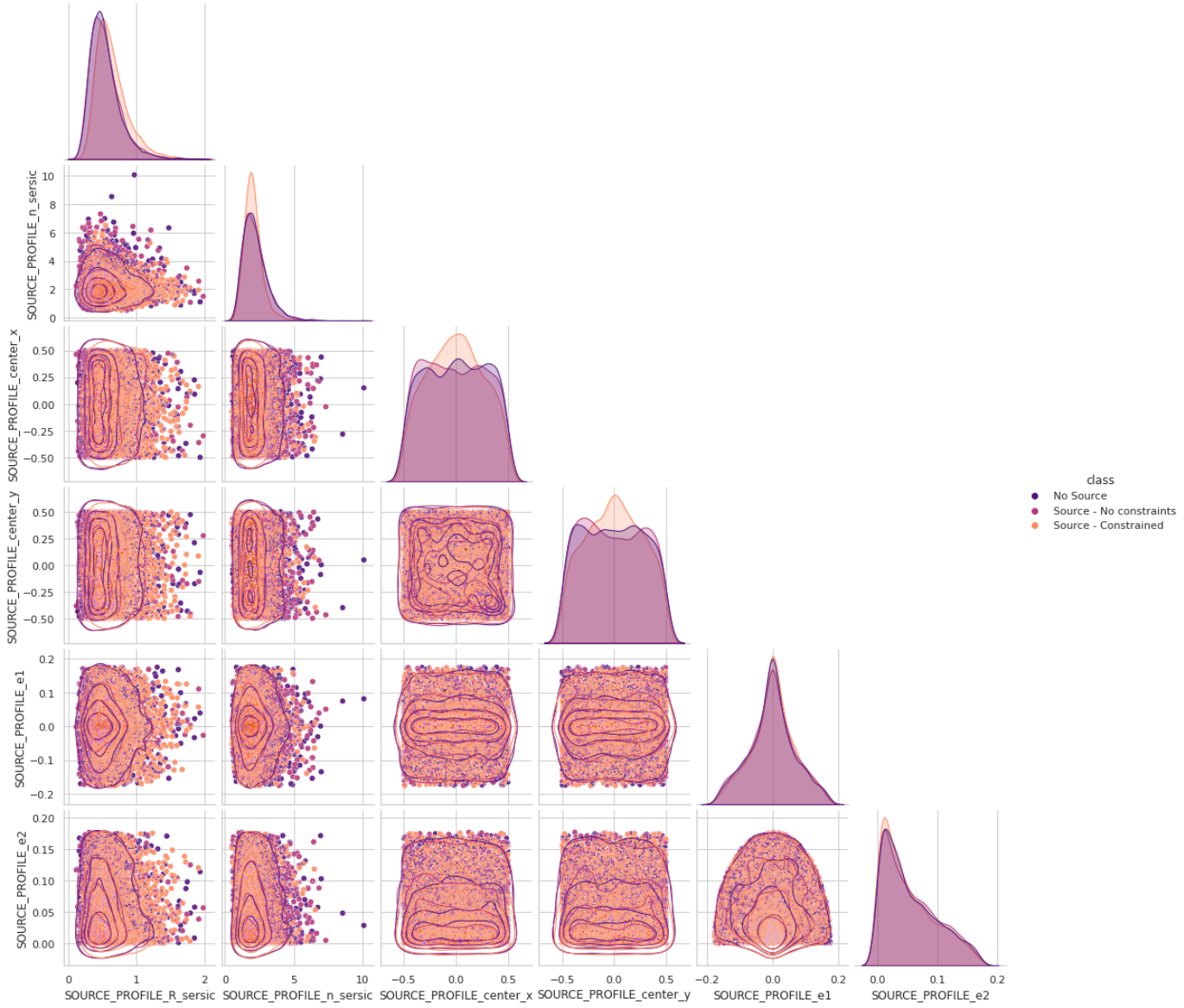


Fig. 10: Distribution of the data set over the source parameters for the described distribution in Table I. Sample size : 6000 - Percentage of error : $p_m = 0.5\%$, $p_s = 1.5\%$, $p_{m\&s} = 0.5\%$. Obvious error maps are removed from the constrained dataset such that the pixel's maximum absolute amplitude $|r(n,m)| < 6$. Noise-like maps are removed from the dataset such that chi-squared score of residual maps are $\chi^2 \geq 1.2$. This data set is the data-set used for our final training set.

this method is time-consuming, so we use it in our analysis for hyperparameters selections.

D. Optimizers

Optimizers are methods that change the attributes of the neural network to reduce losses. Learning rate and weight update are highly related to optimizers, explaining why optimizers must be selected wisely. The goal is to achieve a model with fast converges.

a) Stochastic Gradient Descent (SGD): SGD is an extension of Gradient Descent (GD). It handles one sample instead of using the whole dataset, which requires less memory compared to GD. However, it introduces severe oscillations in the learning process, can be easily stuck at a local minimum,

TABLE IV: Optimizers parameters

Optimizer	Parameters
SGD	
Learning rate	$lr = 0.001$
SGD with momentum	
Learning rate	$lr = 0.001$
Momentum	$\gamma = 0.9$
Adam	
Learning rate	$lr = 0.001$
Exponential decay	$\beta_1 = 0.9$, $\beta_2 = 0.99$
Threshold	$\epsilon = 0.07$

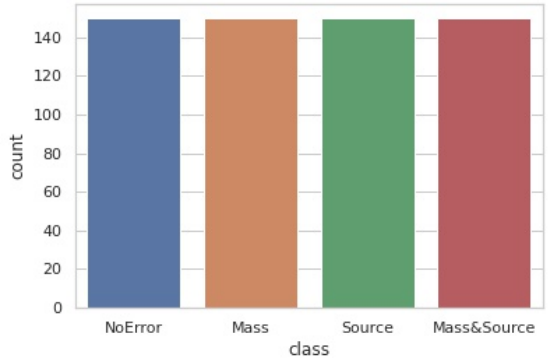


Fig. 11: Classes distribution for an error ratio of 75% : the error ratio consider mass error, source error and mass with source error

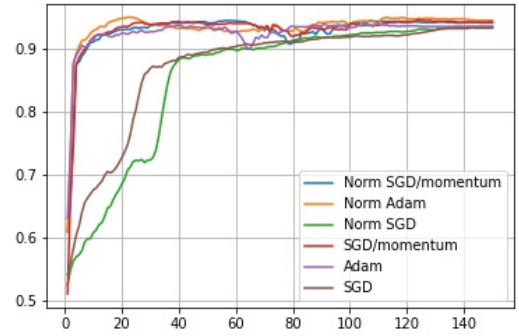


Fig. 12: Optimizers performance on the baseline neural network. Samples : 6000 - Maximum epoch : 150 - Batch size : 50 - Training set : 90% - Data: Table III

and has a slower convergence than GD.

b) Stochastic Gradient Descent (SGD) with momentum: To overcome SGD's problems, we suggest adding momentum to SGD[12]. SGD with momentum reduce the severe oscillations of SGD and provide a faster convergence than the GD. However, we need to specify an additional hyperparameter to our optimizer.

c) Adaptive Moment Estimation (Adam): Adam can be designated as a combination of RMSprop and SGD with momentum[13]. The main properties of this approach are that it handles well sparse gradients on noisy problems and is computationally efficient.

d) Final optimizer: In Tables VI, V and Figure 12, three optimizers are tested, together with data normalization. We subtract the mean of each channel to normalize the data set. The standard deviation of each channel then rescales the data. Firstly, SGD learns slowly compare to the two other optimizers and has a lower AUROC (Figure 12). Finally, SGD with momentum is slower than Adam and provide better results compare to Adam (Table V). In Table VI, we observe that the normalization has a small effect on the resulting classification since maps that only contain noise are already normalized, and the amplitude variation is not that significant in residual maps with error. However, adding normalization for larger variations of amplitude seem to perform better. With Table VI, our final optimizer is consequently SGD with momentum with normalization.

E. Reproducibility

The separation and the generation of the data are performed randomly with a fixed random number generator's seed, such that we obtain the same training and test set. In Pytorch, the weight initialization is done randomly, resulting in classifiers delivering different results for the same training set. We fix the random number generator's seed in Pytorch to avoid this phenomenon. We perform the training and the testing of the baseline neural network to make sure the generated results is the same each.

TABLE V: AUROC of different optimizers on the baseline convolutional neural networks. Samples : 6000 - Maximum epoch : 150 - Batch size : 50 - Training set : 90% - Data: Table III

Optimizer	**SGD**	**SGD with momentum**	**Adam**
Not normalized			
AUROC	0.941	0.950	0.944
Epochs	149	149	127
Normalized			
AUROC	0.944	0.947	0.944
Epochs	141	140	32

IX. Baseline implementation

We build the baseline implementation such that the input (I) image remains identical, allowing an $64 \times 64 \times 1$ input. Through each layer, we must select the kernel (K), the stride (S) and the padding (P) wisely such that the output (O) remains strictly positive. Thus, after one layer, the output has a size of $O \times O \times CH$, where CH corresponds to the output channel.

$$O = \frac{I - K + 2P}{S} + 1 \tag{14}$$

We implement three baseline models that use respectively residual maps, metadata, residual maps and metadata. From Table VII, several conclusions can be drawn. Firstly, adding metadata to the neural networks' training seemed to affect the AUCROC, yet we cannot draw any clear conclusion from this observation without cross-validation. After the 150 epoch, the metadata and the residual maps neural network's training in most cases is not finished as the AUROC is still increasing, suggesting that we can achieve a higher AUCROC with metadata and images. To prove this statement, we must apply cross-validation and train the neural network on a larger number of epochs. Secondly, a more pronounced error provides a better classification, mostly because errors are easier to spot for the neural network. Thirdly, the error variation does not affect the neural metadata network as it is only visible on residual maps. Finally, data imbalance provides a less accurate classifier in most cases. Thus, we must remain critical of that statement,

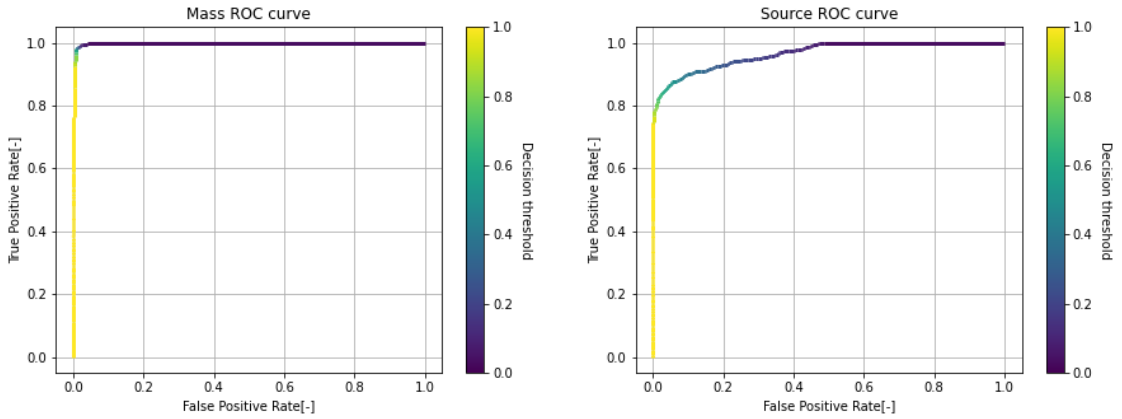


Fig. 13: Receiver operating characteristic curve (ROC) between the mass error and the source error. Samples : 10000 - Batch size : 64 - Optimizer : SGD with momentum - Data : Table III - Epoch : 20 - AUC mass : 0.998 - AUC source : 0.965 - Without rounding

TABLE VI: AUROC of different optimizers on the baseline convolutional neural networks with 5-fold cross validation. Batch size : 50 - Max epochs : 150 - Samples : 60000 - Data : Table III

Optimizer	**SGD**	**SGD with momentum**	**Adam**
Not normalized			
AUROC	0.938	0.950	0.948
Normalized			
AUROC	0.943	0.950	0.949

TABLE VII: Baseline results : Cross entropy loss function for 6000 samples with AUROC metric. Balance corresponds to the quantity of residual maps with errors in the final data sets. Optimizer: SGD with momentum - Batch size : 50 - Maximum epoch : 150

Neural networks	**1%**	**2%**	**5%**
Balance : 50%			
Residual maps	0.934	0.953	0.960
Metadata	0.508	0.503	0.514
Residual maps & metadata	0.939	0.956	0.959
Balance : 75%			
Residual maps	0.932	0.954	0.958
Metadata	0.513	0.509	0.493
Residual maps & metadata	0.925	0.952	0.967
Balance : 99%			
Residual maps	0.922	0.954	0.961
Metadata	0.522	0.512	0.522
Residual maps & metadata	0.932	0.955	0.961

as the metric AUCROC is not adapted for an imbalanced data set. Metrics such as F1-score, recall or precision should bring more relevant values in imbalanced data set.

Finally, in Table VIII, we observe that the removal of the noise-like residual improves the resulting classifier performance. Due to an important pixels' amplitude, the percentage 5% is not studied since most generated maps are not suitable for our maximal absolute amplitude criterion. Just as in the previous observation, the added metadata seems to improve the model's performance, just as increasing the error percentage.

TABLE VIII: Baseline results : Cross entropy loss function for 6000 samples with AUROC metric. The balance corresponds to the quantity of residual maps with errors in the final data sets. Optimizer: SGD with momentum - Batch size : 50 - Maximum epoch : 150 - Data: Table III

Neural networks	**1%**	**2%**
Balance : 50%		
Residual maps	0.944	0.956
Metadata	0.647	0.704
Residual maps & metadata	0.950	0.960
Balance : 75%		
Residual maps	0.944	0.956
Metadata	0.647	0.704
Residual maps & metadata	0.950	0.960
Balance : 99%		
Residual maps	0.946	0.963
Metadata	0.663	0.704
Residual maps & metadata	0.951	0.960

TABLE IX: Baseline results. Samples: 10000 - Data: Table III - Training size : 85% - Batch size: 64 - Optimizer: SGD with momentum - Rounding : $\{0,1\}$

Model	**Baseline**
Epoch	20
Test AUROC	0.947
Mass AUROC	0.982
Source AUROC	0.913

In Figure 13, the mass error classification increases drastically compare to the source error. The range of the mass error is larger than the source error, which implies an easier detection from the neural network and thus a larger area under the ROC curve. Moreover, we can reach a good differentiation between labels, even for a quite simple model (Table IX).

X. Deep Convolutional Neural Network

A. Data preprocessing

1) Padding & Interpolation: Due to the neural network structure of widely used CNNs, the input requires a certain size: 224×224. A larger input admits deeper convolutional neural networks, which may be beneficial for accuracy. Two options are then available to resize the dataset: zero paddings and interpolation. Zero padding consists of adding zeros and the initial picture, while interpolation consists of enlarging the picture to the desired size through numerical interpolation methods. Zero paddings avoid deformation of the image and speed up the calculation compare to interpolation. The neighbouring zero inputs are not activated in the convolutional network, which speeds up the training. The non-activation of units behaves as a dropout on images' corners during the training, which may bring a more generalized model. On the other hand, interpolation will impact the classification while causing an increase in the computation time[14]. For larger images, the advised approach divides the pictures into smaller images and provides them to the neural network. In a convolutional network, the neighbouring units are important for classification, which allow the division of images.

2) Normalization: In the baseline section, we observe that normalization does not carry necessarily better results. Thus, as in general it seems to improve the results, we consider normalization for the training and testing of the presented architecture.

3) Training set: To quantify the efficiency of our model, the training and the testing set should be defined. We use the training set to train neural networks, while the test set shows the neural network's performance for unknown labels. The test set has to be small enough to have a good training set for neural network learning while being large enough to predict neural networks' performance accurately. We set the percentage of the training set to 85%.

B. Models

In our analysis, we focus on image classification neural networks that are widely used. Most are designed for multi-class classification with a softmax function as an output layer. Therefore, we rearrange the output to fit a multi-label classification. The generated data has only one channel, which means that the input channel must be changed (usually three for the RGB channels).

1) Spatial exploitation based CNNs: Spatial exploitation based CNN are based on consecutive CNN layers that combine different correlation levels thanks to filters. Hence, we utilize the position of elements in the image to classify. The main issue with those networks is that deep neural networks suffer from vanishing gradient, stopping the neural network's learning. Our problem focuses on small variation in the residuals, implying that we expect good results for small filters.

a) AlexNet: AlexNet is a convolutional network that was introduced for image classification in 2012 [17]. It is interesting as it is the first convolutional network that introduced a general image classification of objects for numerous data sets. AlexNet is not the most performing CNN for this type of classification since filters are designed to detect large objects.

b) VGG11: VGG11 ensues from AlexNet and incorporates smaller convolutional filters with substantially more maximum pooling [18]. Smaller kernels have the advantage to reduce the number of parameters and bringing a more discriminating decision function. As kernel definition is smaller than AlexNet's, the detection of smaller objects is more accurate than AlexNet.

c) GoogleNet: Introduced in 2015, GoogleNet manages through inceptions layer by reducing images size while keeping valuable spatial information[19]. The main advantage of inception layers is that they study images on different scales and classify small and low-resolution objects well.

2) Multi-path exploitation based CNNs: Multi-path exploitation based CNNs tackled the vanishing gradient and the degradation that occurs in most deep CNNs. We define degradation by a higher training error for deeper neural networks. If we consider a neural network with n layers, we expect a neural network with $m > n$ layers to perform similarly or better than the n layers network. However, it is not the case in practice, so introducing a residual neural network can benefit the training error. This phenomenon comes from the fact that multi-path CNNs use short-cuts to keep the overall shape and contour. However, it has difficulties dealing with complex images, which may cause undesired behaviour.

a) Resnet18: Residual network relies on a residual block that adds a skip connection after several convolution layers [16]. Having deep neural networks is desirable as it increases the level of features and thus gives better predictions. It also maintains lower complexity features compare to DenseNet since each layer has past dense block outputs. s

b) DenseNet 121: Just as ResNet, DenseNet rely on skip connections after several convolutional layers[20]. As it has a larger number of parameters, the training is significant.

3) Resource limited CNNs: We characterized resource-limited CNNs by a few features compared to the previous neural network. The small number of feature enables faster learning while keeping relatively high accuracy.

a) SqueezeNet 1.1: The initial aim of SqueezeNet is to provide a smaller neural network to have faster training by minimizing the number of parameters[15]. We then set the kernel size to one with evenly spaced downsampling, which means early downsampling of images. With its size, it can reach better results than AlexNet, which make it interesting to study.

4) CNNs results: Some neural networks are trained with Adam optimizer as Adam speeds up the learning process compared to SGD with momentum while maintaining good results (Table V and Table VI). We describe the prediction AUROC score in the Table X which correspond to a rounding operation on the neural networks' results ($y_{pred} \in \{0, 1\}$), while the probability AUROC score defined under the ROC curves are directly taken from the neural networks' output ($y_{pred} \in [0, 1]$).

a) Learning rate: In Figure 14, we observe that AlexNet and SqueezeNet tend to be slower than other neural networks during the learning process and achieve less accurate prediction. SqueezeNet is faster than AlexNet due to its structure, yet AlexNet brings better results (Table X). AlexNet, as said previously, focuses on spatial exploitation of images, making it probably more efficient in error detection as it compares the current position in the image with the neighbouring pixels. Other neural networks, on the other hand, learn faster and better than AlexNet. These differences in learning probably come from the smaller kernel definition in other neural networks, which decreases the number of required epochs.

b) Vanishing gradient & degardation: By studying spatial exploitation CNNs, we observe that deeper networks such as GoogleNet tend to deteriorate the resulting predictions compare to other networks such as VGG11. This result is caused by the number of layers that increase the effects of degradation and vanishing gradient. Nonetheless, by performing a longer training over GoogleNet, we may reach better results. On the other hand, multi-path exploitation based CNNs reach a similar performance as VGG11, even though they have more layers. This phenomenon comes from the skip connections that enhanced the training. The degradation and the vanishing gradient are still present, as shown with DenseNet121 and ResNet18, yet the effect is reduced.

c) Simplicity of model: Simpler models seem to bring better results. We see that the best achieved accuracy is the one performed by the baseline (Figures 14 and 13). Indeed, deep, complex neural networks tend to classify poorly due to degradation and vanishing gradient.

d) Classification between labels: Source error classification has a tendency to misclassify compare to mass error classification. In Figure 15 and 16, we witness that the mass error tends to be easily detected compare to source errors, and that mass errors are easier to differentiate compare to source errors. As we perceived in Figure 6, the mass error can be easily detected to the naked eye, while source errors are trickier to spot in mass and source errors. In general, this misclassification can come from an imbalanced data set; however, it is not the case in our analysis. Moreover, we can see that source errors ROC curve can be approximated into three affine functions. It suggests that the source errors tend to be classified as unknown labels ($y_{pred} = 0.5$).

e) Decision threshold: In Figures, 15 and 16, AlexNet source ROC, GoogleNet source ROC and SqueezeNet source and mass ROC have a small variation in threshold compare to other models. In Table X, we notice that the three performs poorly after the rounding operation and that the training is not completed. It means that the distribution between the two classes is overlapping, which intensifies misclassification. A longer training should solve this problem.

f) Rounding: In Figures 14, 15 and 16, we perceive that rounding reduces the final score of our classifier. This observation is strongly linked with the decision threshold, since source errors rounding for example tends to be less accurate than mass error rounding (Table X).

TABLE X: Results on various CNNs. Samples : 10000 - Batch size : 64 - Training set : 85% - Data : Table III - Rounding: {0,1}

Model	**AlexNet**	**VGG11**	**GoogleNet**
Epoch	50	16	30
Test AUROC	0.916	0.941	0.920
Mass AUROC	0.979	0.986	0.986
Source AUROC	0.853	0.896	0.854
Model	**ResNet18**	**DenseNet121**	**SqueezeNet**
Epoch	39	21	50
Test AUROC	0.937	0.940	0.887
Mass AUROC	0.983	0.986	0.947
Source AUROC	0.890	0.894	0.827

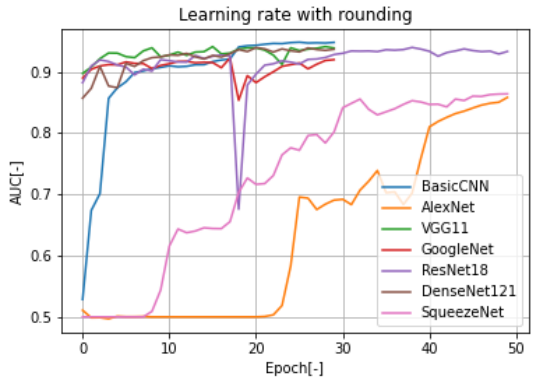


Fig. 14: Learning rate of each neural network. Neural networks that requires a large training time are trained over 30 epochs while others are trained over 50 epochs. Samples : 10000 - Batch size : 64 - Data : Table III - Rounding: {0,1}

XI. Ensemble methods

Ensemble methods rely on the combination of multiple trained models to get better results. We aspire to have a low bias and low variance final model, which is, in practice, for a unique model hard to achieve due to the opposite behaviour of both quantities. Ensemble methods undertake this issue and produce more robust and stable classifiers.

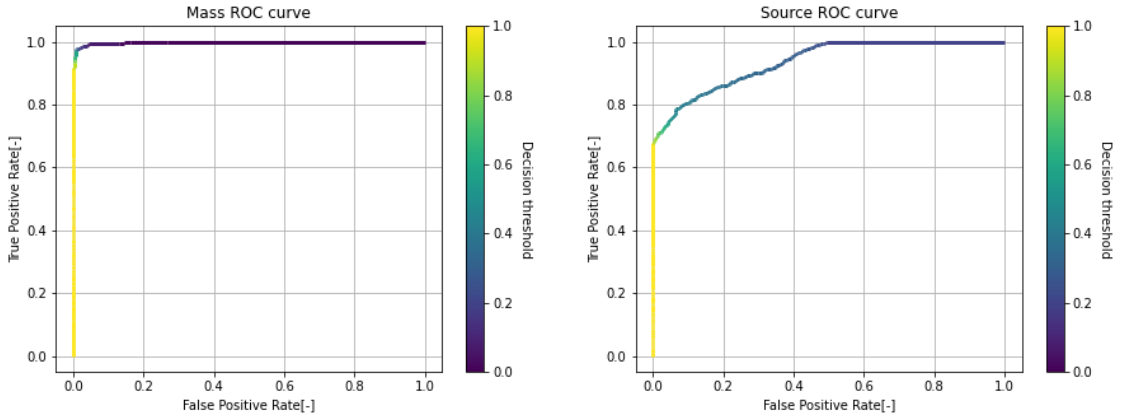
A. Problem transformation

The problem transformation algorithm defines how the classifier processes the input information. We focus on the binary relevance for our analysis, yet we must consider other approaches for further analysis.

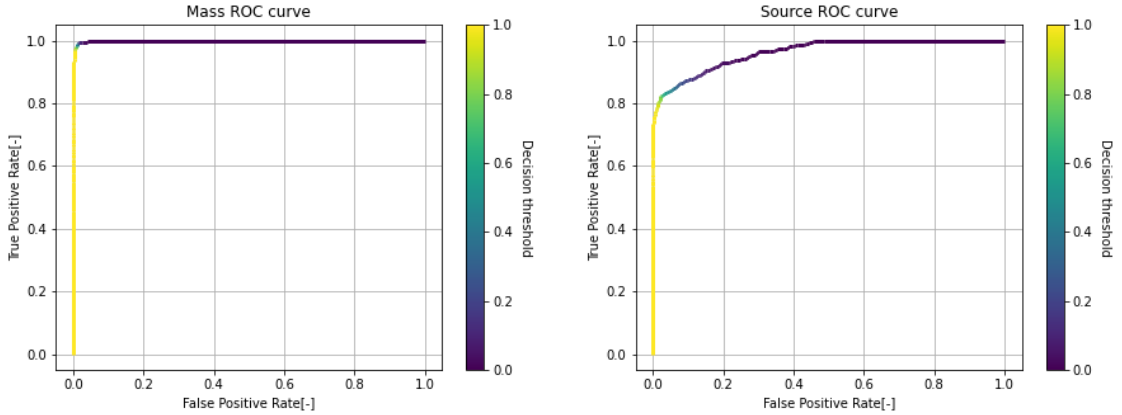
a) Binary relevance: This method consists of training each label separately, which makes its implementation easy[21]. For example, for a classification task of k-labels, each tag is separated and trains models independently. A major con is that this approach does not consider the correlation between stamps that may bring our classifier biases. Furthermore, it is not suitable for numerous labels.

B. Classifiers

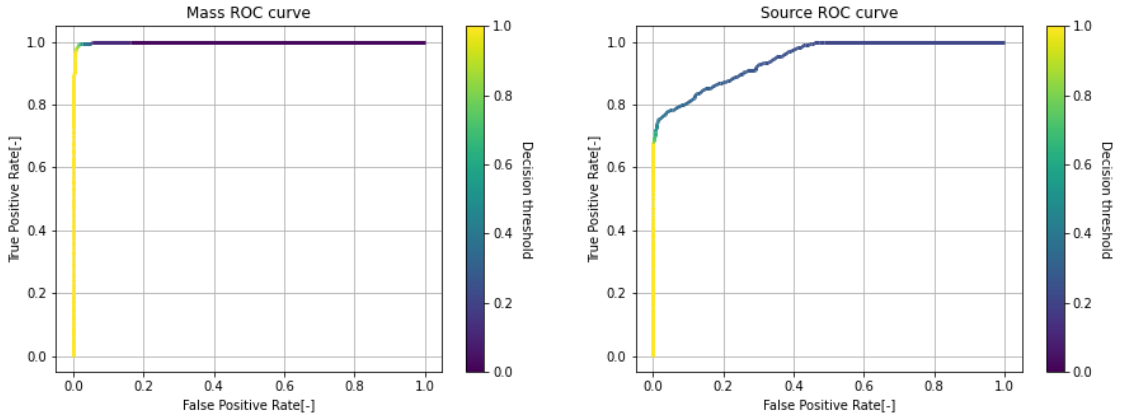
After setting the problem transformation, the newly transformed dataset trains a classifier through a supervised learning



(a) AlexNet ROC. Optimizer : Adam - Epoch : 50 - Mass AUCROC : 0.998 - Source AUROC : 0.938 - Without rounding

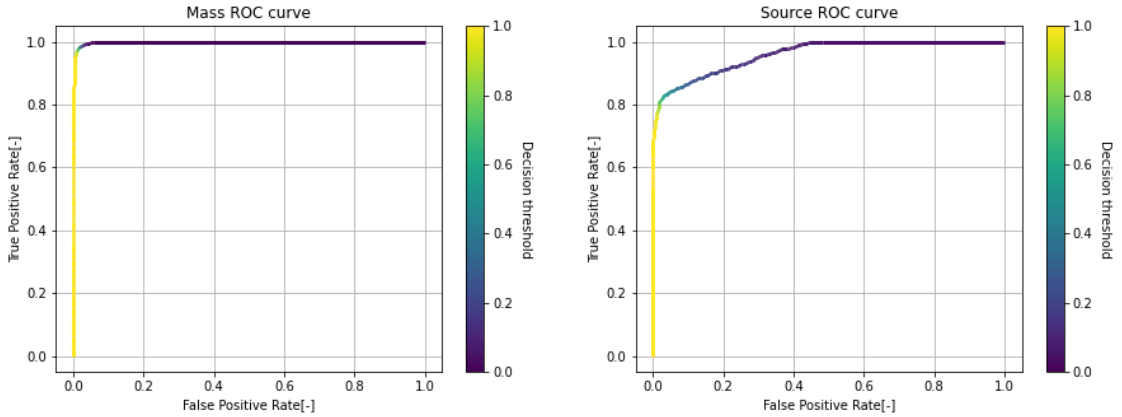


(b) VGG11 ROC. Optimizer : SGD with momentum - Epoch : 16 - Mass AUCROC : 0.999 - Source AUROC : 0.964

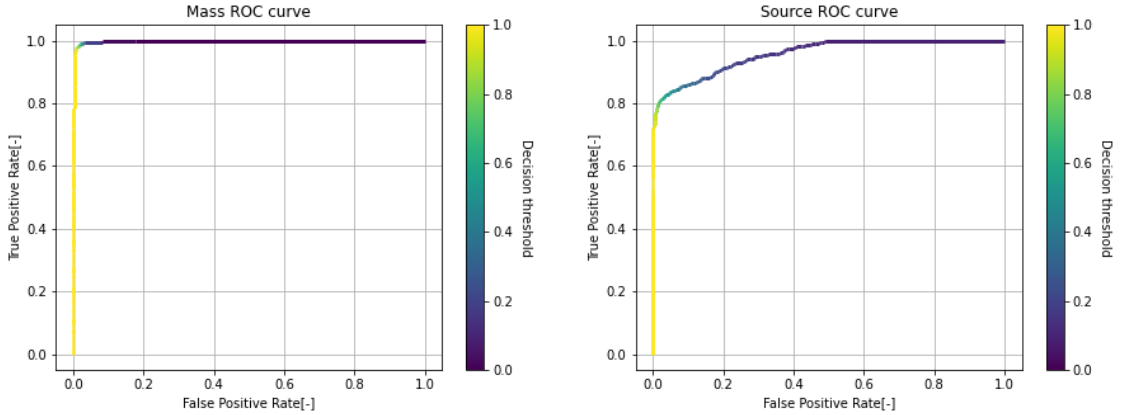


(c) GoogleNet ROC. Optimizer : SGD with momentum - Epoch : 29 - Mass AUCROC : 0.999 - Source AUROC : 0.946

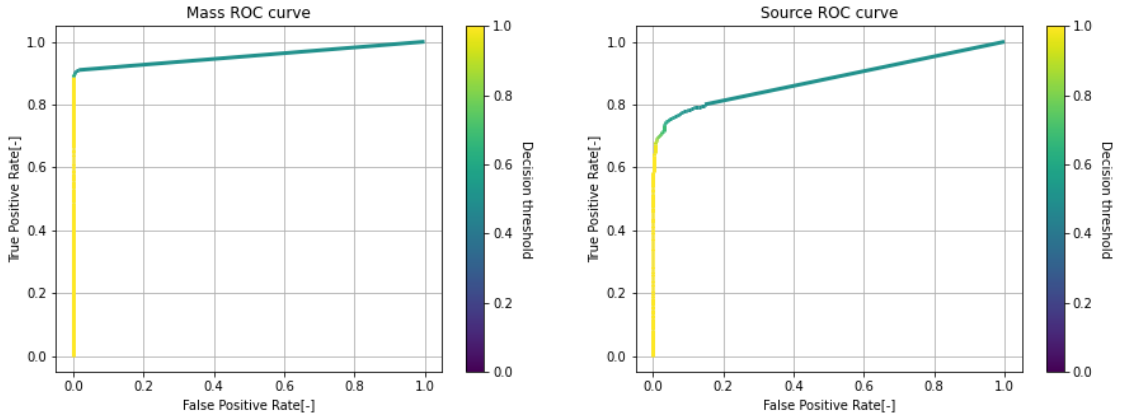
Fig. 15: Final ROC curves over different CNNs. Samples :10000 - Batch size : 64 - Training set : 85% - Data : Table III



(a) ResNet18 ROC. Optimizer : SGD with momentum - Epoch : 39 - Mass AUCROC : 0.999 - Source AUROC : 0.961

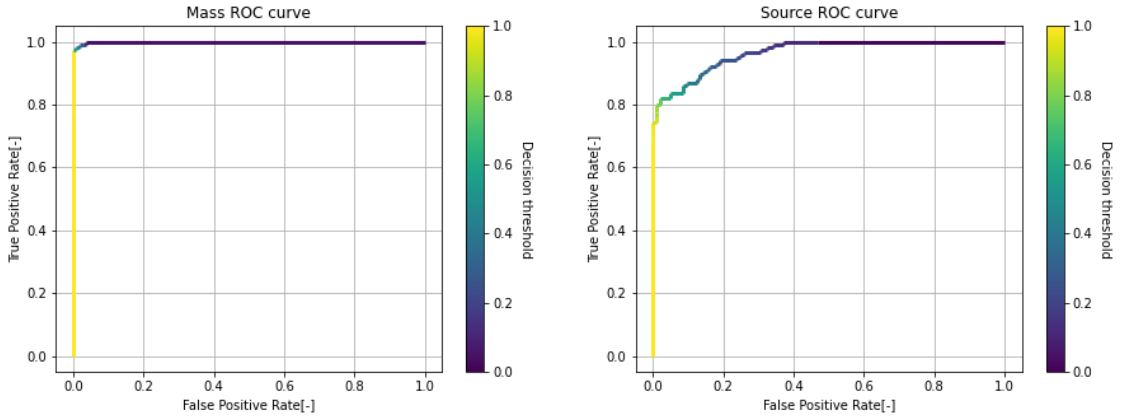


(b) DenseNet ROC. Optimizer : Adam - Epoch : 21 - Mass AUCROC : 0.986 - Source AUROC : 0.894

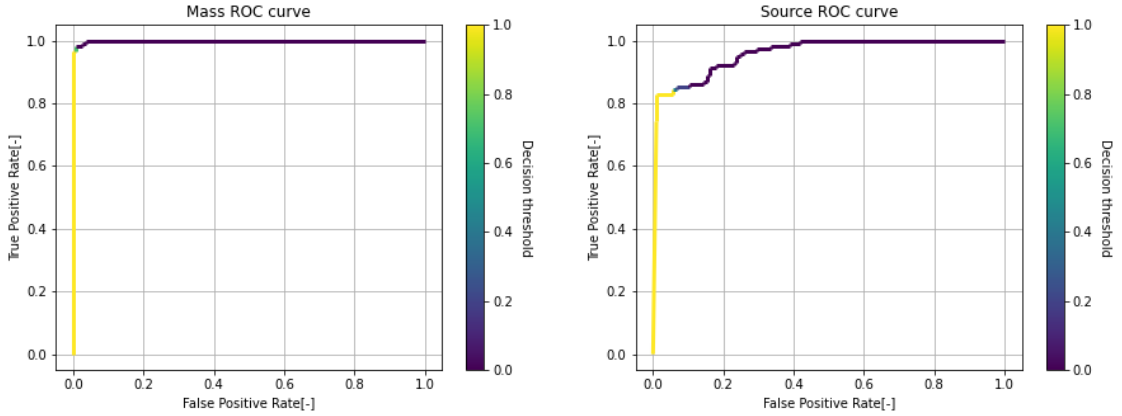


(c) SqueezeNet ROC. Optimizer : Adam - Epoch : 50 - Mass AUCROC : 0.954 - Source AUROC : 0.879

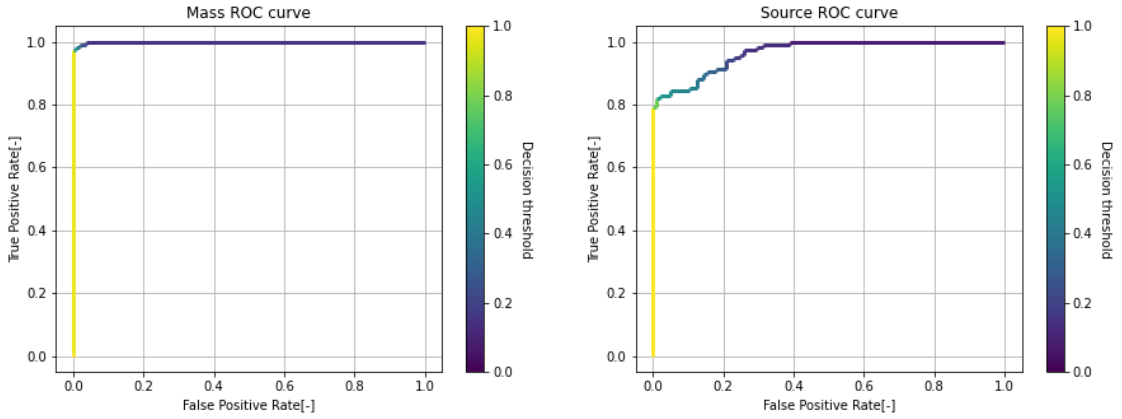
Fig. 16: Final ROC curves over different CNNs. Samples :10000 - Batch size : 64 - Training set : 85% - Data : Table III



(a) Random Forrest ROC. Mass AUCROC : 1.000 - Source AUROC : 0.971 - Without rounding



(b) Naive Bayes ROC. Mass AUCROC : 0.994 - Source AUROC : 0.965 - Without rounding



(c) Ridge ROC. Mass AUCROC : 1.000 - Source AUROC : 0.969 - Without rounding

Fig. 17: Final ROC curves over different blending methods. Samples :1500 - Training set : 85% - Data : Table III

method. Supervised machine learning methods learn from the given data and provide new observations.

a) Random Forest: The random forest uses uncorrelated decision trees where each predicts a class with a given data[22]. The majority voting indicates the category by using trees' predictions. It describes the bagging of several uncorrelated decision trees. The main advantage of this approach is that it is simple to implement, and it is less sensitive to variance variation due to the low correlation between models.

b) Naive Bayes: Naive Bayes focuses on a conditional probability for a given class C_k[23]. Thus, with given features $\boldsymbol{x} = (x_1, ..., x_n)$, the classifier is able to give a prediction on the belonging to a given class. It is recognized as 'Naive', since it consider each feature independently.

$$p(C_k|\boldsymbol{x}) = \frac{p(C_k)p(\boldsymbol{x}|C_k)}{p(\boldsymbol{x})} \tag{15}$$

Thus, the prediction returns the maximum probability for a given value $\boldsymbol{x}$.

$$\hat{y} = \underset{k\in\{1,..,K\}}{argmax}\, p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{16}$$

One advantage of the approach is that it requires a small amount of data and is extremely fast to train compared to other classifiers. Nevertheless, due to its 'Naive' interpretation, it classifies poorly compare to other classification tasks.

c) Ridge regression: Ridge regression consist in optimizing weights between different model's predictions combined with a L2 regularization term such that :

$$\min_{w} \|y - Xw\|_2^2 + \alpha \|w\|_2^2 \tag{17}$$

Where y is the label, X the models' predictions, w the weight vector of models and α a hyperparameter that should be optimized. The L2 regularization limits the overfitting effect and helps us find more general results.

C. Experimental method

The blending method consists of taking multiple models' output and combining them to make the final prediction. We do the combination as follows:

1) Selected models are trained on the train set (Samples: 10000 - Training set: 85%).
2) Predictions are made on the test set (Samples: 10000 - Test set: 15%). The mass and source labels are separated for binary relevance.
3) Random Forrest's maximum depth and features as long as the number of estimation are tested through 10-fold cross-validation on the whole test set and are tested iteratively over different parameter variations(Table XI).
4) Naive Bayes does not require any parameters.
5) Ridge regression's regularization parameter α is determined by 10-fold cross-validation on the whole test set (Table XII).
6) Classifiers are fitted with 85% of the predictions on the test set (Samples: 1275). A final prediction is then performed over the remaining test set (Samples: 225).

For future applications, selected models should be trained on the whole data as long as the classifiers predict labels of an unknown dataset.

TABLE XI: Random tree parameters over 10-fold. Samples: 1500

Mass parameters		**Source parameters**	
Maximum depth	5	Maximum depth	13
Number estimator	38	Number estimator	86
Maximum features	2	Maximum features	2

TABLE XII: Weights computed through the ridge regression with a cross validation over 10-fold. Samples: 1500 - Best alpha mass : $\alpha = 151.74$ - Best alpha source : $\alpha = 1.02$

Model mass weight		**Model source weight**	
BasicCNN	0.161	BasicCNN	0.660
AlexNet	0.141	AlexNet	-0.157
VGG11	0.156	VGG11	0.192
GoogleNet	0.161	GoogleNet	0.200
ResNet18	0.142	ResNet18	-0.023
DenseNet121	0.153	DenseNet121	0.170
SqueezeNet	0.062	SqueezeNet	0.037

D. Blending results

Firstly, we observe in Table XIII that our final classifiers reach similar or better results than our baseline model. We also see in Figures 17b and 17c that differentiation is more obvious than previous classifiers. The profile of the ROC curve looks like a staircase in Figure 17b and 17c which signify a lack of samples in our classifier. Just as previously, mass errors are easier to detect compare to source errors.

In Figure 18, we remark that the three blended models lean-to predict well mass error and source errors. Though, for mass and source errors, classifiers are inclined to classify them as mass errors. It meets our previous observation on the source since at least less than one-quarter of the dataset provides false predictions over the source label.

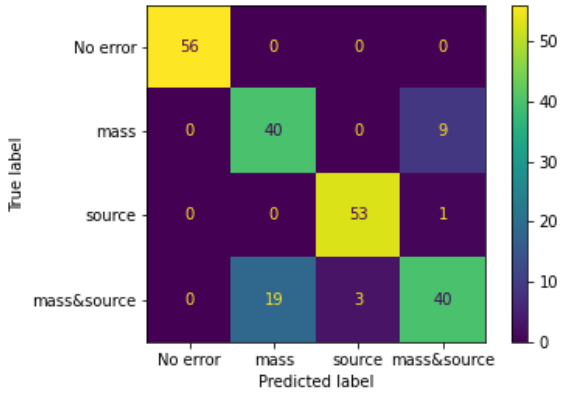
In Figure 19, residual maps seem to be predominant compare to source profile. This problem may come from the way we build the dataset. Mass and source errors have the same percentage, in that case, $p_{m\&s} = 0.5\%$, which suggest a strong parameters' variation of the mass with a small parameters' variation of the source. Since the chi-square test is always passed due to the mass profile, using the same configuration for the source profile as in the source profile dataset should enhance the classification, such that for $p_{m\&s} = \begin{bmatrix} p_m & p_s \end{bmatrix} = \begin{bmatrix} 0.005 & 0.015 \end{bmatrix}$.

Our final classifier is thus Ridge regression classifier with a prediction AUROC of 0.950.

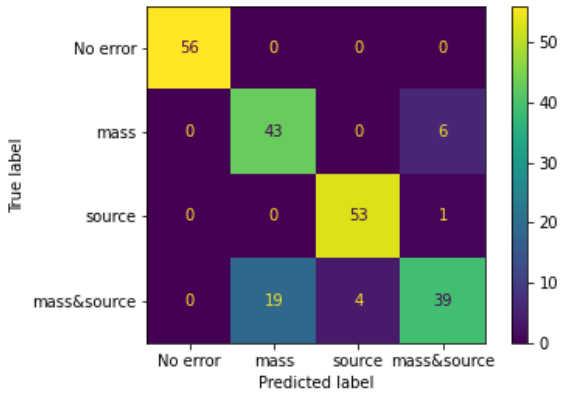
XII. Challenges & Future work

A. Dataset

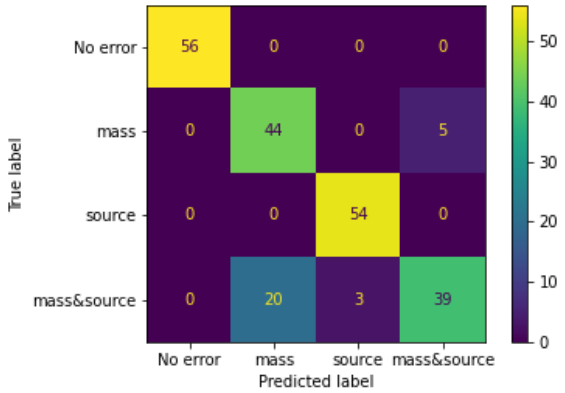
a) Number of samples: For future application, increasing the number of samples should bring more accurate and general predictions over the data set. However, larger datasets engender larger training time and memory usage.



(a) Random Forrest confusion matrix. The numbers correspond to the number of samples



(b) Naive Bayes confusion matrix. The numbers correspond to the number of samples.



(c) Ridge regression confusion matrix. The numbers correspond to the number of samples.

Fig. 18: Confusion matrix over different blending classifiers.

TABLE XIII: Results on various ensemble approaches after 10-fold cross validation. Samples : 1500 - Batch size : 64 - Training set : 85% - Data : Table III - Rounding: {0,1}

Model	Ridge	RandomForest	Naive Bayes
Test AUROC	0.950	0.950	0.947
Mass AUROC	0.990	0.990	0.994
Source AUROC	0.909	0.907	0.903

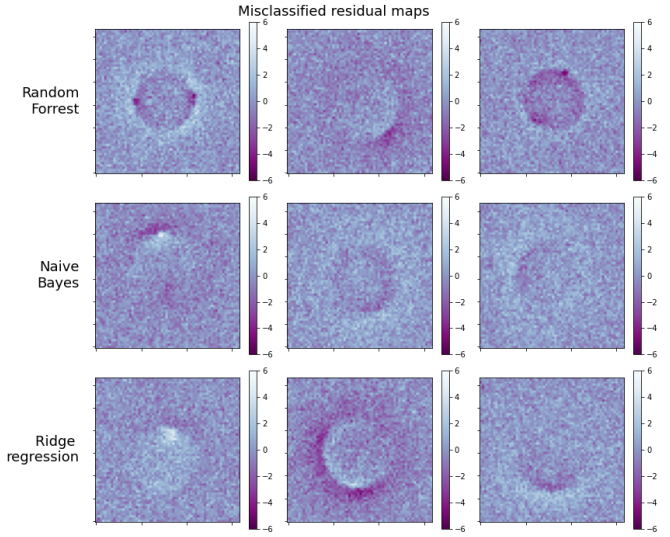


Fig. 19: Misclassified residual maps. It corresponds to mass and source error residual maps predicted as mass error residual maps

b) Errors: Two types of error are implemented in this project. We would expect for future implementation a wider set of error labels. A first step can be done by adding a lens light profile to our classifier. Then, we can think about adding an error label specific to each parameter.

c) Percentage of error: In our analysis, we focus on a different percentage set over each combination of errors. Thus, mass and source parameters' error percentages are the same, which misguide the classification on source and mass error combination. A way to solve this problem is to keep the same combination of parameters as for single label errors, such that $p_{m\&s} = \begin{bmatrix} p_m & p_s \end{bmatrix} = \begin{bmatrix} 0.005 & 0.015 \end{bmatrix}$ in our configuration.

d) Source model: The source label is hardly detected in general by convolutional neural networks due to the prominence of the mass profile. In our configuration, we consider that Sersic profile rules simulated data images and models. By experience, we know that distant galaxies show some irregularities and have more complex source profiles, while closer galaxies have a more smooth definition that matches better the Sersic Profile definition. In this framework, the classifier should handle model mismatch combined with parameters mismatch.

As irregularities and parameters mismatch is present, the classifier should detect source errors more easily. Moreover, real data usage should enhance the classifier for future exper-

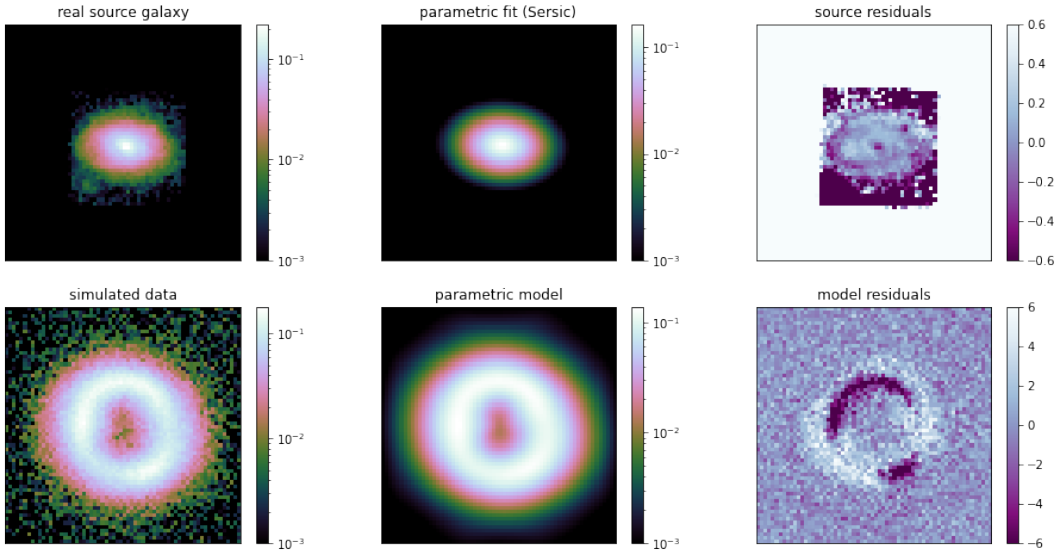


Fig. 20: Comparison between residual maps generated from real source galaxies and simulated data with a Sersic model

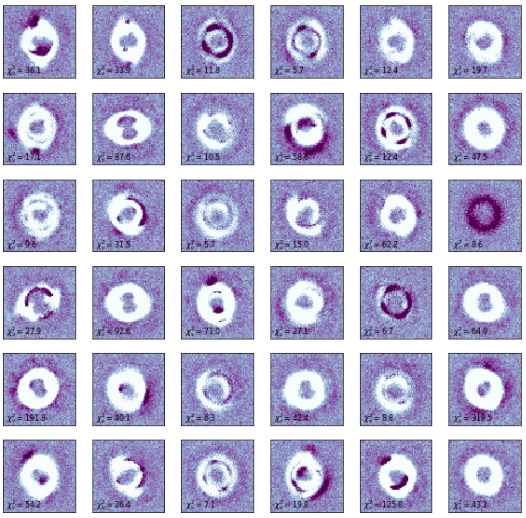


Fig. 21: Sample of residual maps of real simulated data and a Sersic profile model with the same parameters. The colour bar bounds are $[-6, 6]$

imental applications. In Figure 20, we observe that even with the same parameters, the model mismatch is more obvious compare to what we would expect from a residual map without error over parameters. Error in the Sersic simulated data is smoother and has less variation on the ring than the real source galaxy's residual map. In Figure 21, we illustrate a sample of the final data set residuals with model and parameters' mismatch.

The deformation of source arcs is also linked to the definition of the mass profile, as expressed in the Equation (3). In most cases, we assume the PEMD model as a good approximation of the mass profile, which permits us to suspect a problem in the definition of the Sersic profile. Nevertheless, the source and mass model are probably both more complex compared to those models. A deeper analysis of real images and generated models should provide better insight into source and mass model selection.

B. Metric

a) Cross validation: Models' performance estimation for CNNs is done on a fixed training and validation set, making our estimation specific to this dataset. Implementing a k-fold for each model should bring a more general idea of our models' performance.

b) Rounding: When a label is unknown by a classifier, the prediction lean to be equal to 0.5, which in our configuration is rounded necessarily to zero or one. Although, this brings an additional error to our final classification as it increases the misclassification. One solution can be to add the label 0.5 in the rounding process with a customized threshold for each label, as a threshold of 0.25 is probably inadequate.

C. CNNs' recommandations

After studying widely used CNNs, we can deduce several recommendations for future models:

1) We must stick to simple convolutional neural networks to avoid training issues. Moreover, neural networks adapted to a 64×64 input image should add an interesting effect on the final classification.
2) We should consider skipping connections as it tackles vanishing gradient and degradation.
3) We should select optimizers wisely and find a good trade-off between fast and accurate optimizer such as Adam.
4) We only used in our analysis ReLU activation function, yet this activation is less used nowadays since it carries a vanishing gradient during the training. Since we want to keep the non-linearity and reduce this effect, we should favour an activation function such as SeLU.
5) We must analyze neural networks' training time on larger scales to have more significant results.

D. Blending algorithms

a) Models: Increasing the number of uncorrelated models should increase the performance of our final classifier drastically.

b) Problem transformation: Binary relevance does not take into account the correlation between labels, which may increase the final classifier results. Several approaches are possible, such as classifier chains or label power set.

XIII. Conclusion

In conclusion, the initial dataset's properties are key to provide an accurate and realistic classification. We observe that noise-like, obvious error residual maps and badly chosen error percentages bring undesired behaviour in our neural network training, with unrealistic performance throughout our analysis. Our examination over several convolutional neural networks let us believe that we should stick to simpler models and be careful on training features and architectures. Finally, blending models refine our predictions as expected and be applied for future models with a larger set of data and models. Our final result remains promising as our final classifier can detect perfectly separated error labels and residual maps that do not contain errors. Thus, a closer consideration of error percentage and realistic galaxy images should bring a more accurate automated lens modelling pipeline.

Acknowledgements

I thank my supervisors, PhD student Galan Aymeric and PhD Peel Austin, for their consistent guidance and feedback throughout this project. I also thank Pr. Courbin Frederic who made this project possible.

References

Sources verified on June 11, 2021:

[1] Simona Vegetti & L. V. E. Koopmans, *Bayesian strong gravitational-lens modelling on adaptive grids: objective detection of mass substructure in galaxies*, Kapteyn Astronomical Institute, University of Groningen, Oct 2018, https://arxiv.org/pdf/0805.0201.pdf

[2] A. J. Shajib & al., *Is every strong lens model unhappy in its own way? Uniform modelling of a sample of 13 quadruply+ imaged quasars*, Jul 2018, https://arxiv.org/pdf/1807.09278.pdf

[3] F. Cyr-Racine, C. R. Keeton, & L. A. Moustakas, *Beyond subhalos: Probing the collective effect of the Universe's small-scale structure with gravitational lensing*, Aug 2019, https://arxiv.org/pdf/1806.07897.pdf

[4] Simon Birrer & Adam Amara, *lenstronomy - gravitational lensing software package*, 2018, https://github.com/sibirrer/lenstronomy

[5] Adam G. Riess & al., *A 2.4% Determination of the Local Value of the Hubble Constant*, Jun 2016, https://arxiv.org/pdf/1604.01424.pdf

[6] N. Aghanim & al., *Planck 2018 results. VI. Cosmological parameters*, Sep 2018, https://arxiv.org/pdf/1807.06209.pdf

[7] X. Ding & al., *Time Delay Lens Modelling Challenge*, Feb 2021, https://arxiv.org/pdf/2006.08619.pdf

[8] S. Wagner-Carena & al., *Hierarchical Inference With Bayesian Neural Networks: An Application to Strong Gravitational Lensing*, Mar 2021, https://arxiv.org/pdf/2010.13787.pdf

[9] X. Ding & al., *Stellar systems following the R1/m luminosity law.*, Sep 1991, vol 249, pages 99-106 https://ui.adsabs.harvard.edu/abs/1991A&A...249...99C

[10] Rebecca Stone, *Three Ways of Storing and Accessing Lots of Images in Python*, https://realpython.com/storing-images-in-python/

[11] Mohammad S. Sorower, *A literature survey on algorithms for multi-label learning*, 2010 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.364.5612&rep=rep1&type=pdf

[12] Diederik P. Kingma & Jimmy Ba, *On the importance of initialization and momentum in deep learning*, http://www.cs.toronto.edu/~hinton/absps/momentum.pdf

[13] Diederik P. Kingma & Jimmy Ba, *Adam: A Method for Stochastic Optimization*, Jan 2017, https://arxiv.org/pdf/1412.6980.pdf

[14] Mahdi Hashemi, *Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation*, 2019, https://journalofbigdata.springeropen.com/track/pdf/10.1186/s40537-019-0263-7.pdf

[15] Forrest N. Iandola & al., *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size*, Nov 2016, https://arxiv.org/pdf/1602.07360.pdf

[16] K. He & al., *Deep Residual Learning for Image Recognition*, Dec 2015, https://arxiv.org/pdf/1512.03385.pdf

[17] A. Krizhevsky & al., *ImageNet Classification with Deep Convolutional Neural Networks*, https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[18] K. Simonyan & al., *Very Deep Convolutional Networks for Large-Scale Image Recognition*, Sep 2014 https://arxiv.org/pdf/1409.1556.pdf

[19] C. Szegedy & al., *Going Deeper with Convolutions*, Sep 2014 https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43022.pdf

[20] G. Huang & al., *Densely Connected Convolutional Networks*, Jan 2018 https://arxiv.org/pdf/1608.06993.pdf

[21] M. Zhang & al., *Binary relevance for multi-label learning: an overview*, 2018, https://link.springer.com/content/pdf/10.1007/s11704-017-7031-7.pdf

[22] Tony Yiu, *Understanding Random Forest*, Jun 2019, https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[23] H. Zhang, *The Optimality of Naive Bayes*, FLAIRS, 2004, https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

APPENDIX

XIV. DEEPLENSTRONOMY PARAMETRIZATION

XV. MAIN NEURAL NETWORK STRUCTURE

A. Metrics

a) Exact Match ratio: The exact match ratio ignores the partially correct values (considered as incorrect) and compute the resulting accuracy as in a single label classification. The main issue with this approach is that it does not consider partially correct values. An accurate prediction over two classes instead of three is always better than none in multi-label classification. The function I is the indicator function, and the index i stands for the i-th datum.

$$MR = \frac{1}{N}\sum_{i=1}^{N} I(x_i, y_i) \tag{18}$$

b) Hamming Loss: The hamming loss compute the fraction of misclassified labels for each class. Thus, for several L classes and N samples, the hamming loss is equal to :

$$HL = \frac{1}{NL}\sum_{l=1}^{L}\sum_{i=1}^{N} y_i^l \oplus x_i^l \tag{19}$$

The hamming loss is not recommended for imbalanced dataset as a model that assign only one label have a low hamming loss by definition. Finally, in binary cases the hamming loss is related to the accuracy such that : $HL = 1 - Ac$.

c) Accuracy: The accuracy of each class is computed and then averaged. We describe accuracy as the quantity of correct prediction divided by the overall amount of samples. Just as the hamming loss, accuracy is not recommended in imbalanced data sets.

$$Ac = \frac{1}{NL}\sum_{l=1}^{L}\sum_{i=1}^{N} I(x_i^l, y_i^l) \tag{20}$$

d) Recall, precision & F1-score: Recall identifies the fraction of true positive(TP) on the overall positive predictions, while the precision computes the ratio of true positive on the general, correct predictions.

$$\text{Recall} = \frac{TP}{FP + TP} \tag{21}$$

$$\text{Precision} = \frac{TP}{FN + TP} \tag{22}$$

The F1-score finally combines recall and precision through a harmonic mean. The main advantage of those three metrics is that they give a significant indication of imbalanced dataset classification.

$$F1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{23}$$

B. Optimizers

XVI. BASELINE IMPLEMENTATION

XVII. CONVOLUTIONNAL NEURAL NETWORK

TABLE XIV: Choice of the threshold ϵ for Adam optimizer. Learning rate : $lr = 0.001$ - Exponential decay: $\beta_1 = 0.9$ $\beta_2 = 0.99$[13].

Threshold	AUC
$\epsilon = 10^{-8}$	0.68
$\epsilon = 0.01$	0.833
$\epsilon = 0.1$	0.846
$\epsilon = 1$	0.667

TABLE XV: Basic convolutional neural network with only the residual maps. Activation function : SeLU - Loss function : Binary cross entropy

Layer name	Output Size	2-layer
conv1	$60 \times 60 \times 6$	5×5, stride 1
Max pool	$30 \times 30 \times 6$	2×2 max pool, stride 2
conv2	$26 \times 26 \times 16$	5×5, stride 1
Max pool	$13 \times 13 \times 16$	2×2 max pool, stride 2
Fully connected	120	$13 \times 13 \times 16 \times 120$ fully connected
	84	120×84
Sigmoid	2	

TABLE XVI: Basic linear neural network with only the metadata. Activation function : SeLU - Loss function : Binary cross entropy

Layer name	Output Size
lin1	16
lin2	8
Sigmoid	2

TABLE XVII: Basic linear neural network with metadata and residual maps. Activation function : SeLU - Loss function : Binary cross entropy

(a) Residual map convolutional neural network

Layer name	Output Size	2-layer
conv1	$60 \times 60 \times 6$	5×5, stride 1
max pool	$30 \times 30 \times 6$	2×2 max pool, stride 2
conv2	$26 \times 26 \times 16$	5×5, stride 1
max pool	$13 \times 13 \times 16$	2×2 max pool, stride 2
fully connected	120	$13 \times 13 \times 16 \times 120$ fully connected
	84	120×84

(b) Metadata linear neural network

Layer name	Output Size
lin1	16
lin2	8

(c) After the concatenation of the metadata and residual maps outputs

Layer name	Output Size
lin1	60
Sigmoid	2

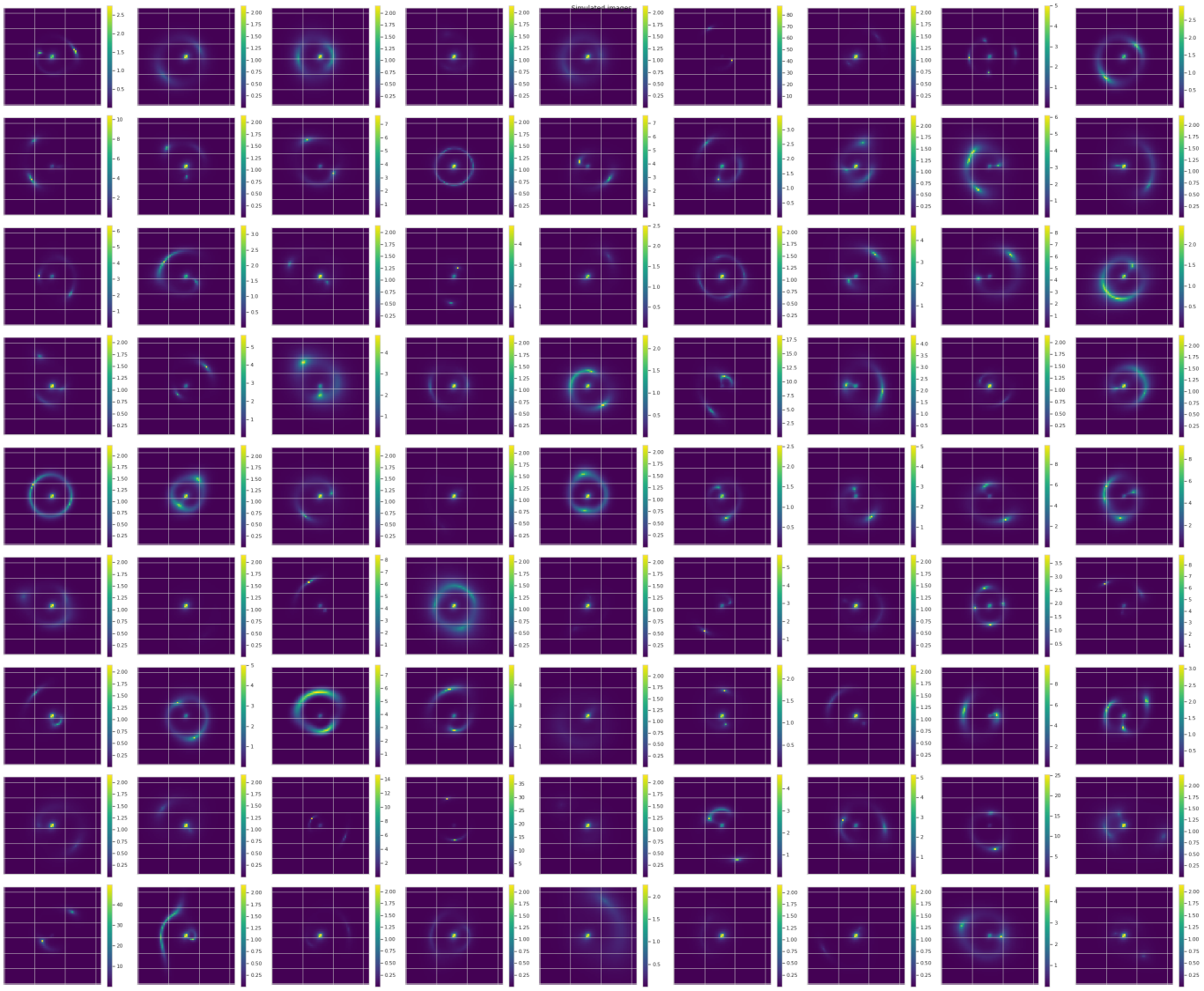


Fig. 22: Simulated data set for the described distribution in Table I with deeplenstronomy

TABLE XVIII: AlexNet neural network structure for residual maps[17]. Activation function : ReLU - Loss function : Binary cross entropy

Layer name	**Output Size**	**11-layer**
conv1	$55 \times 55 \times 96$	11×11, stride 4
conv2	$27 \times 27 \times 256$	3×3 max pool, stride 2
		5×5, stride 1
conv3	$13 \times 13 \times 384$	3×3 max pool, stride 2
		$[3 \times 3$, stride $1] \times 2$
conv4	$13 \times 13 \times 256$	3×3, stride 1
max pool	$6 \times 6 \times 256$	3×3 max pool, stride 2
fully connected	4096	$6 \times 6 \times 256 \times 4096$ fully connected
	4096	4096×4096
sigmoid	2	

TABLE XIX: VGG11 neural network structure for residual maps[18]. Activation function : ReLU - Loss function : Binary cross entropy

Layer name	**Output Size**	**11-layer**
conv1	$224 \times 224 \times 64$	3×3, stride 1
max pool	$112 \times 112 \times 64$	2×2 max pool, stride 2
conv2	$112 \times 112 \times 128$	3×3, stride 1
max pool	$56 \times 56 \times 128$	2×2 max pool, stride 2
conv3	$56 \times 56 \times 256$	$[3 \times 3$, stride $1] \times 2$
max pool	$28 \times 28 \times 256$	2×2 max pool, stride 2
conv4	$28 \times 28 \times 256$	$[3 \times 3$, stride $1] \times 2$
conv5	$14 \times 14 \times 512$	2×2 max pool, stride 2
		$[3 \times 3$, stride $1] \times 2$
max pool	$7 \times 7 \times 512$	2×2 max pool, stride 2
fully connected	4096	$7 \times 7 \times 512 \times 4096$ fully connected
	4096	4096×4096 fully connected
sigmoid	2	

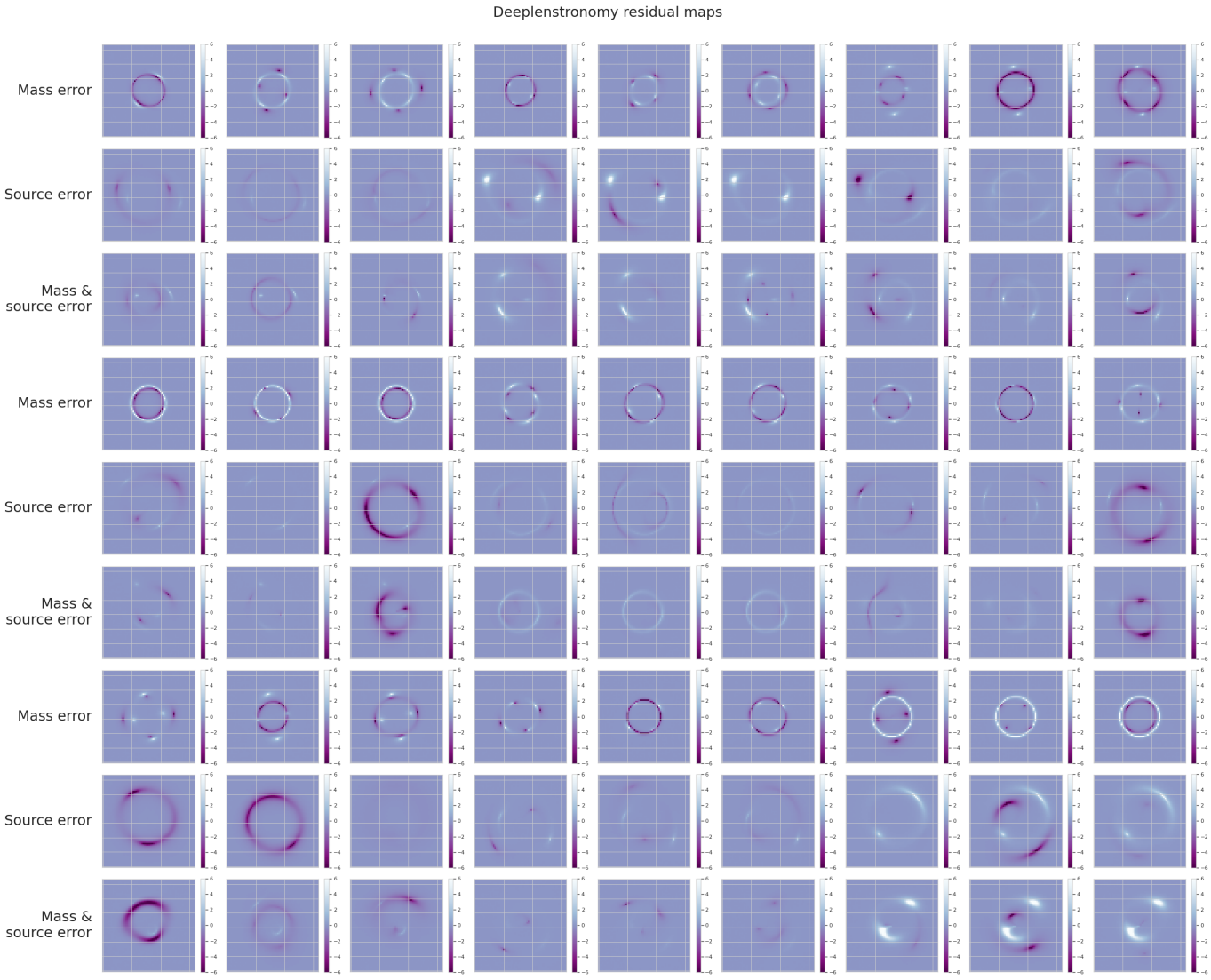


Fig. 23: Dataset generated with deeplenstronomy. The residuals are combination of strictly different models.

TABLE XX: GoogleNet neural network structure for residual maps[19]. Activation function : ReLU - Loss function : Binary cross entropy

Layer name	Output Size	21-layer	1×1	3×3	5×5	max pool
conv1	$112 \times 112 \times 64$	7×7, stride 2				
max pool	$56 \times 56 \times 64$	3×3, stride 2				
conv2	$56 \times 56 \times 192$	3×3, stride 1		192 red:64		
max pool	$28 \times 28 \times 192$	3×3, stride 2				
inception	$28 \times 28 \times 256$		64	128 red:96	32 red:16	32
	$28 \times 28 \times 480$		128	192 red:128	96 red:32	64
max pool	$14 \times 14 \times 480$	3×3, stride 2				
inception	$14 \times 14 \times 480$		192	208 red:96	48 red:16	64
	$14 \times 14 \times 480$		160	224 red:112	64 red:24	64
	$14 \times 14 \times 480$		128	256 red:128	64 red:24	64
	$14 \times 14 \times 480$		112	288 red:144	64 red:32	64
	$14 \times 14 \times 480$		256	320 red:160	128 red:32	128
max pool	$7 \times 7 \times 832$					
inception	$7 \times 7 \times 832$		256	320 red:160	128 red:32	128
	$7 \times 7 \times 1024$		384	384 red:192	128 red:48	128
average pool	$1 \times 1 \times 1024$	7×7, stride 1				
linear	1000	1024×1000 dropout 40%				
sigmoid	2					

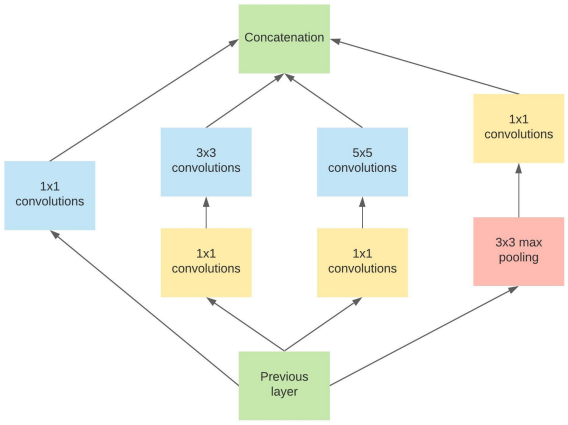


Fig. 24: Inception neural network structure with dimensions reductions in GoogleNet [19].

TABLE XXI: ResNet18 neural network structure for residual maps. conv_x correspond to residual blocks[16]. Activation function : ReLU - Loss function : Binary cross entropy

Layer name	Output Size	18-layer
conv1	$112 \times 112 \times 64$	7×7, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, stride 1 $\times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, stride 1 $\times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, stride 1 $\times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3 \\ 3 \times 3 \end{bmatrix}$, stride 1 $\times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connected
sigmoid	2	

TABLE XXII: DenseNet121 neural network structure for residual maps[20]. Activation function : ReLU - Loss function : Binary cross entropy

Layer name	Output size	DenseNet-121
conv	$112 \times 112 \times 64$	7×7,stride 2
max pool	$56 \times 56 \times 64$	3×3 max pool,stride 2
dense_1	$56 \times 56 \times 256$	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix}$, stride 1×6
transition_1	$56 \times 56 \times 128$	1×1, stride 1
	$28 \times 28 \times 128$	2×2 average pool, stride 2
dense_2	$28 \times 28 \times 512$	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix}$, stride 1×12
transition_2	$28 \times 28 \times 256$	1×1, stride 1
	$14 \times 14 \times 256$	2×2 average pool, stride 2
dense_3	$14 \times 14 \times 1024$	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix}$, stride 1×24
transition_3	$14 \times 14 \times 512$	1×1, stride 1
	$7 \times 7 \times 512$	2×2 average pool, stride 2
dense_4	$7 \times 7 \times 1024$	$\begin{bmatrix} 1 \times 1 \\ 3 \times 3 \end{bmatrix}$, stride 1×16
average pool	$1 \times 1 \times 1024$	7×7 average pool, stride 2
fully connected	1000	1024×1000 fully connected
sigmoid	2	

TABLE XXIII: SqueezeNet neural network structure for residual maps[15]. Activation function : ReLU - Loss function : Binary cross entropy

Layer name	Output Size	18-layer	1×1 squeeze	1×1 expand	3×3 expand
conv1	$111 \times 111 \times 64$	3×3, stride 2			
max pool	$55 \times 55 \times 64$	3×3, stride 2			
fire	$55 \times 55 \times 128$		16	64	64
	$55 \times 55 \times 128$		16	64	64
max pool	$27 \times 27 \times 128$	3×3, stride 2			
fire	$27 \times 27 \times 256$		32	128	128
	$27 \times 27 \times 256$		32	128	128
max pool	$13 \times 13 \times 256$	3×3, stride 2			
fire	$13 \times 13 \times 384$		48	192	192
	$13 \times 13 \times 384$		48	192	192
	$13 \times 13 \times 512$		64	256	256
	$13 \times 13 \times 512$		64	256	256
conv2	$13 \times 13 \times 2$	1×1, stride 1 dropout 50%			
average pool	$1 \times 3 \times 2$				
sigmoid	2				

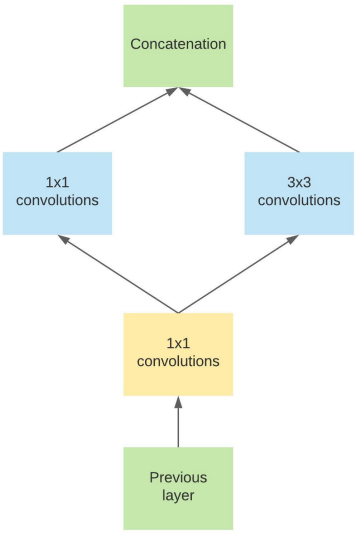


Fig. 25: Fire neural network structure with dimensions reductions in SqueezeNet [15].