Event Horizon Recognition through Quasar Microlensing Lightcurves using Deep Learning

Master Thesis in Physics

Center for Theoretical Astrophysics and Cosmology
Institute for Computational Science
University of Zurich

Shiva Farghar

Supervisor
Prof. Dr. Prasenjit Saha,

November 2021

Abstract

Gravitational microlensing of quasars enables to probe its innermost region, i.e. a supermassive black hole surrounded by an accretion disk, at very small scales inaccessible by other methods. When a quasar is lensed by a galaxy, multiple images are produced by the galaxy as a whole. These multiple images show brightness changes, which are associated with the central engine of the quasar traversing a complicated network of caustics produced by the stars in the lensing galaxy. In this work, it is investigated whether deep learning may be utilised successfully for the recognition of the event horizon through these brightness variations. The goal is to present a neural network model for the classification of brightness changes corresponding to different sources given by simple geometric models. The silhouette of a black hole over an accretion disk is represented by a crescent-shaped source. The neural network model that yields the most accurate performance is a convolutional neural network model. With this model, even when considering a more realistic lensing galaxy macromodel and number of lensing stars and by including random Gaussian noise, a very high average accuracy is achieved, i.e. (99.30 ± 0.18)%. Moreover, it is sensitive to the crescent parameters. These crescent parameters provide an estimate for the size of the accretion disk and the gravitationally magnified Schwarzschild radius.

Contents

1 Introduction

1.1 Quasars and Active Galactic Nuclei

Quasars, for *quasi-stellar radio sources*, are active galactic nuclei (*AGN* for short) according to the AGN unification model (e.g., Antonucci 1993; Urry & Padovani 1995). They contain a supermassive black hole, i.e. a black hole with a mass above 10^6 solar masses, which is surrounded by an accretion disk (e.g., Kembhavi & Narlikar 1999; D'Onofrio et al. 2012). The size of the accretion disk is typically less than 0.01 pc (Morgan et al. 2010), making quasars compact objects.

AGNs are thought to be powered by the accretion of matter from the surrounding disk falling into the supermassive black hole. The gravitational energy of the infalling material is radiated away as electromagnetic radiation. In the work of Shakura & Sunyaev (1973), a simple thin-disk model of the viscous material in the accretion disk was presented, which is inspiraling and radiates the released energy. A particle in an accretion disk moving on a geodesic will lose energy due to friction and move in a spiral into smaller and smaller radii. Approximately, the particle can be thought of as slowly moving from one stable circular orbit to the next, eventually reaching the ISCO (short for: *innermost stable circular orbit*) and falling into the supermassive black hole (e.g., Reall 2020). For this process, the fraction of rest mass that is converted into radiation is $\eta_{acc} \approx 0.06$, which is much higher than, for example, the fraction of rest mass energy released in nuclear fusion $\eta_{fusion} \approx 7 \times 10^{-4}$ (e.g., Frank et al. 2002). Thus, AGNs, especially quasars, are one of the most energetic phenomena in the universe.

As of August 2020, more than 750'000 quasars have been discovered, with the great majority of them at redshifts $z > 2$ (Pâris et al. 2014; Lyke et al. 2020). Thanks to their high luminosity, quasars can be detected despite the long distance. Quasars found at redshifts $6 < z < 7$ are powered by supermassive black holes up to 10^{10} solar masses (Wu et al. 2015).

1.2 Quasar Microlensing as a Way to probe the Structure of the Central Engine

Quasar microlensing enables the probe of supermassive black hole properties and surrounding accretion disk structures in quasars. This would otherwise not be accessible due to their compactness and the large distances (e.g., Schmidt & Wambsganss 2010). The theory of gravitational microlensing relevant for the scope of the present thesis is summarised in chapter 2. Microlensing considers the case of gravitational lensing where the separation of the resulting multiple images is far below the limiting resolution given by observational constraints. Gravitational lensing of quasars, discussed in section 2.4, occurs in two regimes. On the one hand, the galaxy acts as a lens in strong lensing regime and produces multiple images. On the other hand, individual stars inside the galaxy act as microlenses, which results in brightness changes of the multiple images providing the *quasar microlensing lightcurves* (e.g., Wambsganss 1999; Courbin et al. 2002).

From the quasar lightcurves physical information can be extracted (e.g., Kochanek 2004). The usual approach is as follows. Lightcurves are simulated for different parameters of the used model and fitted to the data (e.g., Poindexter et al. 2008; Poindexter & Kochanek 2010). For the simulation of lightcurves, a common technique is to use a *magnification map*. A magnification map represents caustic networks in the source plane (see section 2.3.2) given by the macromodel of the lensing galaxy and the number of individual lensing stars. By using the code of Wambsganss (1999) described in section 4.2, a magnification map can be generated.

Kamruddin & Dexter (2013) introduced a simple geometric crescent model and argued that the crescent represents the silhouette of the *event horizon*. Considering a Schwarzschild blackhole with a mass M, the event horizon, defined by its *Schwarzschild radius*: $R_S \equiv 2GMc^{-2}$, gives a boundary in spacetime beyond which events cannot affect observers, i.e. a one-sided causal boundary from which not even light can escape (Schwarzschild 1916). In the work of Tomozeiu et al. (2017), an indirect method, which is related to Agol & Krolik (1999) and Mediavilla et al. (2015), was used to examine the black hole shadow and its vicinity inside a quasar. They presented a model to probe the event horizon structure in quasars from lightcurves obtained by considering a crescent-shaped source.

1.3 Expanding the Possibilities thanks to Machine Learning and Deep Learning

With regard to possible methods for reconstructing parameters characteristic of the structure of quasars from lightcurves of multiple microlensing events, *big data* will play an important role. By analysing large amounts of data, new tools can be developed and new insights gained. Machine learning and deep learning provide promising solutions when dealing with big data, in particular for the classification of data. The basic concepts for classifications with machine learning and deep learning are explained in chapter 3.

1.4 The Purpose of the present Work

The goal of this work is to present a neural network model for the recognition of quasar microlensing lightcurves representing the event horizon. For this purpose, a classification task is defined where lightcurves, given by discrete time-dependent signals, are assigned to three different sources. For the three sources, simple geometric models with constant surface brightness are considered. The three source shapes are a *disk*, a simple geometric *crescent* model from Kamruddin & Dexter (2013), and a *random* source. The first two models are implemented in a similar way as in the work of Tomozeiu et al. (2017). For the latter, four different subtypes of randomly shaped sources, which have no physical meaning, are introduced. Since the training of a neural network requires large amounts of data, an algorithm is developed to generate automatically a large number of lightcurves for a given magnification map. Using this algorithm two datasets for two different magnification maps are provided, where the lightcurves of one dataset additionally include random Gaussian noise representing observational noise. The simulation of the lightcurves and the written algorithm are described in chapter 4. For each dataset, a classification is performed with an *Adaptive Boosting Classifier* (see section 3.3.2) and with three different neural network models. The approach for the classification of the lightcurves and a description of the different neural network models are given in chapter 5. The results are presented in chapter 6 and discussed in chapter 7. The conclusion can be found in chapter 8.

2 Microlensing

According to the Einstein's General Theory of Relativity, photons travel on the null geodesics of the spacetime metric. All matter between an observer and a light source affects the path, size, and cross section of a light bundle propagating through spacetime. Light emitted by a source gets deflected in the neighbourhood of the lens. In Figure 1, the configuration for a gravitational lensing event is illustrated. In the regime of *gravitational strong lensing*, multiple images of the source or an *Einstein ring* (section 2.2) can be observed. *Gravitational microlensing* considers the case of gravitational lensing where the image separation of the resulting multiple images is far below the limiting resolution given by observational constraints. (e.g., Schneider et al. 2006)

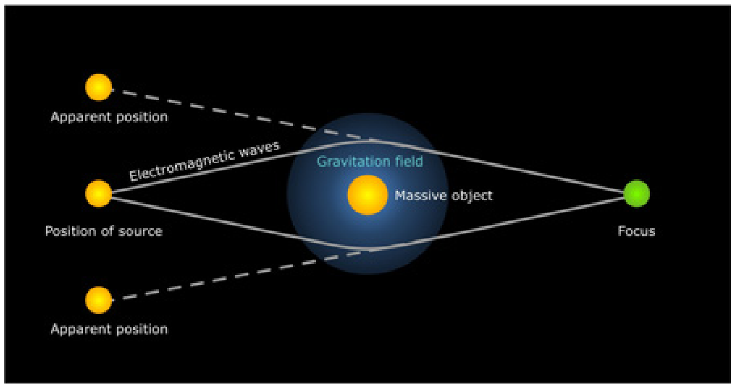


Figure 1: Sketch of a gravitational lensing event, credits: German Aerospace Center, 2009.

In this chapter, the theory of gravitational lensing relevant for microlensing and required for the present work is summarised using Wambsganss (1998), Schneider et al. (1999), Petters et al. (2001), Mollerach & Roulet (2002), Wambsganss (2006), and Schneider et al. (2006).

2.1 The Gravitational Lens Equation

Assuming the *thin lens approximation*, i.e. that the lensing action is dominated by a single inhomogeneity of matter at a single distance between source and observer where the deflection effect takes place, the light rays smoothly curved in the vicinity of the lens can be replaced by two straight rays with a kink. The magnitude and the direction of the kink are determined by the mass distribution of the deflector and the impact vector of the light ray and are described by the deflection angle $\tilde{\alpha}$. Figure 2 shows the geometry of gravitational lensing, where D_L and D_S are the angular diameter distances to the lens and source, θ is the apparent sky position of the source, and β is the true but unobservable position of the source. In figure 3 the source and lens plane in a lensing geometry are illustrated.

In the case of deflection by a point mass, the deflection angle is

$$\tilde{\alpha} = \frac{4GM}{c^2 \xi} \,, \tag{1}$$

where M is the mass of the lens, G is the gravitational constant, c is the speed of light, and ξ is the impact parameter. It is assumed that ξ is much larger than the Schwarzschild radius of the lens, $\xi \gg R_S \equiv 2GMc^{-2}$. Under this condition the deflection angle is small, $\tilde{\alpha} \ll 1$. The validity of the

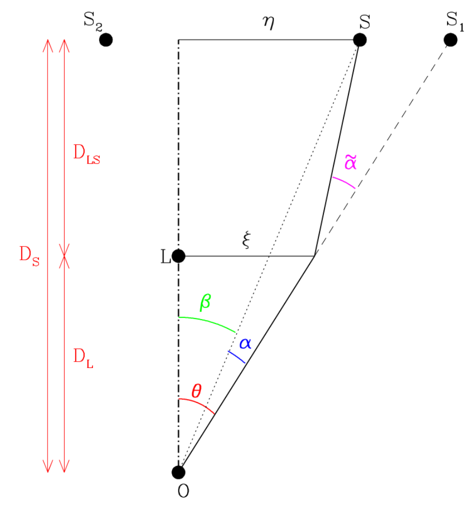


Figure 2: Sketch of gravitational lensing geometry, adopted from Wambsganss (1998).

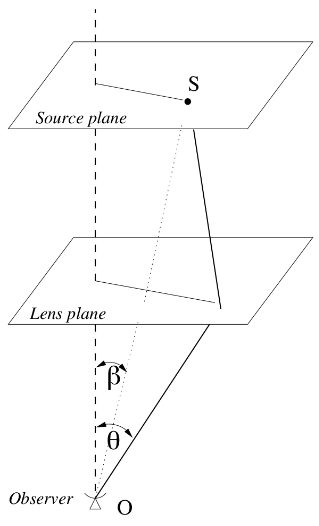


Figure 3: Source and lens plane in a gravitational lensing geometry, from Courbin et al. (2002).

following relation can be seen directily from figure 2:

$$\theta \, D_S = \beta \, D_S + \tilde{\alpha} \, D_{LS} \, . \tag{2}$$

This expression can be rewritten as:

$$\beta = \theta - \frac{D_{LS}}{D_S} \, \tilde{\alpha}(\theta) \, . \tag{3}$$

The reduced deflection angle can be defined as:

$$\alpha = \frac{D_{LS}}{D_S} \, \tilde{\alpha} \, . \tag{4}$$

Generalised to the case of an extended lense with a three-dimensional non-symmetric mass distribution, the two-dimensional deflection angle $\vec{\alpha}$ is given by considering the sum over all mass elements in the lens plane:

$$\vec{\alpha} = \frac{D_{LS} D_L}{D_S} \frac{4G}{c^2} \int \Sigma(\vec{\theta}') \, \frac{\vec{\theta} - \vec{\theta}'}{|\vec{\theta} - \vec{\theta}'|^2} \, d^2\vec{\theta}' \, , \tag{5}$$

where the two-dimensional surface mass density distribution

$$\Sigma(\vec{\theta}) = \int_0^{D_S} \rho(\vec{r}) \, dz \tag{6}$$

is given by the projected density $\rho(\vec{r})$ along the line of sight onto the lens plane. Then, the gravitational lens equation can be written as:

$$\vec{\beta} = \vec{\theta} - \vec{\alpha}(\vec{\theta}) \, . \tag{7}$$

2.2 Einstein Radius

For the special case that observer and source are in perfect alignment, i.e. $|\vec{\beta}| = 0$, a ring shaped image results. The angular radius of this *Einstein ring* is called the *Einstein radius*:

$$\theta_E = \sqrt{\frac{4GM}{c^2} \frac{D_{LS}}{D_L D_S}} \, , \tag{8}$$

and defines the angular scale for a lensing event. In the regime of strong gravitational lensing, the Einstein radius is of the order of *arcsec* and multiple images can be observed. For microlensing θ_E is of the order of *microarcsec*, which is why multiple images cannot be resolved.

2.3 Magnification

2.3.1 Magnification Matrix

The *distortion matrix* is the derivative of the real source position with respect to the apparent source position:

$$\mathbb{D}(\vec{\theta}) = \left(\frac{\partial \beta}{\partial \theta} \right) \, , \tag{9}$$

and can be written as

$$\mathbb{D} = (1 - \kappa) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} \gamma_1 & \gamma_2 \\ \gamma_2 & -\gamma_1 \end{pmatrix} \, , \tag{10}$$

where κ is the convergence and γ_1 and γ_2 are the components of the shear $\vec{\gamma}$. Its inverse is the *magnification matrix*:

$$\mathbb{M}(\vec{\theta}) = \mathbb{D}^{-1} \, . \tag{11}$$

Whether a point source will brighten or dim, i.e. the brightness amplification of its image, is described by the determinant of the magnification matrix:

$$\mu(\vec{\theta}) = det \, |\mathbb{M}(\vec{\theta})| = \frac{1}{(1 - \kappa)^2 - \gamma^2} \, , \tag{12}$$

$$\mu(\vec{\theta}) \begin{cases} > 1 & for \ brightening \\ < 1 & for \ dimming \end{cases} \, . \tag{13}$$

For multiple images $\vec{\theta}_i$ of the same source at $\vec{\beta}$ that are not observationally resolved, there is a total brightness amplification given by

$$\mu_{total} = \sum_i \mu(\vec{\theta}_i) \, . \tag{14}$$

2.3.2 Critical Curves and Caustics

The set of positions on the $\vec{\theta}$-plane where

$$det \, |\mathbb{D}(\vec{\theta})| = 0 \, , \tag{15}$$

lead formally to an infinite magnification. These curves are called *critical curves*. By mapping the critical curves onto the $\vec{\beta}$-plane (source plane), using the lens equation (7), the *caustics* are obtained. In gravitational lensing, caustics can be divided into two categories: *cusp caustics* (points) and *fold caustics* (concave curves). Mathematically these are different orders of catastrophes. The significance of caustics in gravitational lensing of quasars was first presented in Chang and Refsdal (1979). For the simulation of light curves (2.3.4), caustic networks represented by a *magnification map* (Wambsganss 1999) are used and will be discussed in section 4.2.

2.3.3 Magnification near a Caustic

An extensive presentation of the magnification near a caustic can be found in Blandford and Narayan (1986), Schneider and Weiss (1992) , Gaudi and Petters (2001), and Gaudi and Petters (2002). Considering a point source near a caustic, the magnification at distance $r = |\vec{r}|$ from a fold causting is given by

$$\mu(r) = \mu_0 + C_0 \, \frac{1}{\sqrt{r}} \, \Theta(r) \, , \tag{16}$$

where μ_0 is the magnification concerning other effects and can be assumed as locally constant, C_0 is the proportionality constant depending on the constraints in the vicinity of the caustic, and Θ is the Heaviside function. The second term in equation 16 varies inversely with the square root of the distance of the source to the caustic.

2.3.4 Lightcurve of a Microlensing Event

Considering an arbitrary source shape whose centre is positioned at (p_s, q_s) in a two-dimensional coordinate system (p, q), the source can be described by a brightness function $S_{2D}(p - p_s, q - q_s)$. During a microlensing event, the coordinates of the source centre change in time due to the motion of the source. The lightcurve, described by the flux $F(t)$, is given by the convolution of the brightness function describing the source and the magnification near a caustic (equation 16):

$$F(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} S_{2D}(p - p_s(t), q - q_s(t)) \; \mu(r(t)) \; dq \; dp \, , \tag{17}$$

where the caustic is assumed to be fixed. If the coordinate system is chosen such that:

$$\vec{e_q} = \frac{\vec{r}}{r} , \tag{18}$$

μ (equation 16) has no q-dependence. Defining the one-dimensional brightness function of the

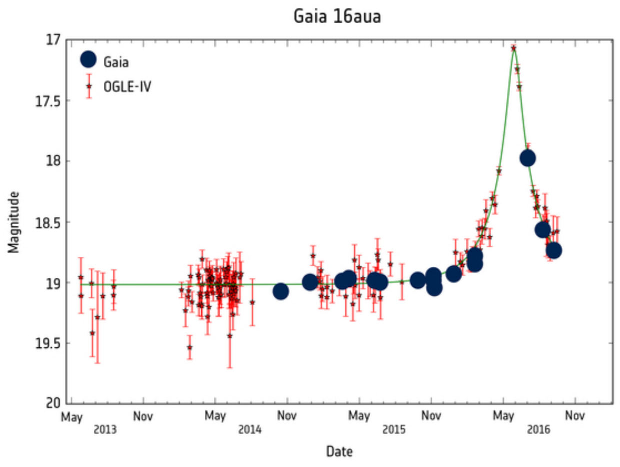


Figure 4: Change in brightness of the star Gaia16aua during a microlensing event, Credit: ESA/Gaia/DPAC, L. Wyrzykowski, OGLE team (Warsaw), Z. Kostrzewa-Rutkowska (SRON/RU).

source by integrating the two-dimensional brightness function over q:

$$S_{1D}(p - p_s(t)) = \int_{-\infty}^{\infty} S_{2D}(p - p_s(t), q - q_s(t)) \; dq \, , \tag{19}$$

equation 17 can be rewritten:

$$F(t) = \int_{-\infty}^{\infty} \mu_t(p) \; S_{1D}(p - p_s(t)) \; dp \, . \tag{20}$$

As an example of an observationally detected lightcurve, figure 4 shows the change in brightness of the star *Gaia16aua* during a microlensing effect, detected by *Gaia*.

2.4 Quasar Lensing

Descriptions of the gravitational lensing of a quasar by a galaxy can be found in several texts, e.g. Wambsganss (1999) and Courbin et al. (2002). When a quasar is lensed by a galaxy, lensing occurs at two mass scales. Figure 5 shows an example of such a lens situation with two images of a quasar. The galaxy as a whole acts as a lens, producing multiple images separated by the order of *arcsec*. Inside the galaxy, the distribution of matter is "granular". Each star in the galaxy acts as a gravitational lens with an Einstein radius of the order of *microarcsec*. As discussed in section 2.2, the multiple micro-images produced cannot be resolved. But microlensing also has an effect on the measured brightness of a quasar image. These changes in brightness are due to the central engine of the quasar traversing a complicated network of caustics generated by the stars in the lensing galaxy. When a quasar crosses a caustic, the brightness changes abruptly, whereas most astrophysical sources straddle multiple caustics and their brightness therefore varies smoothly with location. This is because the innermost region of a quasar, which is the only part of the quasar whose light is affected by microlensing (e.g., Sluse et al. 2012), is smaller than the typical distance between the caustics. Thus, we obtain an upper limit for the size of the central engine of quasars. This effect can be used as well to study the mass distribution and kinematics of stars in the lensing galaxy (e.g., Pooley et al. 2012). Due to the large length scales of the extragalactic regime, which are only partially compensated by the larger velocity scales, the typical time scale on which quasar microlensing occurs is about 10 times longer than for galactic microlensing events (e.g., Kochanek 2004). Therefore, quasar microlensing events have a duration of $1 - 10\ yr$ rather than $0.1 - 1\ yr$.

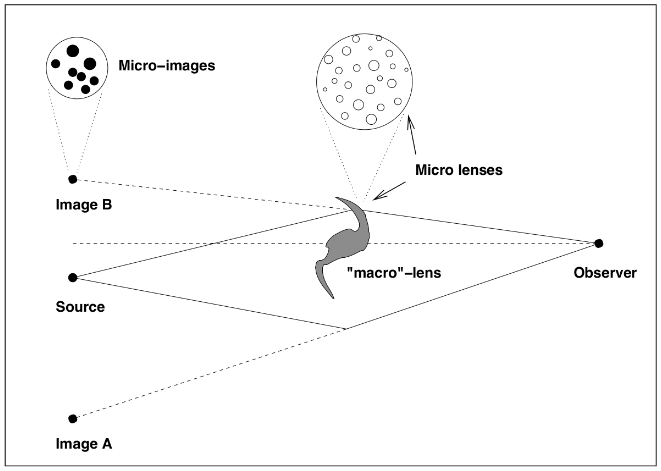


Figure 5: Sketch of a quasar lensed by a galaxy. The galaxy as a whole producing two images of the quasar. The stars inside the galaxy causing a microlensing effect of one of the images. From Courbin et al. (2002)

3 Machine Learning and Deep Learning

This chapter covers a brief overview of the basic concepts of machine learning and deep learning for classification that are relevant for this work. A more detailed presentation of the theoretical background is given in several references, such as Hastie et al. (2009), Frochte (2019) and Moroney (2020).

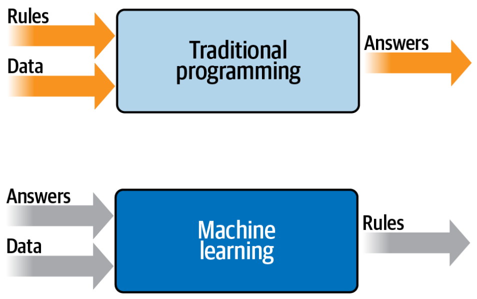


Figure 6: Traditional programming vs. machine learning, from Moroney (2020)

The idea of machine learning is to feed a machine with inputs and outputs, instead of writing a programme that returns an output for an input, such that it gives a programme (see figure 6). The goal is to learn a mathematical function

$$f : X \rightarrow Y . \tag{21}$$

For the purpose of this work, supervised learning is considered. This means, a sufficiently large set of inputs and outputs is provided that already have the correct function value. Data sets are then referred to as labelled datasets. The dataset is usually divided into training and test examples.

3.1 Classification

For classification, the codomain Y from 21 is discrete. Given a number of characteristics, called *features*, according to which the belonging to a class with the corresponding *label* is to be determined. Formally, this can be summarised as follows (Frochte 2019):

Let X be the space of feature vectors and C a set of classes. There is a usually unknown function

$$c : X \rightarrow C , \tag{22}$$

which does the error-free classification. We are generally only aware of a set of examples:

$$D = \{ (x_1, c(x_1)), (x_2, c(x_2)), ...(x_n, c(x_n)) \} \subseteq X \times C . \tag{23}$$

The goal of the classification problem is to construct this function c.

The classification problem can also be thought of as a set of datapoints in the space of feature vectors that are to be subdivided into their respective classes by demarcation. Figure 7 shows a simplified example of a set of datapoints with two classes with labels "+" and "$*$" and two features x_1 and x_2. In classical machine learning algorithms, this grouping is done by hyperplanes. Using neural networks, more complex continuous functions can be approximated.

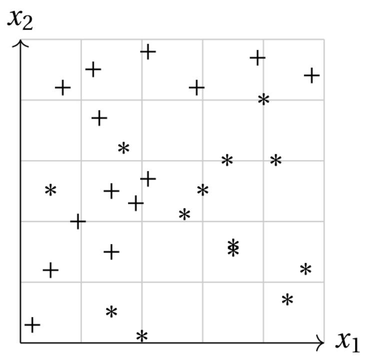


Figure 7: Example of a set of data points in two dimensions, from Frochte (2019)

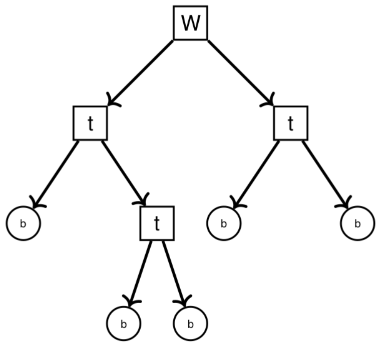


Figure 8: Example of a binary tree, from Frochte (2019)

3.2 Decision Trees

Decision trees are classifiers based on the basic data structure of a tree, which is well known in computer science. A tree consists of a *root*, which is the only node without *parent node*, several *internal nodes* (also: *branch nodes*), and several *leaf nodes*, which are nodes without *child nodes*. A tree can be built recursively as a collection of nodes, starting with the root node, by dividing the nodes into a number of child nodes and constraining the formation of child nodes. Figure 8 shows an example of a binary tree for which each node, except leaf nodes, has two children.

For a decision tree the process of evaluation begins at the root. A decision is made at the root and at each internal node. At the leafs, which represent the labels, the classification is made. The branches represent the conjunction of the features that lead to these labels. On the first level, the algorithm divides all training examples into as many branches as labels are given. On the second level, the respective branches are again divided and so on. The number of levels is indicated by the maximal depth of the tree classifier. The result is an $(n-1)$-dimensional hyperplane in n-dimensional feature space. For a given data set there can be different splitting conditions. The model has to learn to split the data optimally. At each level, it compares every possible split and chooses the one that maximises the *information gain (IG)* by going through all possible features and feature values to find the best feature and corresponding threshold. A way to measure information contained in a node can be done by using the *entropy*:

$$H(node) = \sum -p_i \, log(p_i) \, , \tag{24}$$

where p_i is the probability of class i. The highest possible value of entropy is 1, which is the entropy corresponding to the root node. The information gain corresponding to a split is defined as:

$$IG = H(parent) - \sum H(child_i) \ . \tag{25}$$

The advantage of decision trees is that they can split data that would not be separable by a single linear function. However, they are highly sensitive to the training dataset and tend to *overfit*, which means that the algorithm fits very well to the classification of the training data, but does not generalise.

3.3 Ensemble Machine Learning Algorithms

3.3.1 Random Forests

A random forest (e.g., Ho 1995) is a collection of multiple random decision trees. The first step in building a random forest is to generate new datasets by randomly selecting datapoints from the original dataset. Each new dataset contains the same number of datapoints as the original data set. This process is called *bootstrapping* and it makes the model less sensitive to the original training dataset. In the next step a decision tree is trained on each of the bootstrapped datasets independently. For each tree a subset of feature is randomly chosen for training, which reduces the correlation between the trees. For the classification of a datapoint traversing the random forest, the average decision of all classifications of the individual trees is selected. Combining the results from several models is called *aggregation*. The combination of bootstrapping and aggregation is called *bagging*.

3.3.2 Adaptive Boosting Classifier

The bagging method mentioned above combines the results of optimised individual classifiers to achieve better generalisation. In contrast, the boosting method is used to train multiple different *weak classifiers*, which are models that are slightly better than random guessing, to combine them into a *strong classifier*. One of the most popular boosting algorithms is *Adaptive Boosting* (short: *AdaBoost*). A detailed description of the algorithm is given in Freund and Schapire (1995). The idea is to use a forest with shallow trees as weak classifiers. The procedure is iterative, adding a random weak classifier in each iteration step. Misclassified training examples are weighted higher and correctly classified ones are weighted lower. Unlike in a random forest, in a forest made with AdaBoost the final classification is determined by a weighted sum of the component classifications and the order of trees plays an important role.

3.4 Artificial Neural Networks

3.4.1 Introduction

The elementary component of a neural network is the *artificial neuron*. Figure 9 shows the schematic of how a neuron operates as part of a neural network. The outputs of the previous neurons are multiplied by the *weights* and summed up, which is used as the argument of the *activation function*. The value of the activation function is then the output of the neuron in question. Neural networks

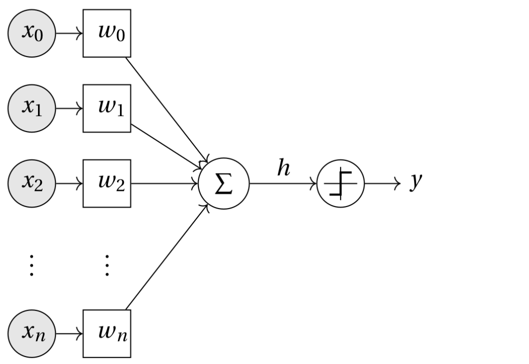


Figure 9: Scheme of an artificial neuron, from Frochte (2019)

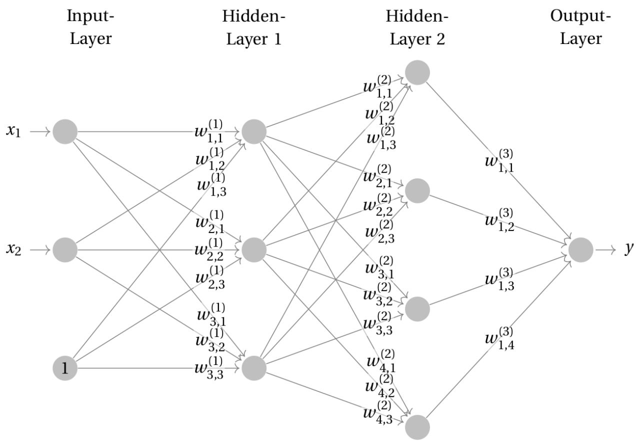


Figure 10: Example of a neural network: fully connected multilayer-perceptron with two hidden layers, from Frochte (2019)

consist of an *input layer*, an *output layer* and possibly one or more *hidden layers*. A layer contains a certain number of neurons. If all neurons of one layer are connected to the neurons of the following layer, this is called a *fully connected neural network* (or *dense neural network*). A neural network in which one layer is only connected to the following layer is referred to as a *feedforward network*. As an example of a feedforward-network, in figure 10 a fully connected *multilayer-perceptron* with two hidden layers is illustrated. Networks with multiple hidden layers are so-called *deep neural networks*. The number of layers, number of neurons per layer, which nodes are connected, which weights are shared, the selected activation function in a respective layer, and whether a *bias* is added in a layer specifies the architecture of the network.

3.4.2 Output

The output vector of a certain layer $L \geq 1$, except the input layer $L = 0$, in a neural network can be written as a matrix multiplication:

$$\mathbf{o}^{(L)} = \begin{pmatrix} o_1^{(L)} \\ \vdots \\ o_n^{(L)} \end{pmatrix} = a(\mathbf{i}^{(L)} \cdot \mathbb{W}^{(L)}) \,, \tag{26}$$

where n is the number of neurons in layer L, $\mathbf{i}^{(L)}$ is the vector giving the input into layer L, and a is the activation function. $\mathbb{W}^L$ is the matrix containing the weights $w_{jk}^{(L)}$ of layer L, where j indicates the considered neuron of layer L and k that one of layer $L-1$. The input of a neural network is given by the components of the feature vector $\mathbf{x} = \mathbf{i}^{(0)}$. The output of a neural network can be written as:

$$\mathbf{y} = \mathbb{W}^{(L_{max})} \cdot a(\mathbb{W}^{(L_{max}-1)} \cdot a(\mathbb{W}^{(L_{max}-2)} \cdot a(\dots a(\mathbb{W}^{(2)} \cdot a(\mathbb{W}^{(1)} \cdot \mathbf{x})) \dots)) \,. \tag{27}$$

3.4.3 One-Hot-Encoding

For a classification with neural networks, integers are only useful as target coding if there exists an ordering in the classes and this ordering can be expressed in the transitions. Since this is the exception, one proceeds to *one-hot encoding*, where the scalar target value is replaced by a vector target value. The dimension of this output vector $\mathbf{y}$, where $y_{1,2,\dots C} \in [0,1]$, is given by the number of classes C and constraints the number of output neurons. The belonging of an element to a class with label l is characterised by the component representing this class being 1 and the remaining components being 0:

$$y_{Di} = \begin{cases} 1, & if \;\; i = l \\ 0, & otherwise \end{cases} \,. \tag{28}$$

3.4.4 Loss Function, Optimisation and Accuracy

In contrast to the network structure and the activation functions of the layers, which are fixed, the weights are supposed to be optimally determined by training the network. Let $D = \{X, Y\}$ be the given training set and (x_D, y_D) its value pairs. The deviation of the obtained output y from the given label y_D is described by a *loss function* $J(\mathbb{W})$ which depends only on the weights. The goal is to minimise the loss function by adjusting the weights. *Optimisation algorithms* are used to determine the weights. Most of them are based on the *gradient descent method* (e.g., Lemaréchal 2012), including *Adam* (short for *Adaptive Moment Estimation*) presented in Kingma and Ba (2015). There are different loss functions, for example the *mean square error*:

$$J(\mathbb{W}) = \frac{1}{N_p} \sum_{p=1}^{N_p} \sum_i \left(y_{Di}^{(p)} - y_i^{(p)} \right)^2 \,, \tag{29}$$

where N_p is the total number of sample datapoints and p indicates a certain datapoint. For classification, the *cross entropy error*

$$J(\mathbb{W}) = -\frac{1}{N_p} \sum_{p=1}^{N_p} \sum_i \, y_{Di}^{(p)} \, log(y_i^{(p)}) \,, \tag{30}$$

is favourable. This requires a probability distribution among the classes, which means that the values for the individual classes add up to 1. To ensure this, the *softmax function* is used as activation function in the output layer. The softmax function is defined as:

$$\sigma_j(\mathbf{z}) = \frac{e^{z_j}}{\Sigma_{k=1}^{K} e^{z_k}} \quad for \;\; j = 1, \dots, n, \tag{31}$$

which assigns to each component of an n-dimensional vector $\mathbf{z}$ a probability.

3.5 Convolutional Neural Networks

Convolutional neural networks (short: *CNN*) in the modern sense have emerged from the work of Lecun (1989). This particular type of feedforward network is applied to data that can be put into a grid structure, such as images and time series. The schematic of a convolutional neural network is illustrated in figure 11. A CNN consists of one or more *convolutional layers* (3.5.1) and *pooling layers* (3.5.2) that are combined in an alternating sequence and connected to a dense neural network (discussed in 3.4.1) by a *flattening process* (3.5.3).

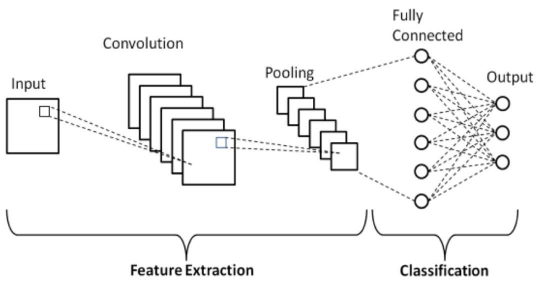


Figure 11: Schematic of a convolutional neural network, credit: Hiep and Joo (2018).

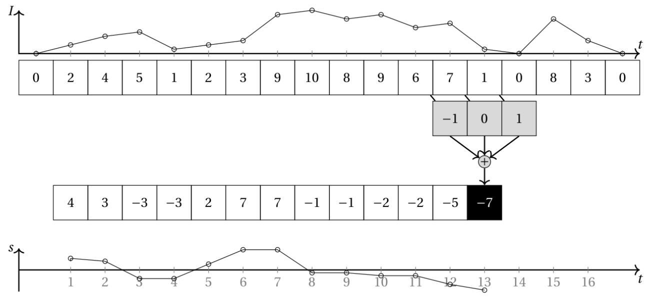


Figure 12: Example of a discrete convolution on a time signal, credit: Frochte (2019).

As seen in section 3.4.2, the output of a layer in a neural network can be written as a matrix multiplication. In a dense layer the matrix is usually fully populated. This means that theoretically every input interacts with every output. This is not the case with a convolutional layer. Inputs outside the domain of the kernel have no influence on the considered value of the convolution, resulting in a *sparse interaction*. This leads to *parameter sharing*, which means that weights are shared. The number of diagonals occupied in the matrix is limited to the number of degrees of freedom that the kernel has.

3.5.1 Convolutional Layer

Considering a discrete time signal $I(t)$, where the data is sampled uniformly along t at a sampling rate Δt, then each sampling point i is at $\Delta t \cdot i$ and the discrete input is indicated as $I\,[i]$. Let K be a *convolution kernel* with size k, then this operation can be written as:

$$s\,[i] = \sum_{j=-(k-1)/2}^{(k-1)/2} I\,[i+j] \cdot K\,[j]\,. \tag{32}$$

An example of a discrete convolution on a time signal can be seen in figure 12. In a convolutional layer, several convolutional kernels (or filters) are used in parallel. Each kernel emphasises different properties of the original signal. Thus, the convolutional layer generates new features. This means, for example, if eight kernels are used, about eight times as many features are obtained.

Thus, a compression of the information is required, which will be explained in 3.5.2.

3.5.2 Pooling Layer

In a pooling layer, the information from the convolution layer is compressed. There are various approaches for this, for example *max-pooling* or *average-pooling*. The output is divided into sections of a certain width z. Using max-pooling, the maximum value is adopted from each section. In the case of average pooling, the average value is taken from each section.

3.5.3 Flattening

In order for the output from the last pooling layer to be used as input for a dense neural network, these values must be converted into a vector. This process is referred to as *flattening*.

3.5.4 Feature Extraction

The sequential process of convolution and pooling generates the features for the connected dense neural network. Instead of specifying or elaborating the features in advance, this task can be outsourced to the layers. Hence, the neural network learns the features by itself.

4 Simulation of Lightcurves

4.1 Approach

A common technique for simulating lightcurves is to use a *magnification map* (e.g., Courbin et al. 2002, Tomozeiu et al. 2017), i.e. a map representing caustic networks in the source plane (see section 2.3.2). Each pixel in such a map is representative for the magnification that a source would undergo at the corresponding pixel's coordinates. A microlensing magnification map can be generated by using the code of Wambsganss (1999) (section 4.2). By using this code, two different magnification maps were generated for this work: *magnification map A* (figure 13) and *magnification map B* (figure 14). The first one is a simplification representing an atypically simple region of a realistic magnification map and the second one is a more realistic magnification map.

Lightcurves can be simulated by tracking a source within a magnification map as it moves along a trajectory that crosses a caustic fold. The total brightness is the convolution of the source brightness function and the magnification at the corresponding pixels of the magnification map that the source traverses, as given in equation 17, where the integral becomes a sum of discrete pixels. In this work three different source shapes with constant surface brightness distributions were considered: *disk* (section 4.3.1), *crescent* (section 4.3.2) introduced by Kamruddin and Dexter (2013), and *random* (section 4.3.3).

The training of a neural network requires large amounts of data. Therefore, an algorithm was developed to automatically generate a large number of lightcurves for the three different source types within a given magnification map (section 4.4). Using this algorithm, two different datasets with two different magnification maps were generated (section 4.5).

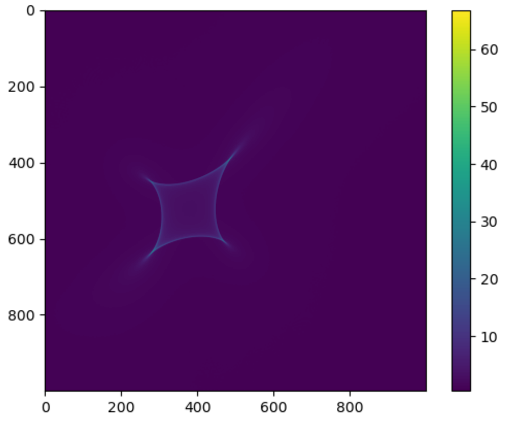


Figure 13: Magnification map A, generated by using the code of Wambsganss (1999).

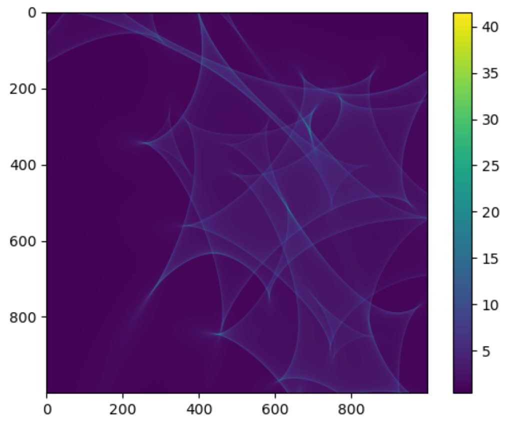


Figure 14: Magnification map B, generated by using the code of Wambsganss (1999).

4.2 Magnification Map

In order to generate the two magnification maps used in this work, the code created by Joachim Wambsganss (Wambsganss 1999) was used. The code uses the technique of *inverse ray shooting*

(Kayser et al. 1986; Schneider and Weiss 1987; Wambsganss 1990) (section 4.2.1) and a *hierarchical tree code* (section 4.2.2) for the calculation of the deflection angles. The idea of a hierarchical tree code was originally presented in the work of Barnes and Hut (1986) for the purpose of stellar dynamical problems.

4.2.1 Inverse Ray Shooting

In the inverse ray shooting technique, light rays are shot from the observer to the lens plane, where the individual stars act as microlenses and deflection occurs. When the deflected rays hit the source plane, they are collected in small squares (pixels), resulting in a two-dimensional density distribution of the light rays, the magnification map. In such a map, the magnification a source would undergo at a certain pixel is directly proportional to the number of rays the corresponding pixel contains.

By using the lens equation (7), a grid of rays from the $\vec{\theta}$-plane (lens plane) to the $\vec{\beta}$-plane (source plane) is mapped, considering the microlensing effect due to the individual lenses. The deflection angle $\vec{\alpha_i}$ of a light ray i shot through a single-lens plane is given by the sum of the deflection angles induced by all individual stars acting as microlenses. For $N_\star$ point lenses, the deflection angle is given by:

$$\vec{\alpha_i} = \sum_{j=1}^{N_\star} \vec{\alpha_{ji}} = \frac{4G}{c^2} \sum_{j=1}^{N_\star} M_j \frac{\vec{r_{ij}}}{|\vec{r_{ij}}|^2}, \tag{33}$$

where $\vec{r_{ij}}$ is the two-dimensional projected distance vector between light ray i and point lens j and M_j is the mass of point lens j. For the computation of the resulting deflection, the effect due to continuously distributed matter, meaning a smoothed out surface mass density contributing as an additional constant, and external shear, i.e. the tidal force caused by asymmetrically distributed matter far away from the region under consideration, are also included. These are specified by the macromodel of the gravitational lens.

The number of individual lensing stars $N_\star$ is constrained by the mass model of the lensing galaxy. To achieve a high resolution, a large number of pixels N_{pix} is needed. For a realistic $N_\star$, by assuming a high density of rays per pixel on average and considering a large N_{pix} the direct calculation causes a long computation time. This can be reduced by using a hierarchical tree method.

4.2.2 Computation of the Deflection Angle with a Hierarchical Tree Code

From equation 1 it can be seen that the deflection angle has a r^{-1} dependence, meaning lenses that are further away from the light ray play a less important role. Therefore, lenses can be treated differently according to their distance to the light ray. In the hierarchical tree code, the lenses are grouped into cells of different sizes. For a certain light ray, the deflection angles induced by nearby lenses are computed directly by using equation 33. More distant lenses are bunched together in groups and their net contribution is approximated by the first few terms of a multipole expansion.

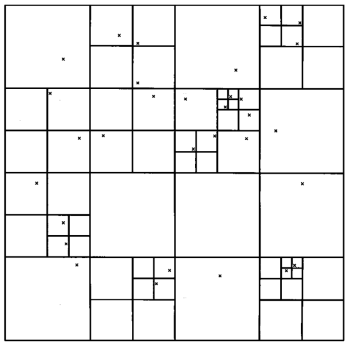


Figure 15: Example of the hierarchical grouping with a tree code for 31 randomly distributed lenses, from Wambsganss (1999).

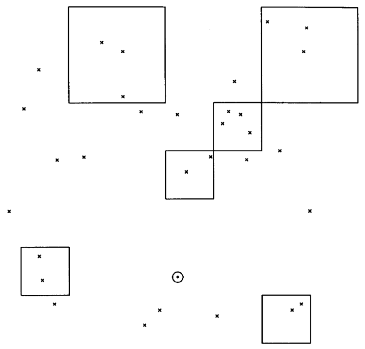


Figure 16: Example of the decision for computation of the deflection angle induced by 31 randomly distributed lenses for a given position of a light ray (circled dot), from Wambsganss (1999).

The tree code uses a hierarchical grouping of the individual lenses starting from the root node containing the total field with $N_\star$ lenses. Since the lens plane is two-dimensional, a quadtree is applied. This means that each node, excluding the leaf nodes, is divided into four child nodes with half the side length of the parent node. This subdivision takes place until there is maximally one lens in each node, giving the leaf nodes. In figure 15 an example of the hierarchical grouping for $N_\star = 31$ lenses is illustrated.

The decision whether to perform the calculation directly or with the multipole approximation is made by comparing the distance of the position of the considered light ray to a group with the size of the node containing this group. If the distance is much larger than the size of the node, the multipole approximation is used. Otherwise the direct calculation occurs. This criterion can be indicated by the opening angle of the node as seen from the position of the ray. Figure 16 shows an example of the decision for the computation of the deflection angles for a lightray at a given position induced by $N_\star = 31$ lenses.

4.3 Models for extended Sources with constant Surface Brightness Distribution

Using the existing models of a a constant brightness *disk* (section 4.3.1) and a *crescent* shaped source (section 4.3.2), additionally four different models of randomly shaped sources, which have no physical meaning, are developed, all under the label *random* (section 4.3.3).

The model of a *crescent*-shaped source was introduced by Kamruddin and Dexter (2013), where the *crescent* represents the silhouette of a black hole over an accretion disk. The model of a constant brightness *disk* is typically found earlier in the literature to describe the luminous parts of quasars. These two models are implemented in a similar way as in the work of Tomozeiu et al. (2017).

4.3.1 Disk Source

The brightness function of a *disk* source with radius R_p and constant surface brightness can be written as (Tomozeiu et al. 2017):

$$S_{2D}^{D} = \frac{S_0^D}{\pi R_p^2}\,\Theta\left(R_p^2 - (p - p_{sp})^2 - (q - q_{sp})^2\right)\,, \tag{34}$$

where S_0^D is the total flux from the *disk* source, Θ is the Heaviside function, and (p_{sp}, q_{sp}) are the coordinates of the centre of the *disk*.

4.3.2 Crescent Source

A *crescent* source can be constructed by starting from a constant brightness disk with radius R_p and cut a smaller non-concentric disk with radius R_n out of it. The distance from the centre of the larger disk to the centre of the smaller disk is described by the parameter c. Figure 17 shows the geometry of a *crescent*-shaped source. It's surface brightness distribution is given by the

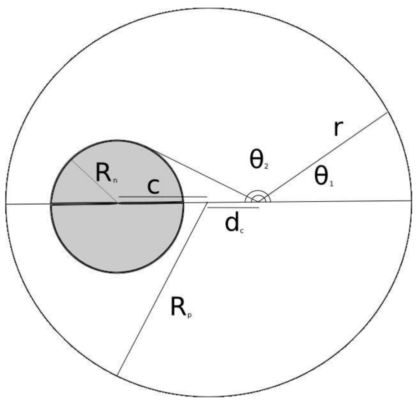


Figure 17: Geometry of a *crescent*-shaped source, from Tomozeiu et al. (2017)

superposition of the surface brightness distributions of the larger disk contributing positively to the total flux and of the smaller disk contributing negatively. The brightness function of a *crescent* source can be written as (Tomozeiu et al. 2017):

$$S_{2D}^{C} = \frac{S_0^C}{\pi\left(R_p^2 - R_n^2\right)}\left[\Theta\left(R_p^2 - (p - p_{sp})^2 - (q - q_{sp})^2\right) - \Theta\left(R_n^2 - (p - p_{sn})^2 - (q - q_{sn})^2\right)\right]\,, \tag{35}$$

where S_0^C is the total flux from the *crescent*-shaped source, (p_{sp}, q_{sp}) are the coordinates of the centre of the larger disk and (p_{sn}, q_{sn}) that of the smaller disk. Two constraints on the parameters must be satisfied:

I) The two radii must obviously fulfil the relation:

$$R_p \geq R_n\,. \tag{36}$$

II) The smaller disc must always be positioned inside the larger disc:

$$R_p \geq R_n + \sqrt{a^2 + b^2}\,, \tag{37}$$

where $a \equiv p_{sn} - p_{sp}$, $b \equiv q_{sn} - q_{sp}$ and $\sqrt{a^2 + b^2} \equiv c$ giving the distances between the centres of the two disks.

4.3.3 Random Source

Random sources are constructed by starting from a bright disk with radius R_p and cut an area A_i out of it. A_i is given by:

$$A_i = \iint_{A_d} F_i(p,q)\, dp\, dq\,, \tag{38}$$

where $A_d = R_p^2 \pi$ is the area of the bright disk and $F_i(p,q)$ is a function specified further below for each subtype of *random* sources. Similar as for the *crescent* source, the surface brightness distribution for a given subtype of *random* source results from the superposition of the surface brightness distributions of the disk, which contributes positively to the total flux, and that of the cut-out area, which contributes negatively. It's brightness function can be written as:

$$S_{2D}^{R_i} = \frac{S_0^{R_i}}{\pi\,(A_d - A_i)} \left[\Theta\left(R_p^2 - (p - p_{sp})^2 - (q - q_{sp})^2\right) - \Theta\left(F_i(p - p_{sp}, q - q_{sp})\right)\right], \tag{39}$$

where $S_0^{R_i}$ is the total flux from the *random* source and (p_{sp}, q_{sp}) are the coordinates of the centre of the disk.

Four different subtypes of *random* sources were considered. They are defined by the function $F_i(p,q)$:

- **Random-1**

$$F_i(p,q) = F_1(p,q) = p\, q\, \pi - \xi\,, \tag{40}$$

where ξ is a parameter with dimension $(\mathit{length})^2$.

- **Random-2**

$$F_i(p,q) = F_2(p,q) = (p - \sqrt{\xi})\,(q - \sqrt{\xi}) - \xi\,. \tag{41}$$

- **Random-3**

$$F_i(p,q) = F_3(p,q) = (p + \alpha)\,(q + \beta)\,\pi - \xi\,, \tag{42}$$

where α and β are parameters with dimension *length*.

- **Random-4**

$$F_i(p,q) = F_4(p,q) = (p+\alpha)\, q\, \pi - \xi \,. \tag{43}$$

Examples of the subtypes of a *random* source can be found in the appendix (A).

4.4 Lightcurve Simulation Algorithm

In order to generate automatically a large number of lightcurves for a given magnification map, an algorithm was developed that consists of four parts. **Part I** is a preprocessing work, which has to be done once for a given magnification map. **Parts II-IV** are the main part of the algorithm.

- **Part I: Finding appropriate Regions for Start and End Points around different Caustic Folds**
  Let $\mathbb{F}$ be the set of all caustic fold pixels in a given magnification map, i.e. the set of all pixels with discrete pixel coordinates (p_F, q_F) on which a source would undergo a magnification $\mu(p_F, q_F) > \mu_{F_\downarrow}$, where $\mu_{F_\downarrow}$ defines the lower limit for the caustic fold pixels and depends on the magnification map. Around each pixel (p_G, q_G) of a subset $\mathbb{G} \subseteq \mathbb{F}$ a circle with radius R_C is generated and the number of pixels N_{CF} providing a magnification $\mu(p_i, q_i) > \mu_F$, where $\mu_F > \mu_{F_\downarrow}$, are counted. If $N_{CF} = 2$, the circle intersects exactly one caustic fold, and the coordinates of the caustic fold pixel (p_{G_j}, q_{G_j}) and the radius of the circle R_{C_j} are saved.

  For each circle C_j, a square S_j with the same centre as the circle and side length $s_j = 2\, R_{C_j}$ is constructed. $\mathbb{S}_j$ is the set containing the pixel coordinates of all pixels located in this square. When a source crosses a caustic fold, it crosses the boundary between two regions, with the pixels of one region providing higher magnification than those of the other $\mu_{F_\downarrow} > \mu(p_\uparrow, q_\uparrow) > \mu_B > \mu(p_\downarrow, q_\downarrow)$, where μ_B sets clearly the threshold and depends on the caustic fold. In order to select the start and end points of the path on the opposite sides of the caustic fold respectively, the pixels with coordinates in $\mathbb{S}_j$ are divided into two subsets: $\mathbb{S}_{j\uparrow} := \{(p,q) \in \mathbb{S}_j \mid \mu_{F_\downarrow} > \mu(p,q) > \mu_B\}$ and $\mathbb{S}_{j\downarrow} := \{(p,q) \in \mathbb{S}_j \mid \mu(p,q) < \mu_B\}$. For each caustic fold, the direction in which the source is to move is selected, either from $\mathbb{S}_{j\downarrow}$ to $\mathbb{S}_{j\uparrow}$ or from $\mathbb{S}_{j\uparrow}$ to $\mathbb{S}_{j\downarrow}$.

- **Part II: Generating a Path that crosses a Caustic Fold**
  Let $\mathbb{K}$ be the set containing the indices j of all appropriate circles C_j found for a given magnification map and let $\mathbb{Y} \subseteq \mathbb{K}$ be a subset of it. First, an appropriate region $j \in \mathbb{Y}$ is randomly selected. According to the direction in which the source is to move, the starting point of the path is randomly chosen from the corresponding set:

$$(p_{SP}, q_{SP}) \in \begin{cases} \mathbb{S}_{j\downarrow}, & if\ \ \mathbb{S}_{j\downarrow} \text{ to } \mathbb{S}_{j\uparrow} \\ \mathbb{S}_{j\uparrow}, & if\ \ \mathbb{S}_{j\uparrow} \text{ to } \mathbb{S}_{j\downarrow} \end{cases} . \tag{44}$$

  Around the chosen starting point (p_{SP}, q_{SP}) a circle C_{SP} with radius $R_{C_{SP}} = L_{Path}$ is generated, where L_{Path} is the length of the path. According to the direction in which the source is to move, the set of possible end points is given by the intersection between C_{SP} and $\mathbb{S}_{j\uparrow}$ or C_{SP} and $\mathbb{S}_{j\downarrow}$:

$$\mathbb{E} = \begin{cases} C_{SP} \cap \mathbb{S}_{j\uparrow}, & if \ \mathbb{S}_{j\downarrow} \text{ to } \mathbb{S}_{j\uparrow} \\ C_{SP} \cap \mathbb{S}_{j\downarrow}, & if \ \mathbb{S}_{j\uparrow} \text{ to } \mathbb{S}_{j\downarrow} \end{cases} . \tag{45}$$

The end point of the path is then chosen randomly: $(p_{EP}, q_{EP}) \in \mathbb{E}$. (p_{SP}, q_{SP}) and (p_{EP}, q_{EP}) uniquely define the path. $\mathbb{P}$ is the set of all pixel coordinates of the points on the path.

- **Part III: Generating Sources**
  For each generated path, three sources are generated, i.e. one for each of the three source labels: *disk* (section 4.3.1), *crescent* (section 4.3.2), and *random* (section 4.3.3). For the label *random*, one of the four subtypes, *random-1*, *random-2*, *random-3*, *random-4*, is selected randomly. The values of the parameters corresponding to the respective source labels are chosen randomly. R_p determines the size and is chosen the same for all three source types for a given path.

  The limits defining the minimum and maximum of the respective parameters were chosen as follows:

  For all three sourcetypes

$$20 \geq R_p \geq 35 \ . \tag{46}$$

  For crescent sources

$$\begin{cases} 0.10 \cdot R_p \geq R_n \geq 0.70 \cdot R_p \ \ and \ \ R_n \neq 0 \ , \\ -0.30 \cdot R_p \geq a \geq 0.30 \cdot R_p \ , \\ -0.30 \cdot R_p \geq b \geq 0.30 \cdot R_p \ , \end{cases} \tag{47}$$

  by taking the condition given in equation 37 into account.

  For random sources

$$\begin{cases} 0.20 \cdot R_p \geq \xi \geq 0.80 \cdot R_p \ \ and \ \ R_n \neq 0 \ , \\ 0.20 \cdot R_p \geq \alpha \geq 0.80 \cdot R_p \ , \\ 0.20 \cdot R_p \geq \beta \geq 0.80 \cdot R_p \ . \end{cases} \tag{48}$$

  The threshold values are given in units of the length of one pixel.

- **Part IV: Simulating Lightcurves by sending the Sources along the Path**
  The lightcurves are simulated by tracking the sources as they move along the generated

path. As given in equation 17, the total brightness is the convolution of the source brightness function and the magnification at the corresponding pixel coordinates of the magnification map that the source traverses, where the integrals are replaced by sums over the pixel coordinates. Instead of the total brightness, the dimensionless amplification factor of the brightness is computed. For simulations with a magnification map, it is the number of pixels landing on the source if the lens is present, divided by the number of pixels landing on the source if no lens is present. In other words, it is the sum of all magnifications $\mu(p_i, q_i)$ contained in the pixels i covered by the source when it is located at a point of the path. For a given point on the path, $P = (p_P, q_P) \in \mathbb{P}$, the amplification factor of the brightness can be written as:

$$m(p_P, q_P) = \sum_i \mu(p_{P_i}, q_{P_i}) \,, \tag{49}$$

where (p_{P_i}, q_{P_i}) are the coordinates of the pixels i covered by the source when the source is at point P. The amplification factor is calculated for a sample of the points on the path, i.e. for a subset of $\mathbb{P}$ given by the number of timesteps chosen, N_T. The source is then moved along the path by the length of L_{Path}/N_T per timestep, with the amplification factor, $m(t) = m(p_{P_t}, q_{P_t})$, being calculated for each timestep.

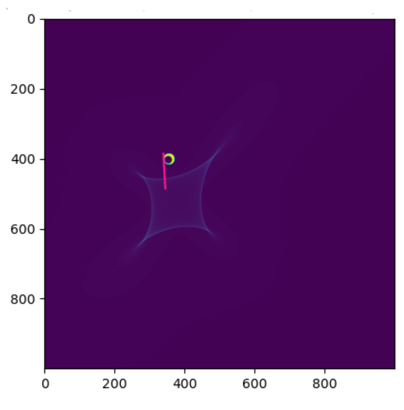
4.5 Simulated Lightcurves and Datasets

With the algorithm described in 4.4 two datasets were generated: *dataset-A* by using *magnification map A* (figure 13) and *dataset-B* by using *magnification map B* (figure 14). For each magnification map two different caustic folds were considered.

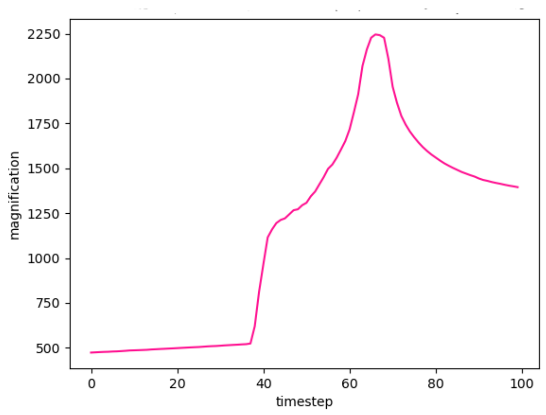
- **Dataset-*A***
  This dataset consists of 179'925 datapoints, with each label accounting for one third of the datapoints. For the length of the path L_{Path}, the length of 100 pixels was chosen, with the sources being moved by one pixel per timestep. The number of timesteps was set to $N_T = 100$. This gives for each datapoint 100 magnification values, i.e. values for the amplification factor of the brightness. Figures 18 and 19 show examples of lightcurves from *dataset-A*.

- **Dataset-*B***
  For this dataset 311'010 datapoints were generated. Likewise, each label constitutes one third of the datapoints. L_{Path} was set to the length of 130 pixels in order to minimise discretisation noise that occurs when using *magnification map B*. The sources were moved by the length of 1.3 pixel per timestep with $N_T = 100$, which also yields 100 magnification values for each datapoint. Additionally, Gaussian noise was added with 4% of the current magnification. Examples of lightcurves from *dataset-B* are illustrated in figures 20 and 21.

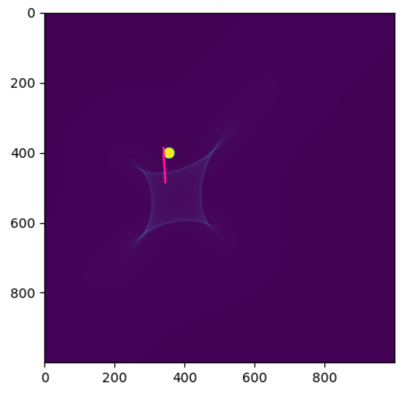
Examples of Lightcurves from *Dataset-A*



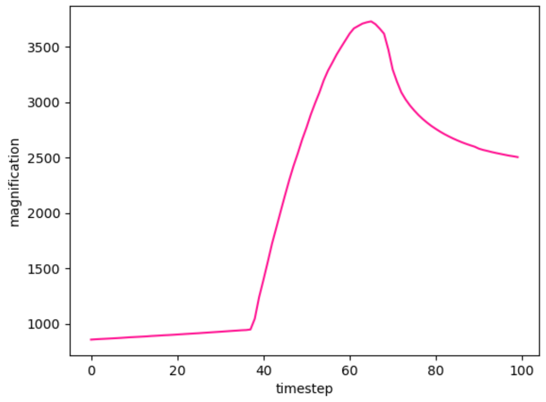
(a)



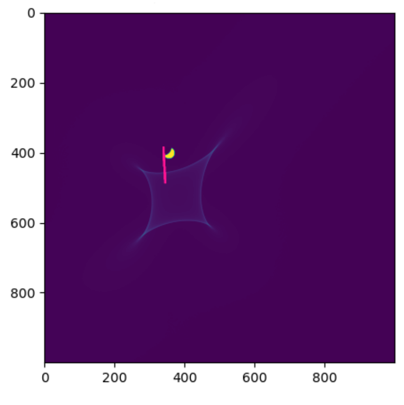
(b)



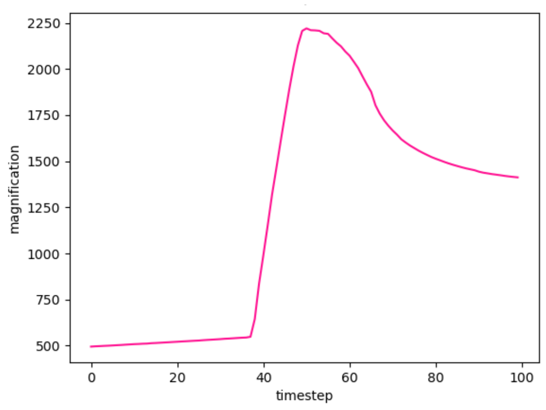
(c)



(d)



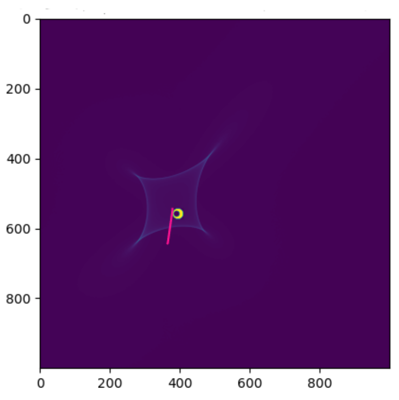
(e)



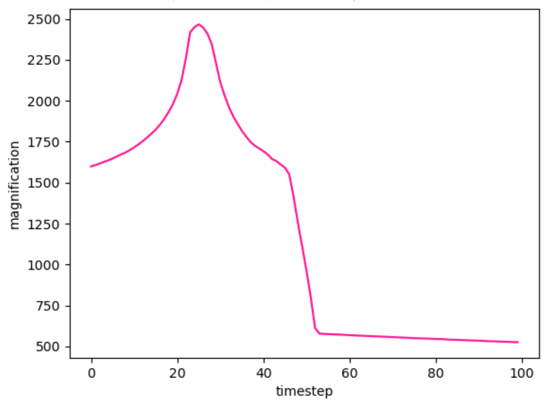
(f)

Figure 18: For each source label, an example of a source with $R_p = 33$ moving along a path crossing *caustic fold-I* in *magnification map A*. (a) Crescent source with $R_n = 0.45 \cdot R_p$, $a = -0.15 \cdot R_p$ and $b = 0.15 \cdot R_p$. (b) Resulting lighcurve for the crescent source. (c) Disk source. (d) Resulting lightcurve for the disk source. (e) Random source with subtype random-2. (f) Resulting lightcurve for the random source.

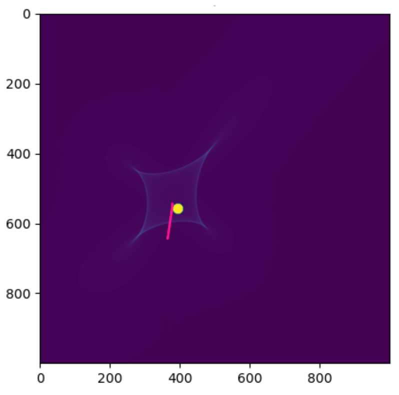
Examples of Lightcurves from *Dataset-A*



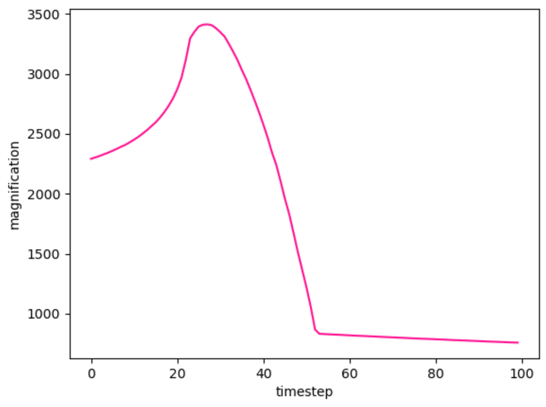
(a)



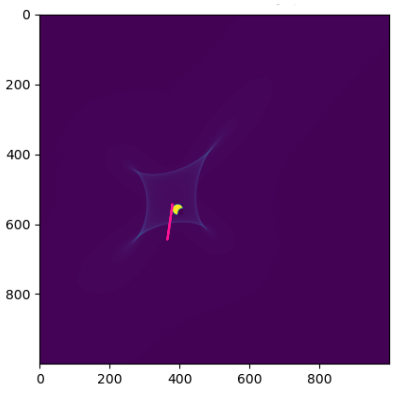
(b)



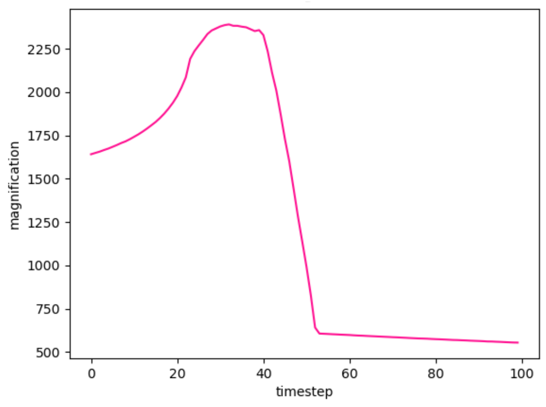
(c)



(d)



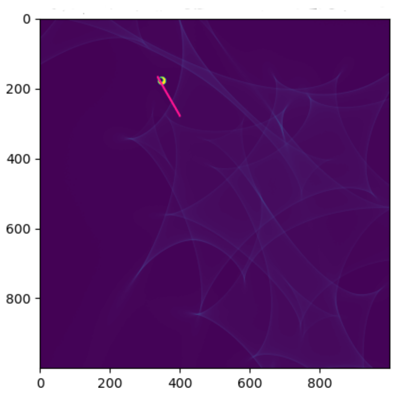
(e)



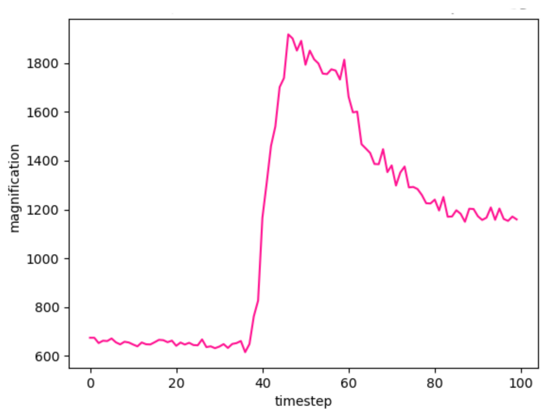
(f)

Figure 19: For each source label, an example of a source with $R_p = 31$ moving along a path crossing *caustic fold-II* in *magnification map A*. (a) Crescent source with $R_n = 0.30 \cdot R_p$, $a = -0.25 \cdot R_p$ and $b = 0$. (b) Resulting lighcurve for the crescent source. (c) Disk source. (d) Resulting lightcurve for the disk source. (e) Random source with subtype random-3. (f) Resulting lightcurve for the random source.

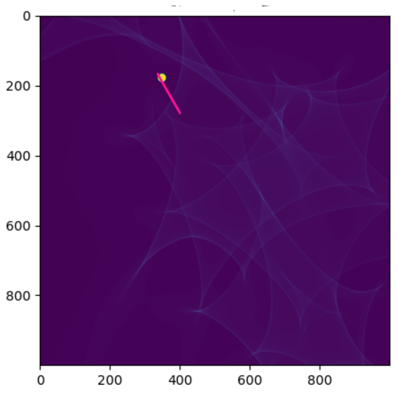
Examples of Lightcurves from *Dataset-B*



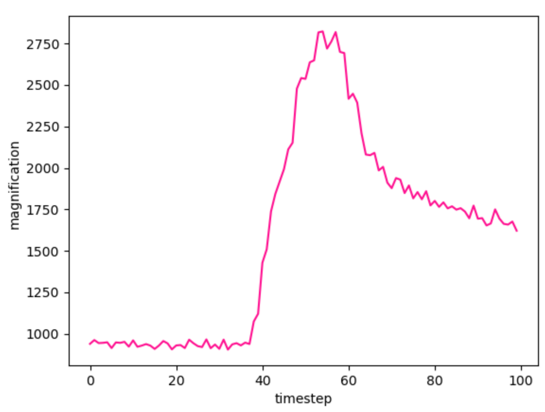
(a)



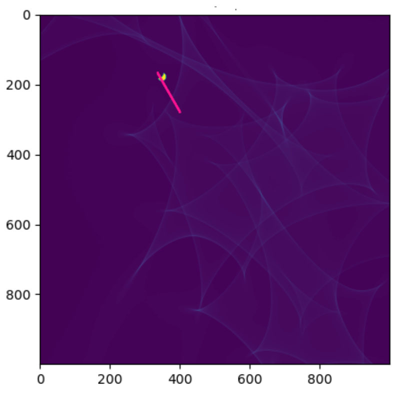
(b)



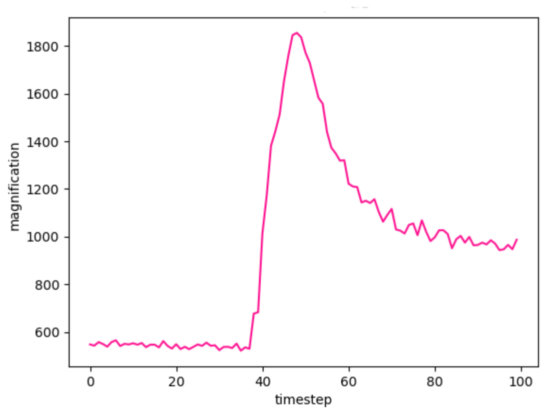
(c)



(d)



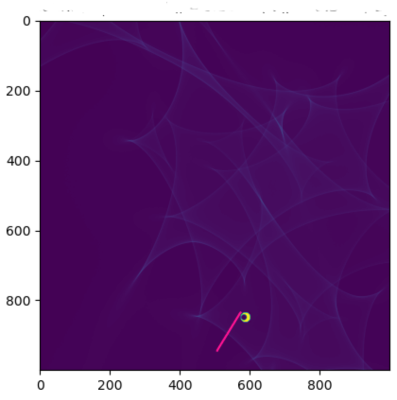
(e)



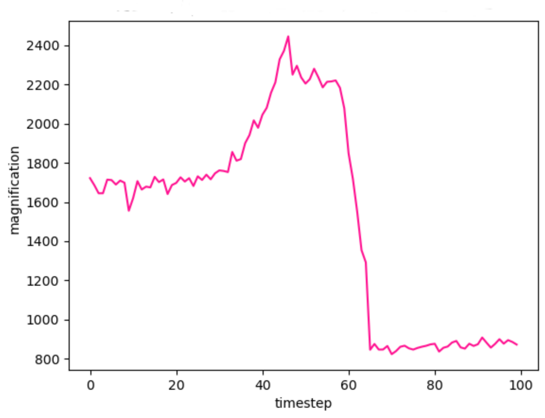
(f)

Figure 20: For each source label, an example of a source with $R_p = 25$ moving along a path crossing *caustic fold-I* in *magnification map B*. (a) Crescent source with $R_n = 0.30 \cdot R_p$, $a = -0.25 \cdot R_p$ and $b = -0.15 \cdot R_p$. (b) Resulting lighcurve for the crescent source. (c) Disk source. (d) Resulting lightcurve for the disk source. (e) Random source with subtype random-2. (f) Resulting lightcurve for the random source.

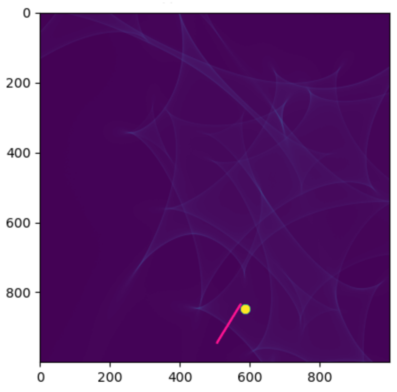
Examples of Lightcurves from *Dataset-B*



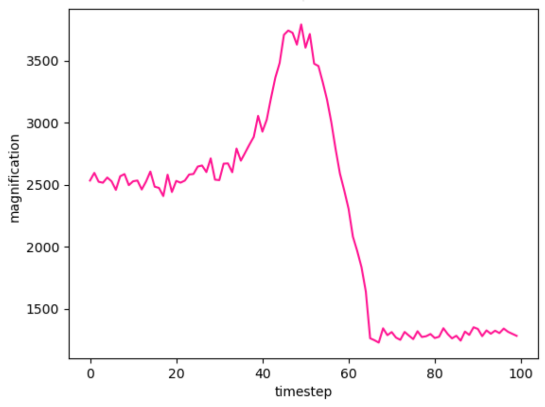
(a)



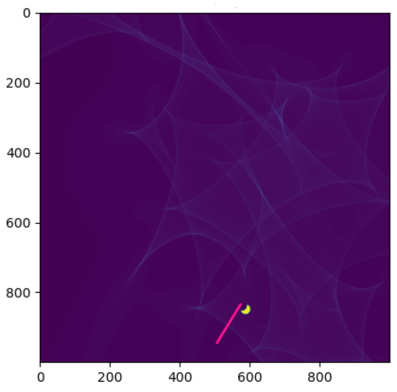
(b)



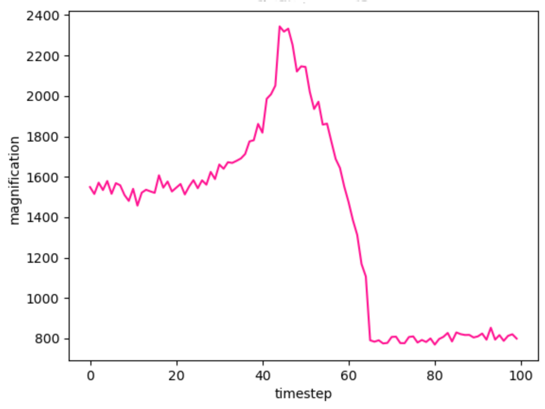
(c)



(d)



(e)



(f)

Figure 21: For each source label, an example of a source with $R_p = 29$ moving along a path crossing *caustic fold-II* in *magnification map B*. (a) Crescent source with $R_n = 0.35 \cdot R_p$, $a = -0.30 \cdot R_p$ and $b = 0$. (b) Resulting lighcurve for the crescent source. (c) Disk source. (d) Resulting lightcurve for the disk source. (e) Random source with subtype random-2. (f) Resulting lightcurve for the random source.

5 Classification of Lightcurves

5.1 Approach

As discussed in section 3.1 the classification can be thought as a geometrical problem in N_f dimensions, with N_f being the number of features. If $N_f > 3$, this exceeds the human imagination, which is why the preferred strategy for finding a neural network model is based on testing different architectures. Theoretically, with a multilayer neural network, using enough neurons, any continuous function can be approximated (Cybenkos 1989; Hornik et al. 1989). However, the risk of overfitting increases when too many neurons are included. An *Adaptive Boosting* classifier (see section 3.3.2) is able to perform better than a simple classifier and is less prone to overfitting than a decision tree, but is not as difficult to train and optimise as a neural network. Classification with *AdaBoost* as a preliminary step to classification with a neural network provides a baseline performance. This can give a first indication of whether the use of machine learning can be a helpful tool for a given problem.

The classification issue of the present work consists of 100 features, i. e. the 100 values for the magnification, which are used to assign the lightcurves to 3 labels specified by the three source types: *crescent*, *disk* and *random*. The goal is to find a neural network model.

As a first approach to extract information from the features, a *Fast Fourier Transformation* (*FFT* for short) (e.g., Cooley & Tukey 1965) is performed for each datapoint in a dataset. Thus, for each dataset, an additional dataset containing the Fourier-transformed lightcurves is provided. Since the Fourier transform of a real signal is symmetric, the number of features for the dataset consisting of the Fourier transforms is reduced to 50.

First, for each dataset and the corresponding dataset consisting of the Fourier-transformed lightcurves, a classification is performed with *AdaBoost* for a subset. For this task, the *AdaBoost algorithm* from *scikit-learn* (Pedregosa et al. 2011), the standard machine learning library for the Python programming language, is used. *Dataset-a* consists of a subset of 7200 datapoints from *dataset-A* and *dataset-b* consists of a subset of 10800 datapoints from *dataset-B*, with each label constituting one third of the datapoints.

Subsequently, a dense neural network (see section 3.4) was trained for each dataset and for its corresponding dataset of Fourier-transformed lightcurves. Finally, a convolutional neural network (see section 3.5) was trained for the two datasets (without their Fourier-transformed datasets). For these purposes, *Keras* (Chollet 2015), a library written in Python that provides an interface to artificial neural networks, was used with the backend *TensorFlow* (Abadi et al. 2015).

5.2 Selection of Neural Network Models

For the neural network models one-hot encoding (see section 3.4.3) was used since there is no ordering in the class labels. The *cross entropy error* (equation 30) is chosen as loss function, which requires a probability distribution among the classes for the output. Therefore, the *softmax function* (equation 31) is used as activation function in the output layer for each model.

5.2.1 Dense Neural Network

To find an appropriate model, neural networks with various numbers of layers and combinations of numbers of neurons in a layer are built and then trained and tested with a random split of the data into 80% training data and 20% test data. The number of neurons in the input and output layers are already determined by the number of features and the number of labels, respectively. The number of layers is initially set to $L = 3$ (1 input layer, 1 hidden layer and one output layer) with 100 neurons in the hidden layer. Afterwards, the number of layers is increased by +1, and different combinations of the number of neurons in the hidden layers are tested, varying the number of neurons in a layer from 50 to 150. This procedure is repeated, also using different activation functions, as long as the accuracy increases with the number of layers. The combination that provides the highest accuracy is finally selected as the model.

5.2.2 Convolutional Neural Network

As explained in section 3.5 a convolutional neural network is a dense neural network preceded by one or more convolutional layers and pooling layers. The architectures of CNN-models can be divided into two components:

- **CNN-Component 1**
  Containing the alternating sequence of convolutional and pooling layers and at the end the flattening layer.
- **CNN-Component 2**
  Consisting of a dense neural network.

In order to find a convolutional neural network model, various combinations of a dense neural network, considering different architectures, and different convolutional and pooling layers are trained and tested. For the choice of the dense layers, the procedure is the same as described above. Regarding the choice of convolutional layers and pooling layers, different combinations of alternating convolutional layers, considering different activation functions, and pooling layers are tested with a random split of the data into 80% training data and 20% test data. The number of convolutional layers, which is the same as the number of pooling layers, is varied between 1 and 3, the size of the convolutional kernels between 2 and 12 and the number of convolutions in a layer between 1 and 10. For the pooling-operation both max-pooling and average-pooling were tested, with the width of the sections being varied between 2 and 8. Likewise the combination that leads to the highest accuracy is chosen.

5.3 Neural Network Models

Tables 1 - 6 show the architectures of the neural network models used. A dense neural network is built for each dataset (tables 1 and 4) and for the corresponding datasets obtained from the Fourier transforms (tables 2 and 5). A convolutional neural network is constructed for each dataset (tables 3 and 6).

5.3.1 Models for Dataset-A

Model-1A (table 1) is the dense neural network model for *dataset-A* and *model-2A* (table 2) that for the Fourier transforms of the corresponding lightcurves. The CNN-model is *model-3A* (table 3).

Table 1: *Model-1A*: Dense Neural Network for *Dataset-A*

Layer	Layer Type	Number of Neurons	Activation Function
0	Input	100	-
1	Dense	80	elu
2	Dense	50	elu
3	Dense	20	elu
4	Output	3	softmax

Table 2: *Model-2A*: Dense Neural Network for *Dataset-A (FFT)*

Layer	Layer Type	Number of Neurons	Activation Function
0	Input	50	-
1	Dense	50	softplus
2	Dense	25	softplus
3	Output	3	softmax

Table 3: *Model-3A*: Convolutional Neural Network for *Dataset-A*

CNN-Component 1

Layer	Layer Type	Number of Convolutions	Kernel Size	Activation Function	Pooling	Pooling Size
0	Input	-	-	-	-	-
1	Convolution 1D	8	10	relu	-	-
2	Pooling	-	-	-	Max-Pooling	4
3	Flattening	-	-	-	-	-

CNN-Component 2

Layer	Layer Type	Number of Neurons	Activation Function
4	Dense	120	elu
5	Dense	80	elu
6	Dense	50	elu
7	Dense	20	elu
8	Output	3	softmax

5.3.2 Models for Dataset-B

The dense neural network model for *dataset-B* is *model-1B* (table 4) and that for the Fourier transforms of the corresponding lightcurves is *model-2B* (table 5). *Model-3B* (table 6) is the CNN-model.

Table 4: *Model-1B*: Dense Neural Network for *Dataset-B*

Layer	Layer Type	Number of Neurons	Activation Function
0	Input	100	-
1	Dense	100	elu
2	Dense	80	elu
3	Dense	40	elu
4	Output	3	softmax

Table 5: *Model-2B*: Dense Neural Network for *Dataset-B (FFT)*

Layer	Layer Type	Number of Neurons	Activation Function
0	Input	50	-
1	Dense	80	softplus
2	Dense	40	softplus
3	Output	3	softmax

Table 6: *Model-3B*: Convolutional Neural Network for *Dataset-B*

CNN-Component 1

Layer	Layer Type	Number of Convolutions	Kernel Size	Activation Function	Pooling	Pooling Size
0	Input	-	-	-	-	-
1	Convolution 1D	8	10	relu	-	-
2	Pooling	-	-	-	Max-Pooling	4
3	Flattening	-	-	-	-	-

CNN-Component 2

Layer	Layer Type	Number of Neurons	Activation Function
4	Dense	150	elu
5	Dense	100	elu
6	Dense	50	elu
7	Dense	20	elu
8	Output	3	softmax

5.4 Evaluation of the Models

In order to evaluate the models *stratified k-fold cross-validation* with $k = 5$ is used. Each dataset is randomly split into 5 partitions of equal sizes with each label constituting one third of the datapoints. For a partition $i \in \{1, 2, 3, 4, 5\}$, the training set is given by the remaining partitions $j \neq i$ and the test set is the partition i. This provides the 5 folds. The expression *fold* used in this context should not be confused with the expression *caustic fold* from gravitational microlensing theory (see section 2.3.2). The accuracy of a model is determined by the mean of the accuracies for each fold, with the error determined by the standard deviation.

6 Results

In the following, the results of the classifications of the lightcurves are presented. A summary of the accuracies achieved is given in section 6.1. The confusion matrices giving an illustration of the predicted labels vs. true labels can be found in section 6.2. For *model-3B*, in section 6.3 the percentage of misclassified lightcurves for a given label vs. properties of the true sources and that of the predicted sources used for the simulation of the respective lightcurves are shown. The results are discussed in chapter 7.

6.1 Summary

The accuracies of the classifications are summarised below: In tables 7-10 for the classifications of *dataset-a* (subset of *dataset-A*) and *dataset-b* (subset of *dataset-B*) with *AdaBoost* (see section 3.3.2), for the neural network models for *dataset-A* (see section 5.3.1) in tables 11-13 and for the neural network models for *dataset-B* (see section 5.3.2) in tables 14-16.

Table 7: Accuracies *AdaBoost* for *Dataset-a*

Fold	Accuracy
1	91.53%
2	91.04%
3	92.36%
4	92.64%
5	93.82%
Average	(**92.28 ± 1.07**)%

Table 8: Accuracies *AdaBoost* for *Dataset-a (FFT)*

Fold	Accuracy
1	96.67%
2	96.39%
3	96.94%
4	96.74%
5	97.36%
Average	(**96.82 ± 0.36**)%

Table 9: Accuracies *AdaBoost* for *Dataset-b*

Fold	Accuracy
1	89.31%
2	89.21%
3	89.17%
4	90.83%
5	89.77%
Average	(**89.66 ± 0.70**)%

Table 10: *Accuracies AdaBoost* for *Dataset-b (FFT)*

Fold	Accuracy
1	92.27%
2	92.27%
3	91.85%
4	91.30%
5	92.41%
Average	(**92.02 ± 0.45**)%

Table 11: Accuracies *Model-1A*

Fold	Accuracy
1	97.23%
2	97.07%
3	91.99%
4	97.25%
5	97.36%
Average	(**96.18** ± **2.34**)%

Table 12: Accuracies *Model-2A*

Fold	Accuracy
1	95.47%
2	96.05%
3	96.82%
4	96.43%
5	94.77%
Average	(**95.91** ± **0.81**)%

Table 13: Accuracies *Model-3A*

Fold	Accuracy
1	99.82%
2	99.83%
3	99.69%
4	99.75%
5	99.89%
Average	(**99.80** ± **0.08**)%

Table 14: Accuracies *Model-1B*

Fold	Accuracy
1	93.28%
2	95.50%
3	94.95%
4	94.72%
5	92.61%
Average	(**94.21** ± **1.21**)%

Table 15: Accuracies *Model-2B*

Fold	Accuracy
1	93.46%
2	92.53%
3	87.25%
4	93.66%
5	85.92%
Average	(**90.56** ± **3.69**)%

Table 16: Accuracies *Model-3B*

Fold	Accuracy
1	99.27%
2	99.61%
3	99.27%
4	99.15%
5	99.21%
Average	(**99.30** ± **0.18**)%

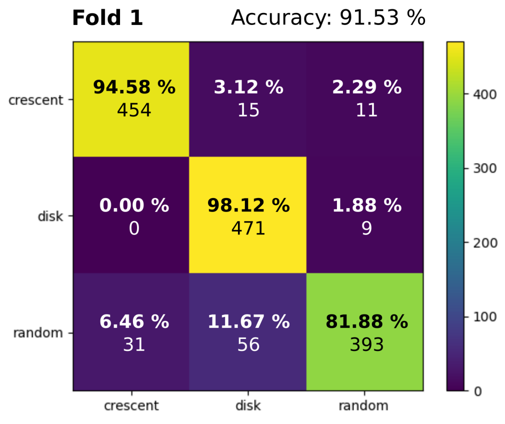
6.2 Confusion Matrices

A confusion matrix visualises how often a considered label was assigned to a certain label. The x-axis represents the true labels and the y-axis the predicted labels.

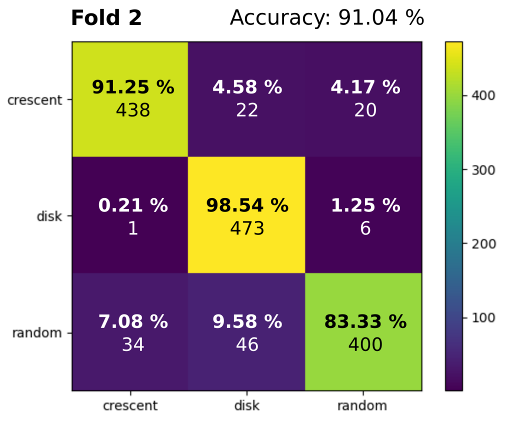
Below, the confusion matrices of the different folds for the respective classifications are presented: In figures 22-25 for the classifications of *dataset-a* (subset of *dataset-A*) and *dataset-b* (subset of *dataset-B*) with *AdaBoost*, for the neural network models for *dataset-A* in figures 26-28 and for the neural network models for *dataset-B* in figures 29-31.

Classification with Adaptive Boosting

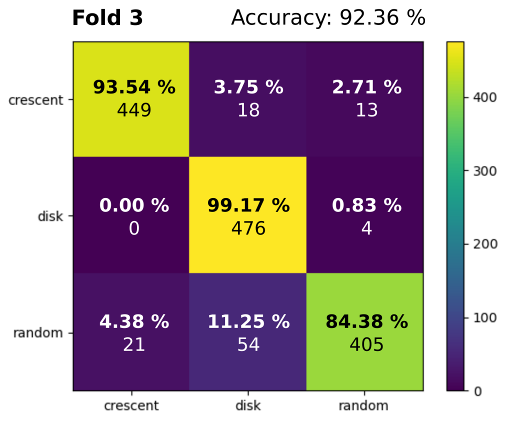
Adaptive Boosting Classifier for *Dataset-a*



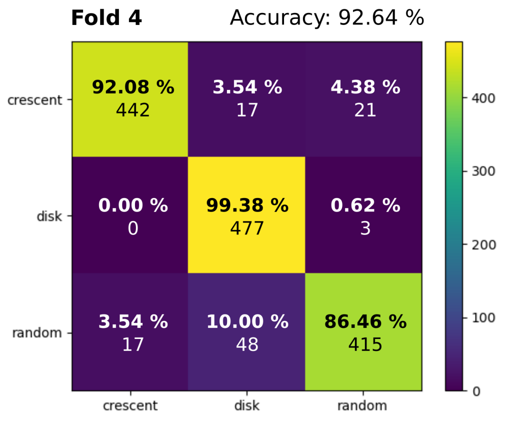
(a)



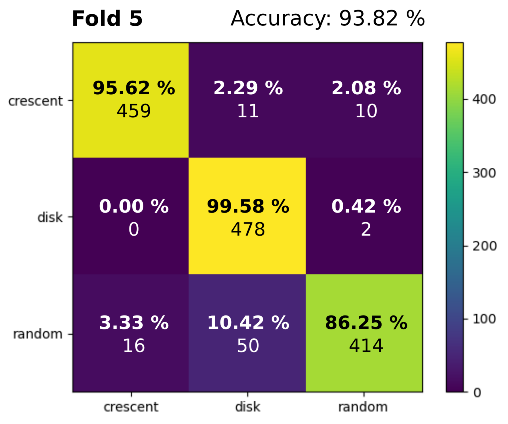
(b)



(c)



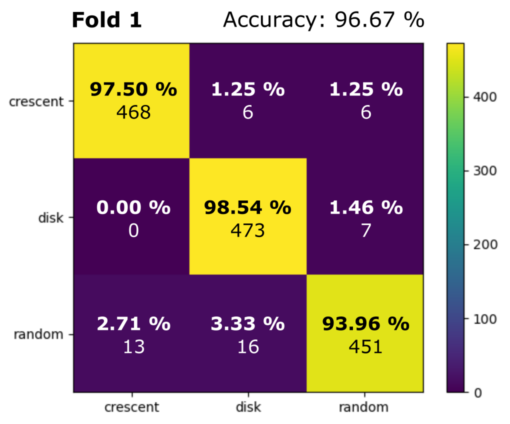
(d)



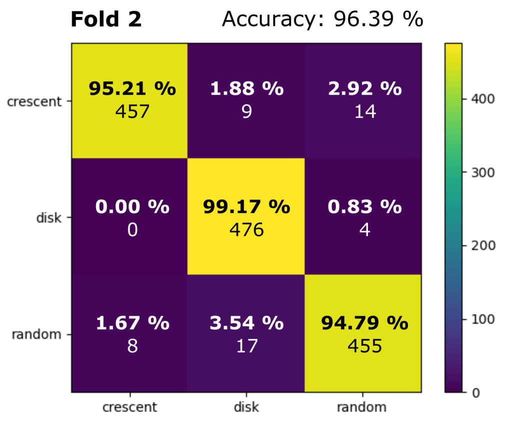
(e)

Figure 22: Confusion matrices and accuracies achieved for folds 1-5 for the classification of *dataset-a* with *AdaBoost*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 480, the total number of lightcurves for each label in each test set.

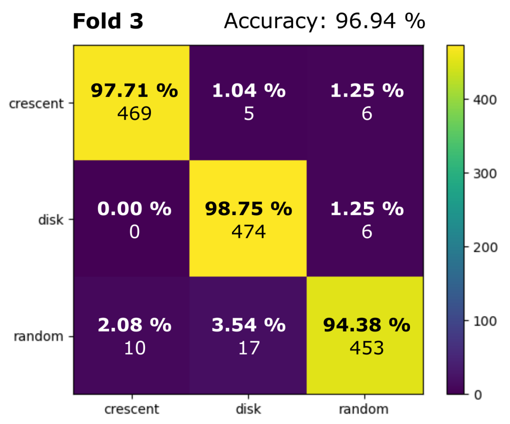
Adaptive Boosting Classifier for *Dataset-a (FFT)*



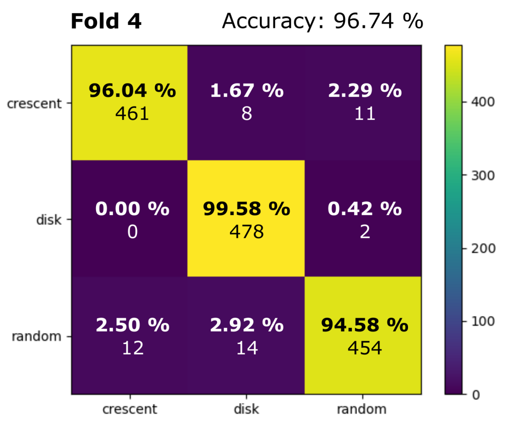
(a)



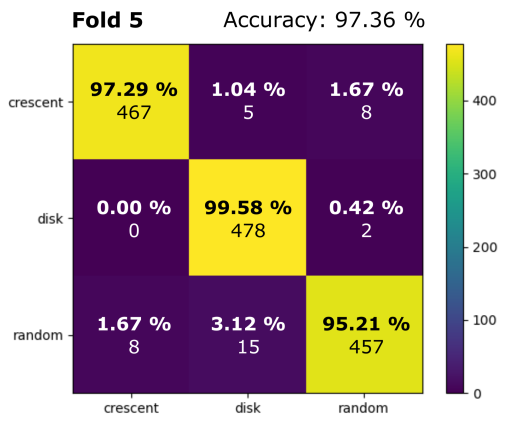
(b)



(c)



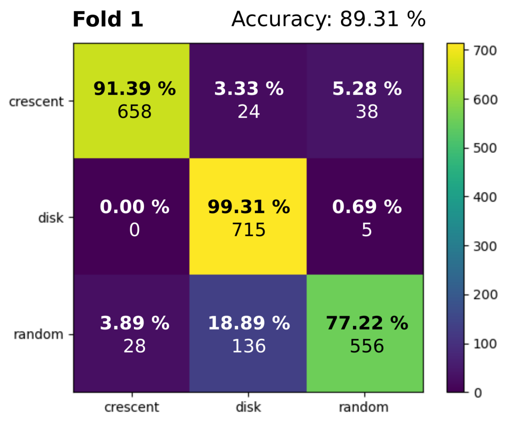
(d)



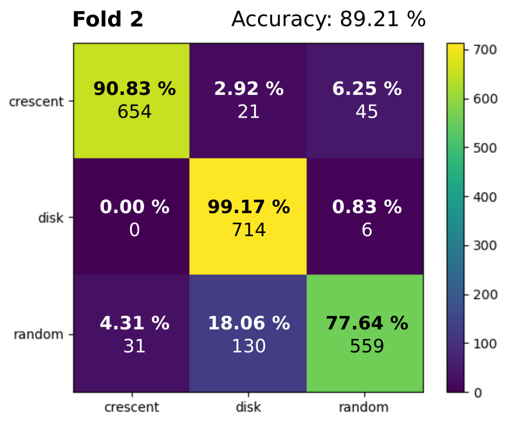
(e)

Figure 23: Confusion matrices and accuracies achieved for folds 1-5 for the classification of *dataset-a (FFT)* with *AdaBoost*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 480, the total number of lightcurves for each label in each test set.

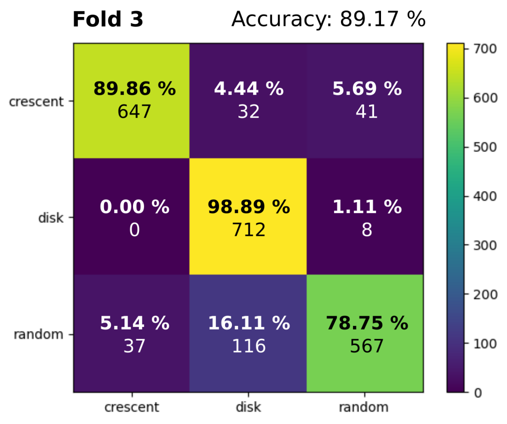
Adaptive Boosting Classifier for *Dataset-b*



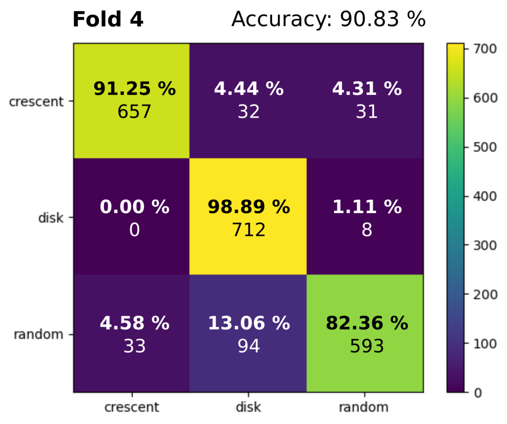
(a)



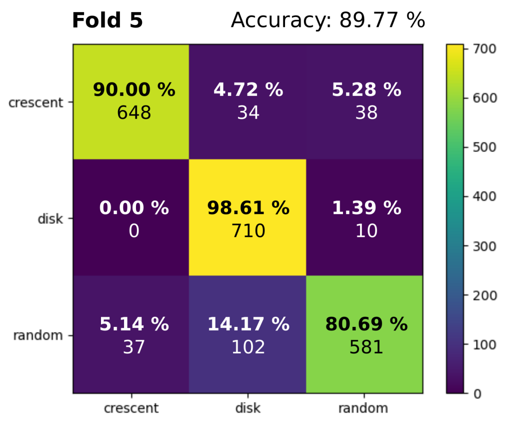
(b)



(c)



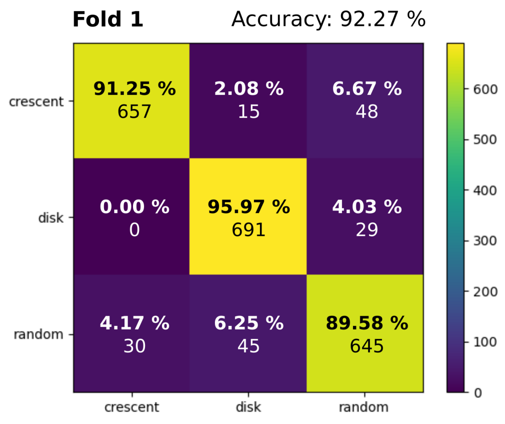
(d)



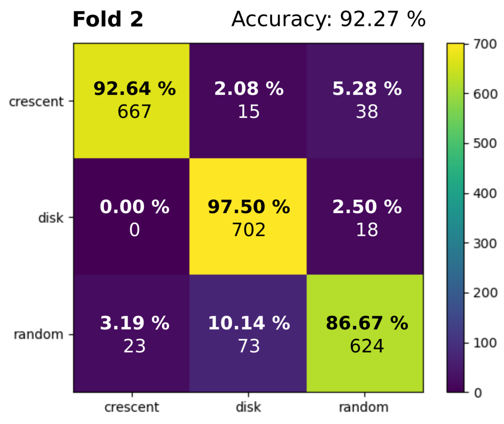
(e)

Figure 24: Confusion matrices and accuracies achieved for folds 1-5 for the classification of *dataset-b* with *AdaBoost*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 720, the total number of lightcurves for each label in each test set.

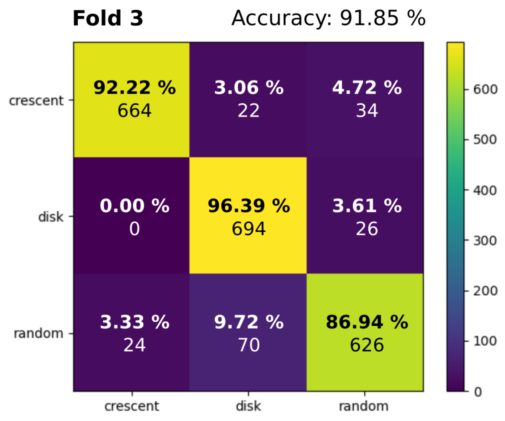
Adaptive Boosting Classifier for *Dataset-b (FFT)*



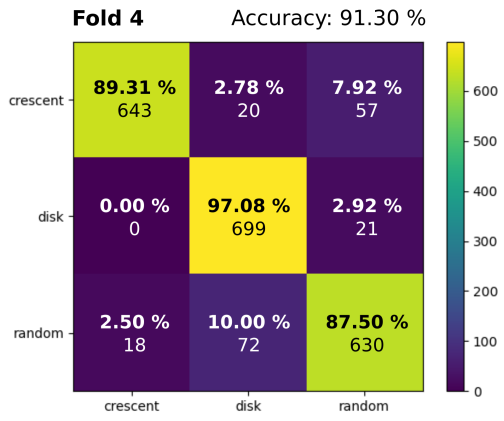
(a)



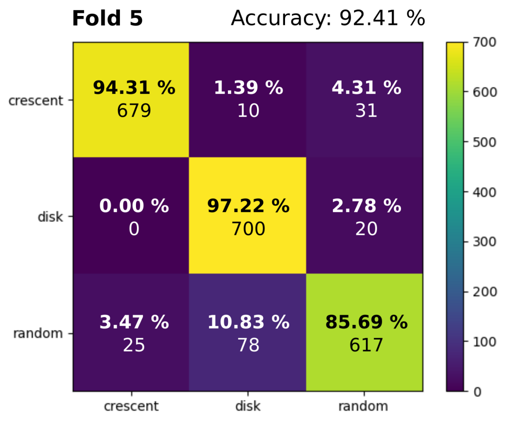
(b)



(c)



(d)

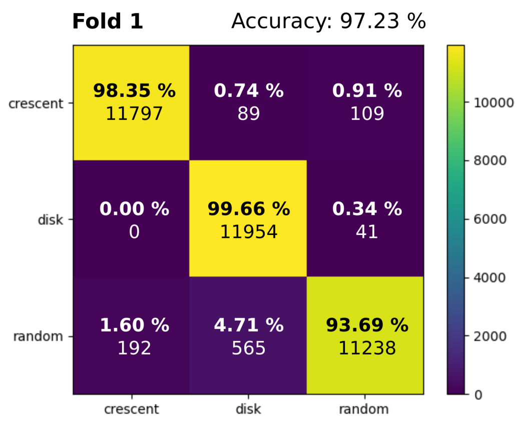


(e)

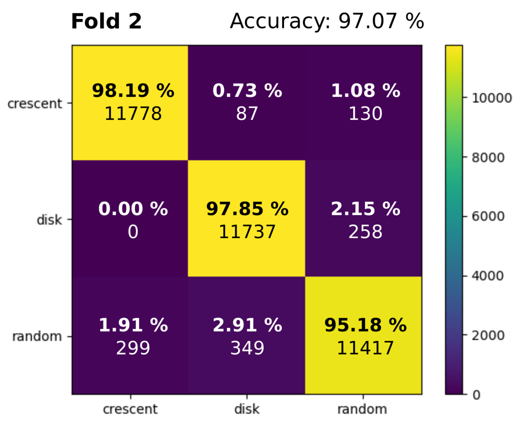
Figure 25: Confusion matrices and accuracies achieved for folds 1-5 for the classification of *dataset-b (FFT)* with *AdaBoost*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 720, the total number of lightcurves for each label in each test set.

Classification with Neural Networks

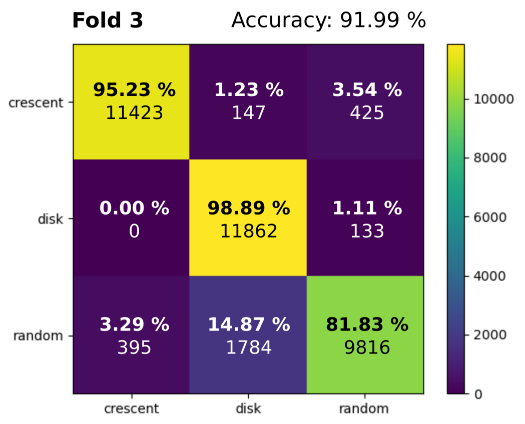
Model-1A: Dense Neural Network for *Dataset-A*



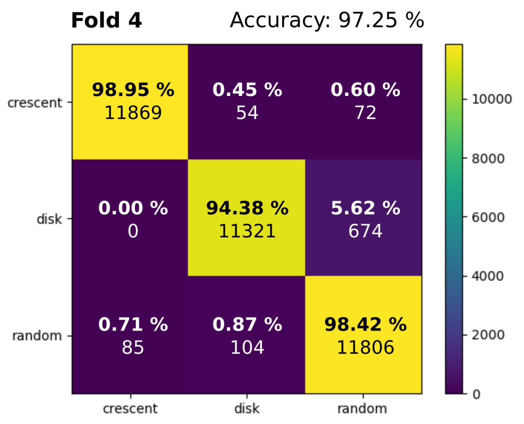
(a)



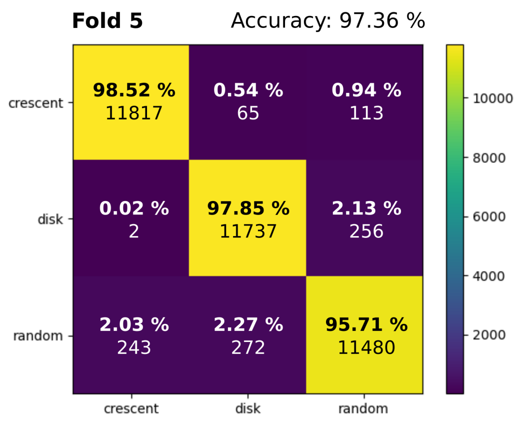
(b)



(c)



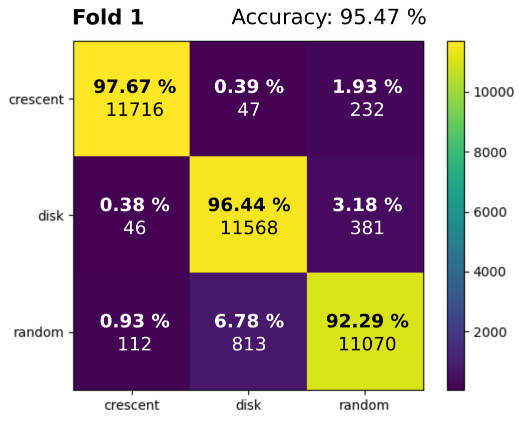
(d)



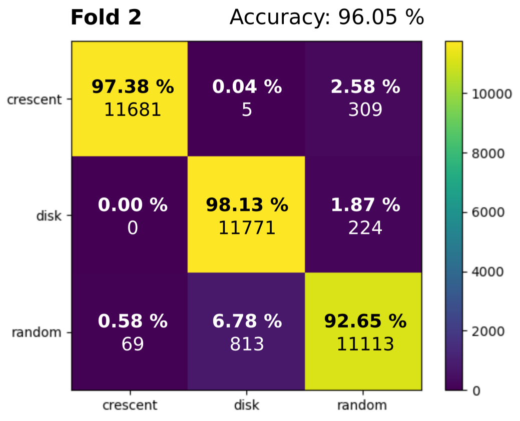
(e)

Figure 26: Confusion matrices and accuracies achieved for folds 1-5 for *model-1A*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 11995, the total number of lightcurves for each label in each test set.

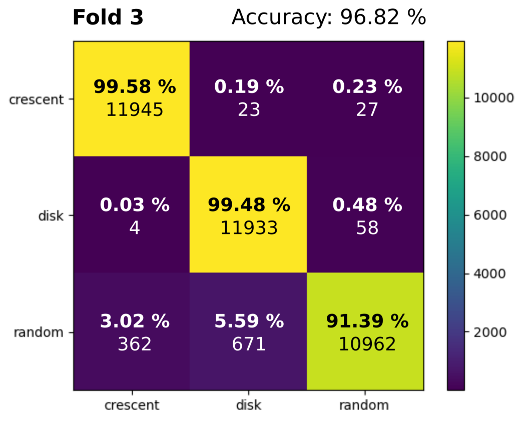
Model-2A: Dense Neural Network for *Dataset-A (FFT)*



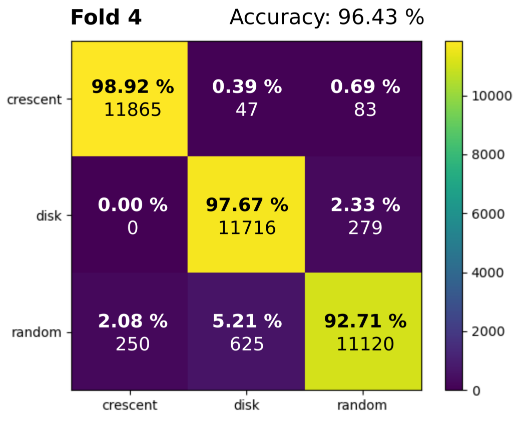
(a)



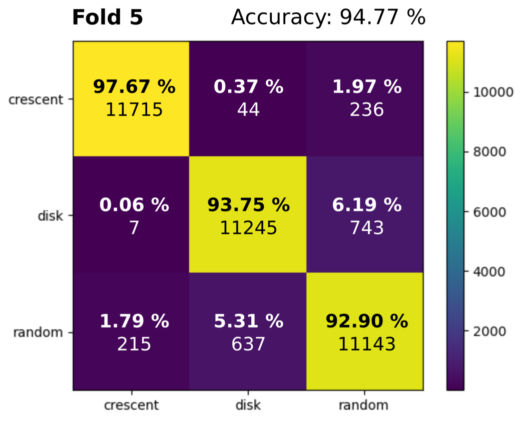
(b)



(c)



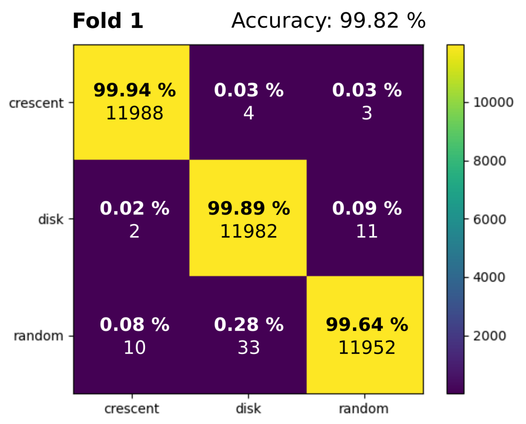
(d)



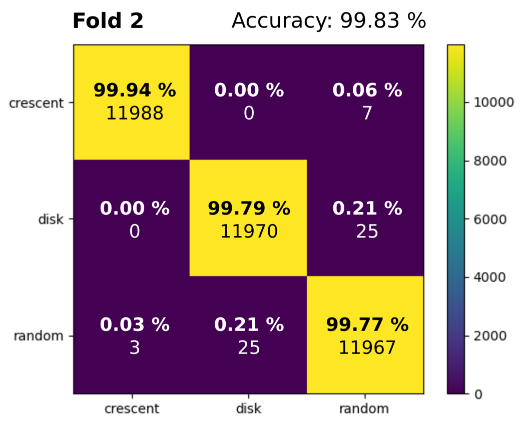
(e)

Figure 27: Confusion matrices and accuracies achieved for folds 1-5 for *model-2A*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 11995, the total number of lightcurves for each label in each test set.

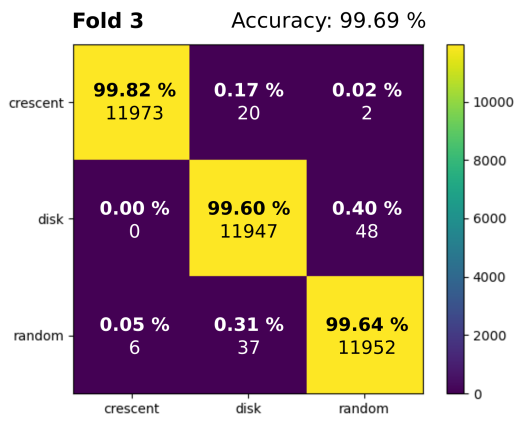
Model-3A: Convolutional Neural Network for *Dataset-A*



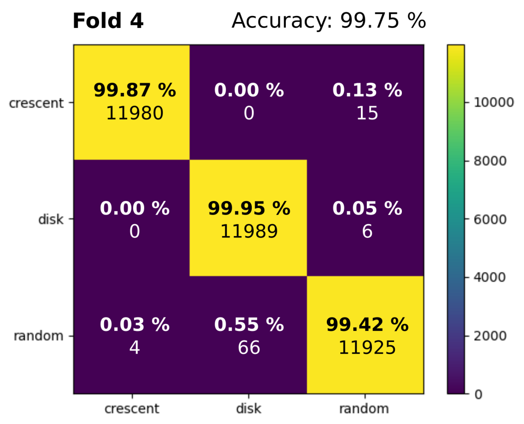
(a)



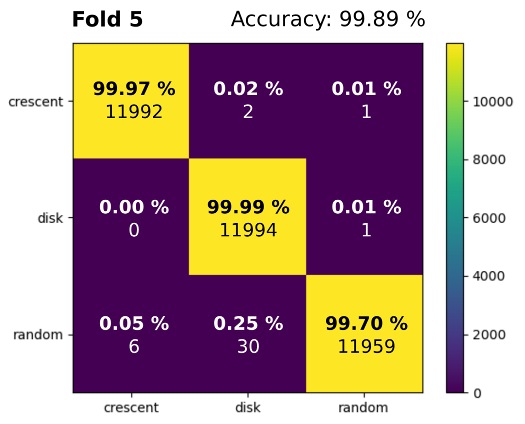
(b)



(c)



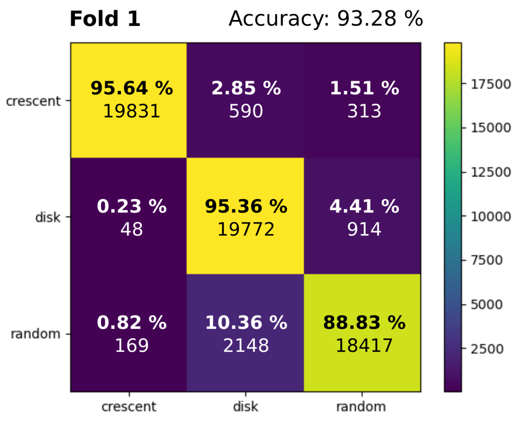
(d)



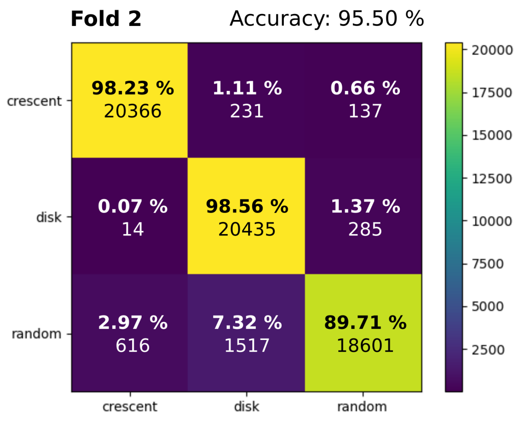
(e)

Figure 28: Confusion matrices and accuracies achieved for folds 1-5 for *model-3A*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 11995, the total number of lightcurves for each label in each test set.

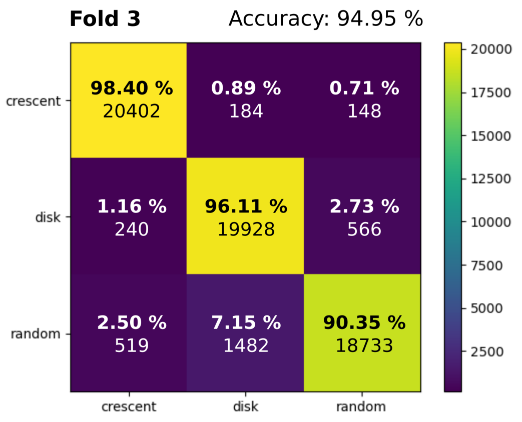
Model-1B: Dense Neural Network for *Dataset-B*



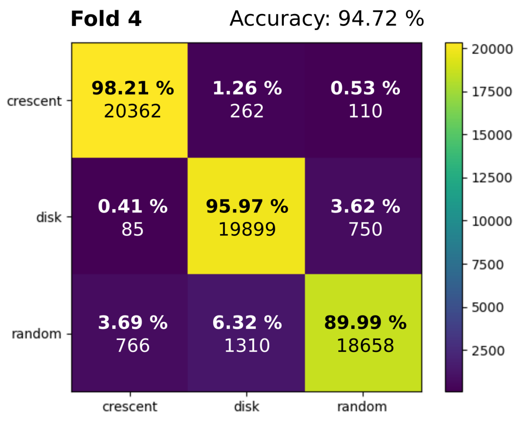
(a)



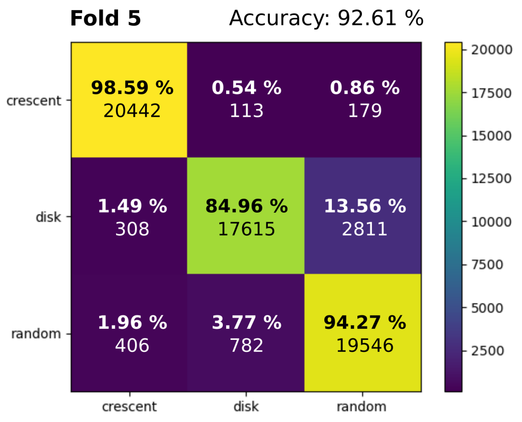
(b)



(c)



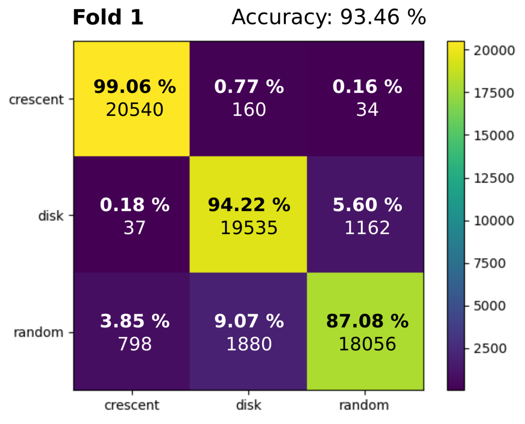
(d)



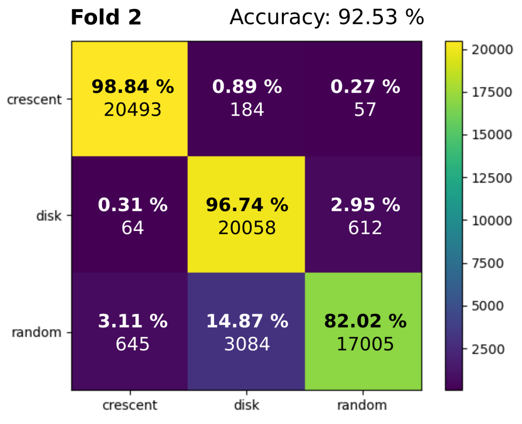
(e)

Figure 29: Confusion matrices and accuracies achieved for folds 1-5 for *model-1B*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 20734, the total number of lightcurves for each label in each test set.

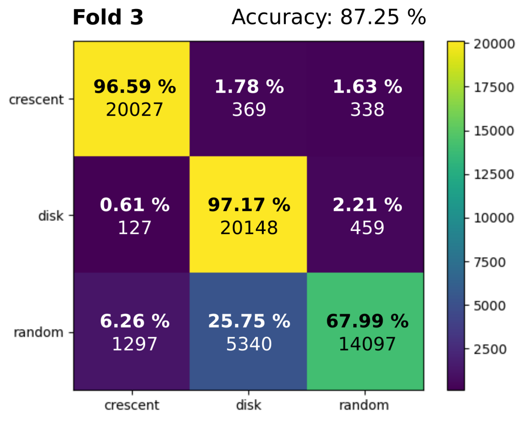
Model-2B: Dense Neural Network for *Dataset-B (FFT)*



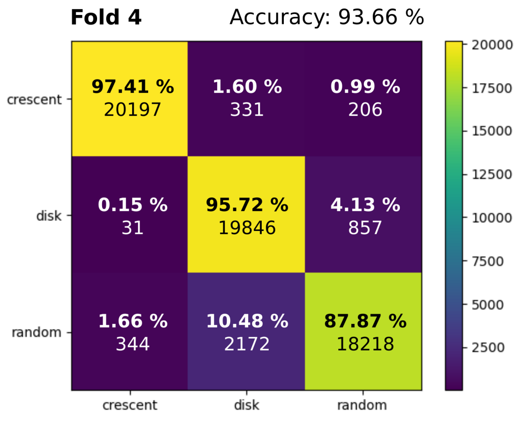
(a)



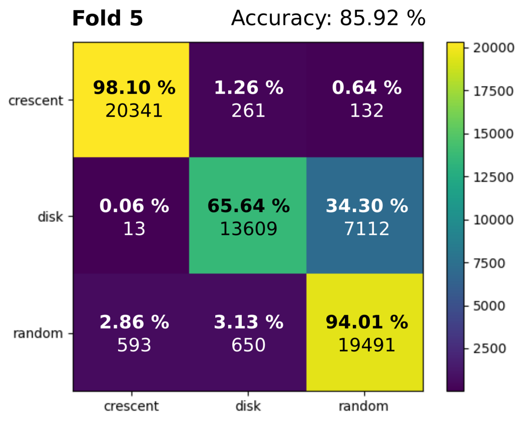
(b)



(c)



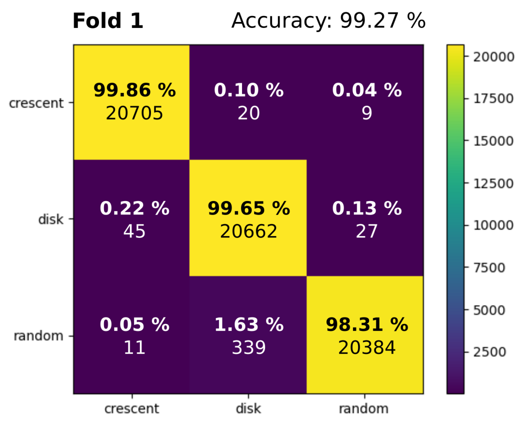
(d)



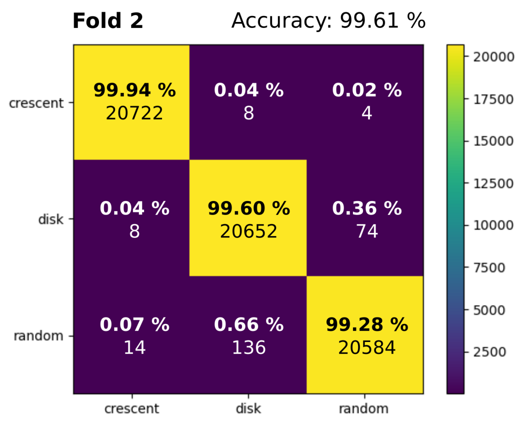
(e)

Figure 30: Confusion matrices and accuracies achieved for folds 1-5 for *model-2B*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 20734, the total number of lightcurves for each label in each test set.

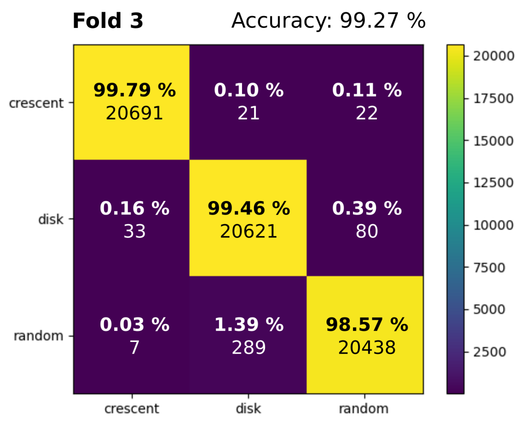
Model-3B: Convolutional Neural Network for *Dataset-B*



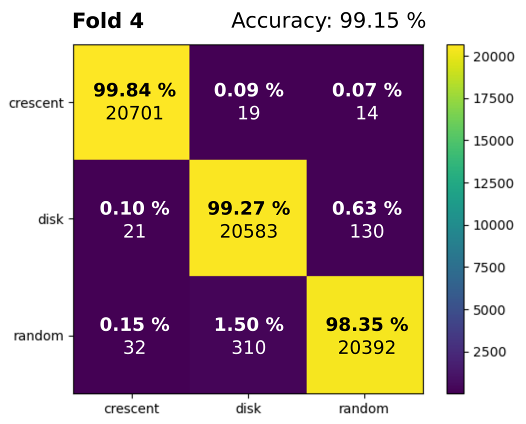
(a)



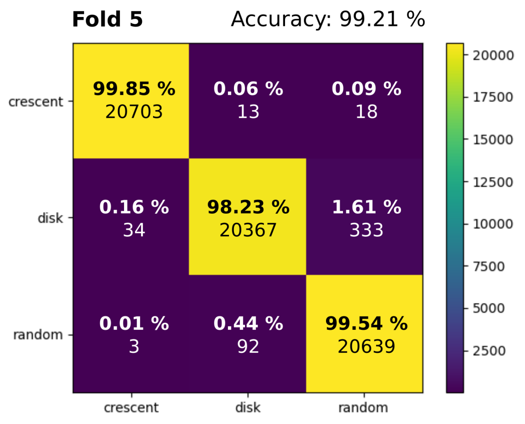
(b)



(c)



(d)



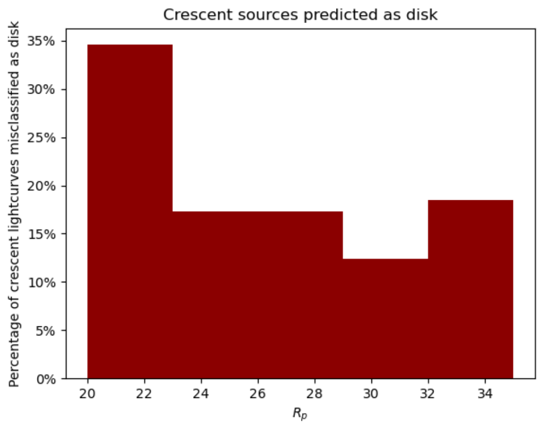
(e)

Figure 31: Confusion matrices and accuracies achieved for folds 1-5 for *model-3B*. For each entry, the absolute value gives the number of lightcurves of a considered label (given on the y-axis) that were classified as a certain label (given on the x-axis). The relative value is the percentage of 20734, the total number of lightcurves for each label in each test set.

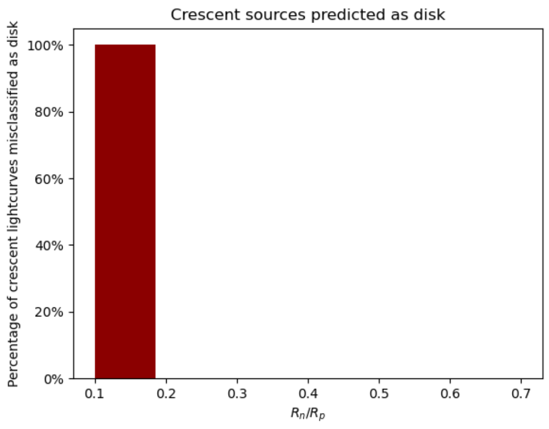
6.3 Misclassified Lightcurves of Model-3B with Regard to Source Properties

In the following, the misclassified lightcurves by *model-3B* vs. source properties are presented. Figures 32 and 33 are histograms illustrating the percentage of the *crescent* lightcurves misclassified as *disk* or *random*, respectively, vs. radii of the sources R_p and radii of the small disk R_n. In figures 34 and 35, the percentage of the *disk* lightcurves or *random* lightcurves, respectively, misclassified as *crescent* vs. R_p and *random* subtypes (for the misclassifications as *random*) are visualised. Figures 36 and 37 show the confusions between *disk* source and *random* source vs. source properties, i.e. disk lightcurves misclassified as *random* vs. R_p and *random* lightcurves misclassified as *disk* vs. R_p and *random* subtypes. The distribution of the lightcurves of a given label from *dataset-B* according to source properties can be found in the appendix A.

Label: Crescent, Prediction: Disk



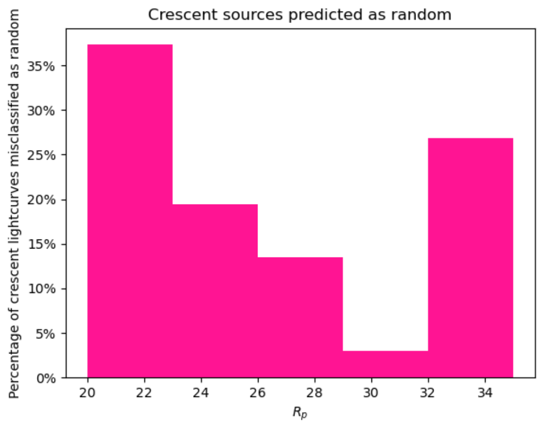
(a)



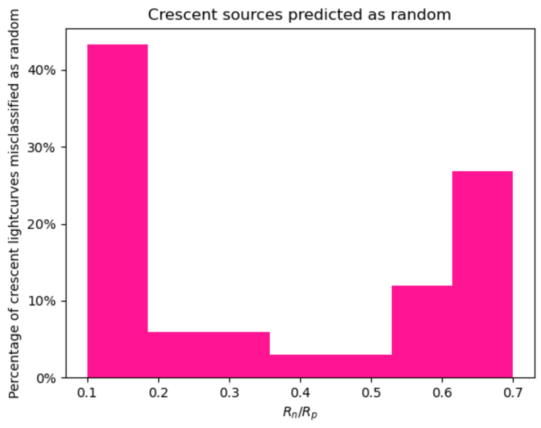
(b)

Figure 32: Histogram showing the percentage of the crescent lightcurves misclassified as disk vs. (a) source radius R_p (b) radius of the small disk R_n .

Label: Crescent, Prediction: Random



(a)



(b)

Figure 33: Histogram showing the percentage of the crescent lightcurves misclassified as random vs. (a) source radius R_p (b) radius of the small disk R_n .

Label: Disk, Prediction: Crescent

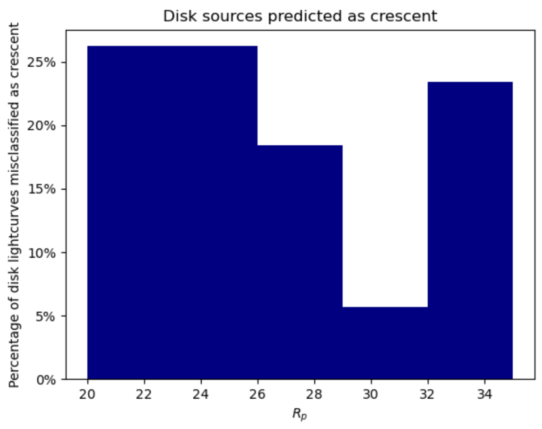
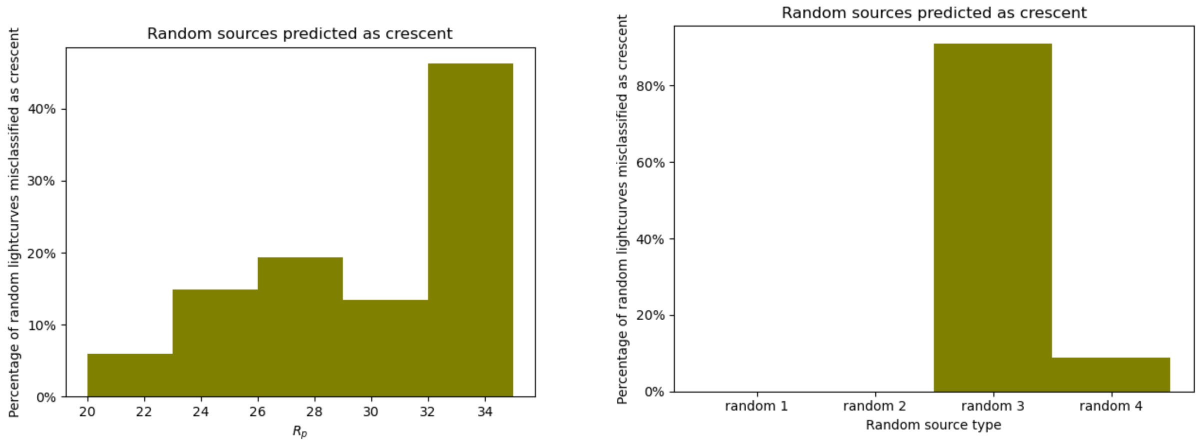


Figure 34: Histogram showing the percentage of the disk lightcurves misclassified as crescent vs. source radius R_p .

Label: Random, Prediction: Crescent



(a)

(b)

Figure 35: Histogram showing the percentage of the random lightcurves misclassified as crescent vs. (a) source radius R_p (b) random subtype.

Label: Disk, Prediction: Random

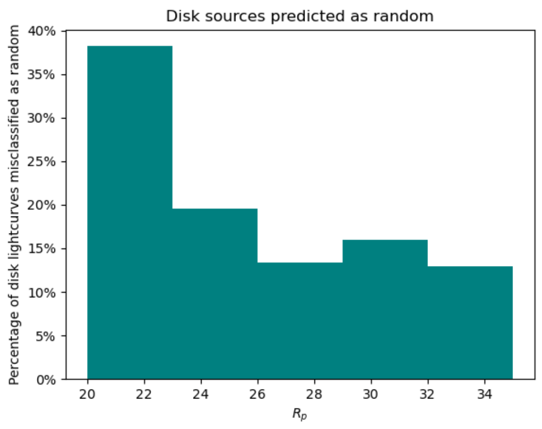
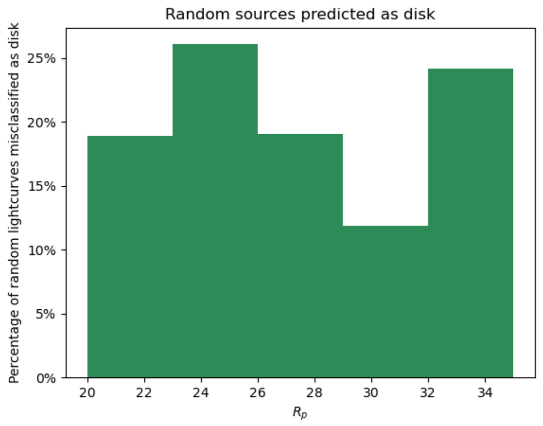
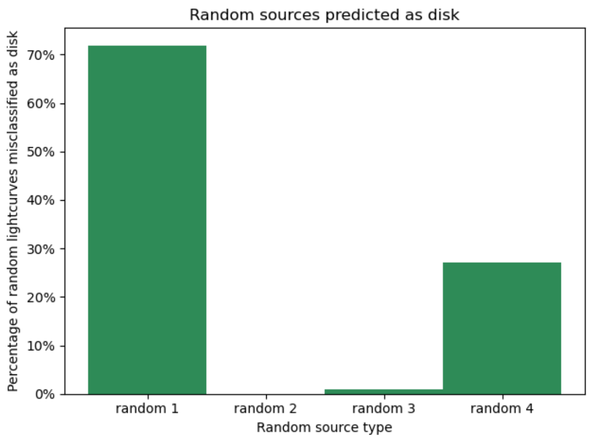


Figure 36: Histogram showing the percentage of the disk lightcurves misclassified as random vs. source radius R_p .

Label: Random, Prediction: Disk



(a)



(b)

Figure 37: Histogram showing the percentage of the random lightcurves misclassified as disk vs. (a) source radius R_p (b) random subtype.

7 Discussion

7.1 Classification with Adaptive Boosting

As discussed in section 5.1, a classification problem with more than three features exceeds the human imagination. Therefore, a classification with *AdaBoost* (see section 3.3.2) is performed first, providing a baseline performance to get an indication of whether the use of machine learning is a helpful tool for our classification problem.

For the classification of *dataset-a* (subset of *dataset-A*) with *AdaBoost*, an average accuracy of (92.28 ± 1.07)% is achieved (see table 7). For the corresponding dataset consisting of the Fourier-transformed lightcurves, the average accuracy is (96.82 ± 0.36)% (see table 8). Thus, for the classification of *dataset-a* with *AdaBoost*, the average accuracy can be increased with a smaller standard devitation by using the Fourier-transforms as features instead of the magnification per timestep. This means, by extraction of information and reduction of the number of features using a Fourier transformation, two things could be achieved. On the one hand, better predictions and, on the other hand, an improved stability of the algorithm for different training and test sets. For the classification of *dataset-b* (subset of *dataset-B*) with *AdaBoost* this is also true, but the average accuracies are lower and the improvement in average accuracy is also less than for *dataset-a*, from (89.66 ± 0.70)% to (92.02 ± 0.45)% (see tables 9 and 10). This leads to the assumption that *AdaBoost* performs less accurately when the data contains noise.

By looking at the confusion matrices (see figures 22 - 25), it can be seen that the *disk* label was predicted most accurately and the *random* label least accurately. The highest number of misclassifications is given by the datapoints with label *random* that were predicted to be *disk*, with the exception of fold 1 of the classification of *dataset-b (FFT)* (figure 25(a)). The datapoints with label *disk* classified as *crescent* yield the least misclassifications. Except for fold 2 of the classification of *dataset-a* (figure 22(b)) where 1 datapoint with label *disk* was predicted as *crescent*, the number of this misclassification is 0 everywhere. The classification of *dataset-a (FFT)* with *AdaBoost* (figure 23) yields already a pretty good accuracy in distinguishing between the labels *crescent* and *disk*. But the confusions in the distinctions between *crescent* and *random* and between *disk* and *random* are noticeable. The distinction between the labels *crescent* and *disk* for *dataset-b (FFT)* is slightly weaker but still good. However, the confusions between *crescent* and *random* and between *disk* and *random* are considerably larger than for *dataset-a (FFT)*.

All in all, it can be said that a classification with *AdaBoost* using the Fourier-transformed data already leads to a high accuracy. The distinction between *crescent* and *disk* is performed in an accurate way, whereas the distinctions between *crescent* and *random* and between *disk* and *random* gives more misclassifications. From the former, it can be seen that using this classifier, a distinction between the lightcurves of an asymmetric source and those of a symmetric source is quite accurate. The latter suggests that the lightcurves of some subtypes of the *random* sources have more similar features with *crescent* lightcurves, whereas the lightcurve features of some other *random* subtypes have more similarities with those of *disk* sources. However, the accuracy is significantly lower for data containing noise, which indicates the limitations of this classifier. Nevertheless, a reasonable baseline performance is obtained, which encourages the assumption that machine learning can be a useful tool for the recognition of quasar lightcurves representing the event horizon.

7.2 Classification with Neural Networks

Classification with Dense Neural Networks

For the classification of *dataset-A* with the two dense neural network models, *model-1A* and *model-2A*, the average accuracies are lower and have larger standard deviations than for the classification with *AdaBoost* for *dataset-a (FFT)* (see tables 11 and 12). Unlike classification with *AdaBoost*, using the Fourier-transformed data does not lead to an improvement in accuracy, but only to an improved stability of the model for different training and test sets. Looking at the accuracies for the different folds for *model-1A* (see table 11), it is immediately noticeable that only fold 3, with an accuracy of 91.99%, has an accuracy of less than 97%. Reducing the number of layers or the number of neurons per layer for *model-1A* decreases the accuracy. Increasing the number of layers or the number of neurons results in an even more unstable model. If the model is run several times with different random initialisations for fold 3, i.e. the same partitioning of the training and test set as for fold 3, an accuracy of over 97% is achieved in most cases, but an accuracy of less than 92% is obtained about 20% of the time. If this procedure is carried out for the remaining four folds, the outcomes are similar. Most likely, the optimiser is unable to find an optimal minimum and gets stuck in a local minimum. Looking at the confusion matrices of the two dense neural network models (see figures 26 and 27), it can be seen that compared to the classification with *AdaBoost*, these two models yield on average a significant improvement in the recognition of *crescent* lightcurves, but a deterioration in the recognition of *disk* lightcurves. Likewise the distinction between the labels *crescent* and *disk* is performed with a high accuracy and the distinctions between *crescent* and *random* and between *disk* and *random* gives more misclassifications. The *random* label was predicted least accurately. However, in fold 4 of *model-1A* (figure 26(d)) the label *random* is predicted very accurately and the label *disk* is predicted significantly less accurately.

Comparing the average accuracies for the classification of *dataset-B* with the two dense neural network models with that of the classification with *AdaBoost* two things are noticeable. *Model-1B*, which uses the magnification values per timestep as features, provides an improvement in accuracy but is less stable (see table 14) compared to the respective classification with *AdaBoost*. For the classification by using the Fourier-transformed lightcurves, *model-2B* performs significantly less accurate and less stable (see table 15) than *AdaBoost*. From the comparison of the two dense neural network models for *dataset-A* with that for *dataset-B*, it can be seen that the average accuracy is considerably lower for data containing noise. Especially using the Fourier-transformed data, the difference is significant. This leads to the assumption that the Fourier modes where the lightcurves of the different sources differ most from each other have an overlap with the Fourier modes of the random noise. Looking at the confusion matrices of the two dense neural network models for *dataset-B* (see figures 29 and 30), it can be seen that the *crescent* label was predicted most accurately. Comparing with the dense neural network models for *dataset-A*, a deterioration in the recognition of *disk* lightcurves with a noticeably larger confusion between *disk* and *random* can be recognised. This implies that with a dense neural network the distinction between *disk* and *random* is performed with a considerably lower accuracy for data containing noise. However, the distinction between *crescent* and *disk* is still quite accurate.

Classification with Convolutional Neural Networks

Performing a classification with a convolutional neural network results in a significantly high average accuracy with a very small standard deviation for both datasets. The achieved average accuracies are for *model-3A*: $(99.80 \pm 0.08)\%$ (see table 13), and for *model-3B*: $(99.30 \pm 0.18)\%$ (see table 16). Thus, using convolutional neural networks for the classification of the lightcurves provides the most accurate predictions and yields the most stable models for different training and test sets, even for data containing noise. This demonstrates the power of a convolutional neural network. With the use of different filters and subsequent compression of the information and by optimising the weights, the features are learned by itself. Thus, relevant features are extracted. Hence, noise can be cancelled out.

A remarkable accuracy in the recognition of *crescent* lightcurves is noticeable when looking at the confusion matrices (see figures 28 and 31). The labels *disk* and *random* were predicted also with a high accuracy. In *model-3A* (figure 28), each label in each fold was predicted correctly with an accuracy $\geq 99.42\%$, in particular the label *crescent* with an accuracy $\geq 99.82\%$. The datapoints with label *random* that were predicted to be *disk* yield also for this model the highest number of misclassifications. Though, at most only 0.55% of *random* lightcurves are classified as *disk*. In *model-3B* (figure 31), in each fold *crescent* lightcurves were predicted correctly with an accuracy $\geq 99.79\%$. The confusion between *disk* and *random* is slightly larger than in *model-3A* but still small. At most 1.63% of *random* lightcurves are classified as *disk* and at most 1.61% of *disk* lightcurves as *random*. Hence, this leads to the assumption that when using the convolutional neural network models, the lower accuracy for the classification of the dataset containing noise compared to that for the classification of the dataset without noise is mostly due to the less accurate distinction between *disk* and *random*.

Misclassified Lightcurves of *Model-3B* with Regard to Source Properties

Examining the properties of the sources that provide the misclassified lightcurves of *model-3B* under consideration (see figures 32 - 37), the following observations can be noted.

First we look at *crescent* lightcurves that were not predicted to be *crescent*. Most of the *crescent* lightcurves that have been misclassified as *disk* are of sources with small source radii, R_p, and their dark disk radii as a fraction of source radii, R_n/R_p, are also small (see figure 32). The percentage of *crescent* lightcurves misclassified as *disk* is about 35% for $20 \leq R_p \leq 23$, given in units of the length of one pixel. Larger source radius categories respectively account for not more than 18%. For 100% of the *crescent* lightcurves misclassified as *disk*, $R_n/R_p \leq 0.2$ can be noticed. Hence, this leads to the suggestion that in the features used by the convolutional neural network of *model-3B*, *crescent* lightcurves have a greater correspondence with *disk* lightcurves when the *crescent* source has very small R_p and exclusively very small R_n relative to R_p. For the *crescent* lightcurves predicted as *random*, for both R_p and R_n/R_p, a peak can be seen for the smallest values and for the largest values (see figure 33). More than 35% have source radii $20 \leq R_p \leq 23$ and for more than 25% the source radii are $32 \leq R_p \leq 35$. For the small disk radii we find for the sources of significantly more than 40% of the lightcurves: $R_n/R_p \leq 0.2$, and of more than 26% of the lightcurves: $0.6 < R_n/R_p < 0.7$. Thus, it can be assumed that *crescent* lightcurves bear a higher resemblance in the features to *random* lightcurves for very small and very large *crescent* source radii and for very small and very large relative dark disk radii.

We next take a look at lightcurves that do not have the label *crescent*, but have been classified as *crescent*. Looking at the *disk* lightcurves misclassified as *crescent*, all source radius categories account for a percentage of more than 18% respectively, except for source radii $29 \leq R_p \leq 32$, which constitute only slightly more than 5% (see figure 34). The latter indicates that for this source radius category *disk* lightcurves have a smaller correspondence with *crescent* lightcurves in the features used by the convolutional neural network. Regarding the *random* lightcurves predicted as *crescent*, significantly more than 40% have source radii $32 \leq R_p \leq 35$ and smaller source radius categories respectively account for not more than 20% (see figure 35)). It can also be seen that about 90% of the *random* lightcurves missclassified as *crescent* are of the *random-3* subtype and about 10% of the *random-4* subtype. The subtypes *random-1* and *random-2* are not misclassified as *crescent*. Therefore, it can be deduced that *random-3* subtype lightcurves bear the most resemblance in the features to *crescent* lightcurves, particularly for very large source radii.

Finally, we look at the confusions between the labels *disk* and *random*. For the *disk* lightcurves misclassified as *random* we find that about 38% of the sources have source radii $20 \leq R_p \leq 23$ and larger source radius categories respectively constitute not more than 20% (see figure 36). This leads to the supposition that *disk* lightcurves have a higher similarity in the features with *random* lightcurves for very small source radii. Looking at the *random* lightcurves misclassified as *disk*, it can be noticed that none of the source radius categories are particularly conspicuous as was the case for the previous misclassifications (see figure 37). However, it can be seen that with over 70% of the *random* lightcurves predicted as *disk*, the subtype *random-1* was most often classified as *disk*. Subtype *random-4* accounts for more than 25% of the *random* lightcurves predicted as *disk* and *random-3* less than 2%. Subtype *random-2* is not misclassified as *disk*. Hence, it can be assumed that *random-1* subtype lightcurves bear the most resemblance in the features and *random-4* subtype lightcurves a slight resemblance in the features to *disk* lightcurves.

Summary

Compared to *AdaBoost*, a dense neural doesn't provide an improvement in average accuracy and stability for the classification of the dataset without noise. For the classification of the dataset including noise, a dense neural network perform more accurately compared to *AdaBoost*. Particularly, a significant improvement in the recognition of *crescent* lightcurves is achieved. However, this is not the case when the Fourier-transforms are used as features.

Using a convolutional neural network results in a very accurate classification and yields a stable model for both datasets. A very high accuracy can be obtained even for the dataset with noise, thanks to the convolution and pooling layer extracting relevant features and thus cancelling out noise. The lower accuracy for the classification of *dataset-B*, the dataset containing noise, compared to that for the classification of *dataset-A*, that one without noise, is mostly induced by the confusion between *disk* and *random* lightcurves. *Crescent* lightcurves are predicted with a very high accuracy. Hence, the convolutional neural network models used in the present work, provide a very useful tool for the recognition of quasar lightcurves representing the event horizon in this simple model, even for data containing noise. For the simulation of the lightcurves of *dataset-B*, a more realistic magnification map was used than for *dataset-A*. This suggests that the CNN model can perform accurately in the classification of lightcurves also when considering more realistic lensing galaxy macromodels and number of individual lensing stars.

Model-3B, which is the CNN model used for the classification of the dataset generated using the more realistic magnification map and including noise, appears to be sensitive to source radii and dark disk radii. The majority of *crescent* lightcurves misclassifications are found for very small source radii and for very small dark disk radii relative to source radii. However, a slightly high percentage of misclassifications are also observed for very large source radii and for very large relative dark disk radii. The largest proportion of *random* lightcurves missclassified as *crescent* can be seen for very large source radii. Among the subtypes of *random* sources, almost exclusively only *random-3* lightcurves are predicted as *crescent*.

8 Conclusion

Quasar microlensing lightcurves enable to probe the structure of the central engine, i.e. of the supermassive black hole and the surrounding accretion disk. In the present work, we investigate whether deep learning provides a possibility for event horizon recognition through quasar microlensing lightcurves and if a neural network model can be found for its classification. For this purpose, a classification is defined where the lightcurves of three different labels, given by simple geometric models with constant surface brightness, are to be predicted. The considered source models are a *crescent* source introduced by Kamruddin & Dexter (2013), a *disk* source, and a *random* source. The *crescent* source represents the silhouette of the event horizon over an accretion disk. The model of a constant brightness *disk* is commonly used in literature to describe the luminous parts of quasars. Two different magnification maps, representing different lensing galaxy macromodels and number of individual lensing stars, were used for the simulation of the lightcurves, providing two different datasets for the classification. The first magnification map is a simplification representing an atypically simple region of a realistic magnification map and the second one is more realistic. The resulting lightcurves when using the more realistic magnification map additionally include random Gaussian noise representing observational noise. The classification of each dataset is performed with *AdaBoost* and with three different neural network models.

The classification with *AdaBoost* already leads to a reasonable baseline performance and indicates the usability of machine learning for the recognition of quasar lightcurves representing the event horizon. But the limitations of this classifier become noticeable when the data contains noise. Futhermore, the approach to extract information and reduce the number of features by using the Fourier-transforms of the lightcurves as features turns out to be beneficial only with the *AdaBoost* classifier for the dataset which doesn't include noise. Thus, this approach is not effective for applications with deep learning and is unusable for real conditions.

A dense neural network, by using the magnification per timestep as features, provides an improvement in the performance for the classification of the dataset including noise compared to *AdaBoost*. In particular an improvement in recognition of *crescent* lightcurves, and hence, in event horizon recognition is noticeable. However, the accuracy achieved is significantly lower than with the convolutional neural network model.

The two CNN models prove to be the most accurate and most stable for the respective dataset. For the dataset without noise the achieved accuracy is $(99.80 \pm 0.08)\%$ and for the dataset generated with the more realistic magnification map and including noise, $(99.30 \pm 0.18)\%$. Therefore, it can be deduced that the classification of the lightcurves with a CNN model lead to high accuracies even when considering a more realistic lensing galaxy macromodel and number of lensing stars and by simulation of observational noise. The lower accuracy for the classification of the dataset including noise is mostly due to the less accurate distinction between *disk* and *random*. The model performs very accurately in distinguishing between *crescent* and non-*crescent* and thus, in the recognition of the lightcurves representing the event horizon.

Moreover, the CNN model is sensitive to source radii, R_p, and dark disk radii, R_n. The recognition of *crescent* lightcurves is less accurate for very small and very large R_p and for very small and very large R_n/R_p. In the work of Tomozeiu et al. (2017), these abstract *crescent* parameters are related to physical quantities. They suggested that R_p gives an estimate of the size of the accretion disk and R_n provides an estimate of the gravitationally magnified Schwarzschild radius of the black hole. Since the gravitationally magnified Schwarzschild radius is a monotonic function of the black hole's mass, it can be used to estimate the latter, assuming a not rotating black hole. The *crescent* sources, whose lightcurves are misclassified as *disk* lightcurves, have mostly small R_p and have exclusively small R_n/R_p. This indicates that the lightcurves representing the event horizon silhouette over an accretion disk bear a higher resemblance in the features used by the model to lightcurves representing a luminous accretion disk, in the mentioned parameter sizes. Hence, for a very small disk size or for a very small magnified Schwarzschild radius relative to the disk radius, the event horizon is recognised less good, given our simple model.

Ultimately it can be concluded that the convolutional neural network models presented in this work perform very accurately in the recognition of lightcurves representing the event horizon, in our simplification. Even for a more realistic lensing galaxy macromodel and number of lensing stars and when considering noise, a CNN model is found that achieves a very high accuracy. Thus, deep learning provides a promising tool for the recognition of the event horizon through quasar microlensing lightcurves.

Future efforts should be directed towards a classification for a dataset consisting of lightcurves simulated with different magnification maps. This would allow us to determine whether a CNN model can be used for the recognition of lightcurves representing the event horizon, regardless of the different lensing galaxy macromodels and number of individual lensing stars.

For further investigations, it should be examined how sensitive the model is to the orientation of the source to the caustic fold. The orientation is given by the components of the disk centres displacement, c, perpendicular and parallel to the caustic fold. However, the value of the disk centres displacement parallel to the caustic fold doesn't have an effect on the lightcurve shape. The relevant orientation parameter can be determined by the angle at which the path on which the source is moving intersects the caustic fold in the magnification map. For this purpose, the algorithm developed in this work would need to be extended. Since the caustic folds are not straight lines but have a curvature, the calculation of the angle would have to be done using an approximation. This angle can be also used to determine the component of the relative velocity between the source and the caustic fold that is perpendicular to the caustic fold, v_p. Hence, a sensitivity to the orientation would represent a sensitivity to v_p in our simple model. Furthermore, the sensitivity of the CNN model to the *crescent* parameters in the classification of the lightcurves provides a useful prerequisite with regard to finding a model for the prediction of the *crescent* parameters from the lightcurves. The approach for the prediction of the *crescent* parameters from the lightcurves with deep learning would then be a regression task and not a classification task, as it has been in this work. As mentioned, the *crescent* parameters can be related to physical quantities specific to the central engine of quasars. Thus, a model that predicts these quantities from lightcurves would be a powerful tool for the investigation of quasars.

9 Acknowledgements

The master thesis is a meaningful step towards scientific work, which is why it is a fortiori important to have a good mentor. Therefore, I would like to express my warmest thanks to Prof. Dr. Prasenjit Saha, as on the one hand I could always count on his help and great support during my work and on the other hand he gave me the chance to work independently and a bit on my own responsibility. That helped me a lot to develop myself. Also, I would like to thank my friends very much for proofreading. From the bottom of my heart, I would also like to thank my family, especially my nieces and nephews, for their mental support and encouragement. I would like to thank the University of Zurich, especially the Physics Institute and the Institute for Computational Science for the excellent education I received. They enabled me to make my childhood dream come true.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[2] E. Agol and J. Krolik, "Imaging a quasar accretion disk with microlensing," vol. 524, pp. 49–64, oct 1999.

[3] T. Anguita, R. W. Schmidt, E. L. Turner, J. Wambsganss, R. L. Webster, K. A. Loomis, D. Long, and R. McMillan, "The multiple quasar q2237+0305 under a microlensing caustic*," *A&A*, vol. 480, no. 2, pp. 327–334, 2008.

[4] R. Antonucci, "Unified models for active galactic nuclei and quasars.," , vol. 31, pp. 473–521, Jan. 1993.

[5] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," , vol. 324, pp. 446–449, Dec. 1986.

[6] M. Bartelmann and P. Schneider, "Weak gravitational lensing," *Physics Reports*, pp. 291–472, 2001.

[7] R. Blandford and R. Narayan, "Fermat's principle, caustics, and the classification of gravitational lens images," *Astrophys. J.; (United States)*, vol. 310, 11 1986.

[8] K. Chang and S. Refsdal, "Flux variations of QSO 0957+561 A, B and image splitting by stars near the light path," *Nature*, vol. 282, pp. 561–564, 1979.

[9] F. Chollet, "Keras," 2015.

[10] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, vol. 19, no. 90, pp. 297–301, 1965.

[11] F. Courbin, P. Saha, and P. Schechter, "Quasar lensing," 2002.

[12] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, pp. 303–314, Dec. 1989.

[13] M. D'Onofrio, P. Marziani, and J. W. Sulentic, *Fifty Years of Quasars*. Springer Berlin Heidelberg, 2012.

[14] M. Deru, A. Ndiaye, and G. Press, *Deep Learning mit TensorFlow, Keras und TensorFlow.js*. Onleihe. E-Book, Rheinwerk Verlag GmbH, 2019.

[15] J. Frank, A. King, and D. J. Raine, *Accretion Power in Astrophysics: Third Edition*. 2002.

[16] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory* (P. Vitányi, ed.), (Berlin, Heidelberg), pp. 23–37, Springer Berlin Heidelberg, 1995.

[17] J. Frochte, *Maschinelles Lernen*. München: Hanser, 2 ed., 2019.

[18] B. Gaudi and A. Petters, "Gravitational microlensing near caustics i: Folds," *ApJ*, vol. 574, 12 2001.

[19] B. S. Gaudi and A. O. Petters, "Gravitational microlensing near caustics. II. cusps," *The Astrophysical Journal*, vol. 580, pp. 468–489, nov 2002.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics, Springer, 2009.

[21] P. Hiep and R. Joo, "A deep learning approach for classification of cloud image patches on small datasets.," *Journal of Information and Communication Convergence Engineering*, vol. 16, pp. 173–178, 09 2018.

[22] T. K. Ho, "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, (M), pp. 278–282, IEEE Computer Society, 1995.

[23] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[24] A. B. Kamruddin and J. Dexter, "A geometric crescent model for black hole images," *Monthly Notices of the Royal Astronomical Society*, vol. 434, pp. 765–771, 07 2013.

[25] A. Kembhavi and J. Narlikar, *Quasars and Active Galactic Nuclei: An Introduction*. Cambridge University Press, 1999.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[27] C. S. Kochanek, "Quantitative Interpretation of Quasar Microlensing Light Curves," , vol. 605, pp. 58–77, Apr. 2004.

[28] Y. Lecun, *Generalization and network design strategies*. Elsevier, 1989.

[29] C. Lemaréchal, "Cauchy and the gradient method," 2012.

[30] B. W. Lyke, A. N. Higley, J. N. McLane, D. P. Schurhammer, A. D. Myers, A. J. Ross, K. Dawson, S. Chabanier, P. Martini, N. G. Busca, H. du Mas des Bourboux, M. Salvato, A. Streblyanska,

P. Zarrouk, E. Burtin, S. F. Anderson, J. Bautista, D. Bizyaev, W. N. Brandt, J. Brinkmann, J. R. Brownstein, J. Comparat, P. Green, A. de la Macorra, A. M. Gutiérrez, J. Hou, J. A. Newman, N. Palanque-Delabrouille, I. Pâris, W. J. Percival, P. Petitjean, J. Rich, G. Rossi, D. P. Schneider, A. Smith, M. Vivek, and B. A. Weaver, "The sloan digital sky survey quasar catalog: Sixteenth data release," vol. 250, p. 8, aug 2020.

[31] E. Mediavilla, J. Jiménez-vicente, J. A. Muñoz, and T. Mediavilla, "Resolving the Innermost Region of the Accretion Disk of the Lensed Quasar Q2237+0305 through Gravitational Microlensing," , vol. 814, p. L26, Dec. 2015.

[32] S. Mollerach and E. Roulet, *Gravitational Lensing and Microlensing*. World Scientific, 2002.

[33] C. W. Morgan, C. S. Kochanek, N. D. Morgan, and E. E. Falco, "The Quasar Accretion Disk Size-Black Hole Mass Relation," , vol. 712, pp. 1129–1136, Apr. 2010.

[34] L. Moroney, *AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence*. O'Reilly, 2020.

[35] C. Nguyen and O. Zeigermann, *Machine Learning : kurz & gut*. O'Reillys Taschenbibliothek, O'Reilly, 2018.

[36] D. Osinga, *Deep Learning Cookbook: Practical Recipes to Get Started Quickly*. O'Reilly Media, 2018.

[37] I. Pâris, P. Petitjean, Aubourg, N. P. Ross, A. D. Myers, A. Streblyanska, S. Bailey, P. B. Hall, M. A. Strauss, S. F. Anderson, and et al., "The sloan digital sky survey quasar catalog: tenth data release," *Astronomy Astrophysics*, vol. 563, p. A54, Mar 2014.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[39] A. Petters, H. Levine, and J. Wambsganss, *Singularity theory and gravitational lensing*. Birkhäuser Boston, 01 2001.

[40] S. Poindexter and C. S. Kochanek, "Microlensing evidence that a type 1 quasar is viewed face-on," *The Astrophysical Journal*, vol. 712, p. 668–673, Mar 2010.

[41] S. Poindexter, N. Morgan, and C. S. Kochanek, "The Spatial Structure of an Accretion Disk," , vol. 673, pp. 34–38, Jan. 2008.

[42] D. Pooley, S. Rappaport, J. A. Blackburne, P. L. Schechter, and J. Wambsganss, "X-Ray and Optical Flux Ratio Anomalies in Quadruply Lensed Quasars. II. Mapping the Dark Matter Content in Elliptical Galaxies," , vol. 744, p. 111, Jan. 2012.

[43] T. Rashid, *Make Your Own Neural Network*. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 1st ed., 2016.

[44] H. Reall, “Lecture notes, part 3 black holes,” tech. rep., Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 2020.

[45] R. Schmidt and J. Wambsganss, “Quasar microlensing,” *General Relativity and Gravitation*, vol. 42, pp. 2127–2150, 09 2010.

[46] P. Schneider, J. Ehlers, and E. Falco, *Gravitational Lenses*. Astronomy and Astrophysics Library, Springer, 1999.

[47] P. Schneider, C. Kochanek, and J. Wambsganss, *Gravitational Lensing: Strong, Weak and Micro*. Springer, Berlin, Heidelberg, 2006.

[48] P. Schneider and A. Weiss, “The gravitational lens equation near cusps,” , vol. 260, pp. 1–13, July 1992.

[49] K. Schwarzschild, “Über das Gravitationsfeld eines Massenpunktes nach der Einsteinschen Theorie,” *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin*, pp. 189–196, Jan. 1916.

[50] N. I. Shakura and R. A. Sunyaev, “Reprint of 1973A&A....24..337S. Black holes in binary systems. Observational appearance.,” , vol. 500, pp. 33–51, June 1973.

[51] D. Sluse, D. Hutsemékers, F. Courbin, G. Meylan, and J. Wambsganss, “Microlensing of the broad line region in 17 lensed quasars,” *Astronomy Astrophysics*, vol. 544, p. A62, Jul 2012.

[52] M. Tomozeiu, I. Mohammed, M. Rabold, P. Saha, and J. Wambsganss, “Microlensing as a possible probe of event-horizon structure in quasars,” *Monthly Notices of the Royal Astronomical Society*, vol. 475, pp. 1925–1936, 12 2017.

[53] C. M. Urry and P. Padovani, “Unified Schemes for Radio-Loud Active Galactic Nuclei,” , vol. 107, p. 803, Sept. 1995.

[54] J. Wambsganss, “Gravitational lensing in astronomy,” *Living reviews in relativity*, vol. 1, no. 1, p. 12, 1998.

[55] J. Wambsganss, “Gravitational lensing: numerical simulations with a hierarchical tree code,” *Journal of Computational and Applied Mathematics*, vol. 109, no. 1, pp. 353–372, 1999.

[56] J. Wambsganss, “Microlensing of Quasars,” *Publ. Astron. Soc. Austral.*, vol. 18, p. 207, 2001.

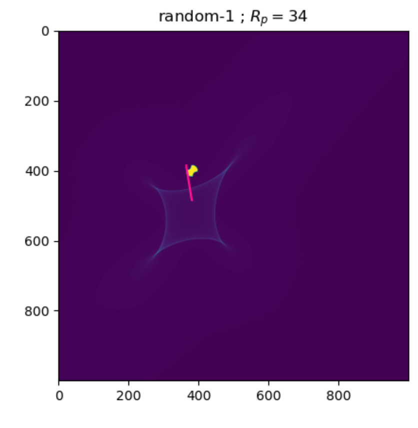
[57] J. Wambsganss, “Gravitational microlensing,” *Gravitational Lensing: Strong, Weak and Micro*, p. 453–540, 2006.

[58] X.-B. Wu, F. Wang, X. Fan, W. Yi, W. Zuo, F. Bian, L. Jiang, I. D. McGreer, R. Wang, J. Yang, and et al., “An ultraluminous quasar with a twelve-billion-solar-mass black hole at redshift 6.30,” *Nature*, vol. 518, p. 512–515, Feb 2015.

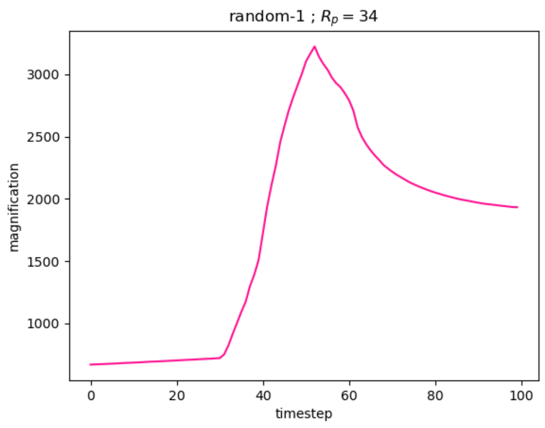
A Appendix

Examples Random Subtypes

Example of a Random-1 Source from Dataset A



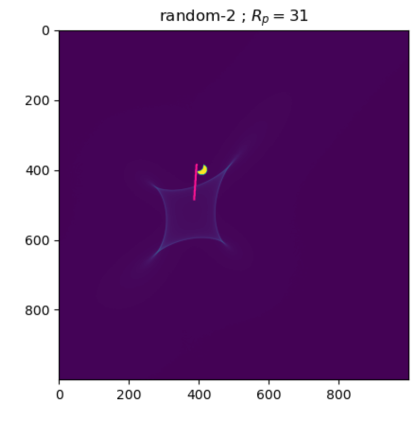
(a)



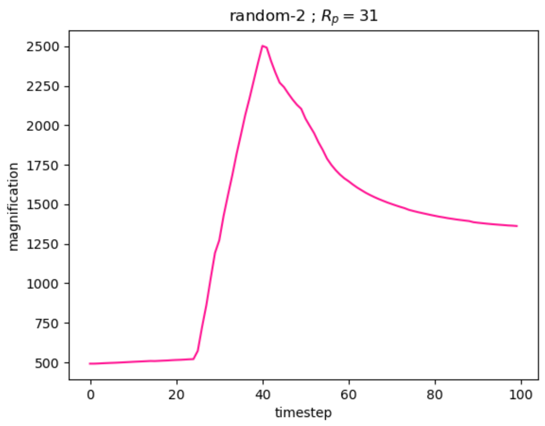
(b)

Figure 38: (a) Example for random-1 source moving along a path in magnification map A. (b) The resulting lightcurve.

Example of a Random-2 Source from Dataset A



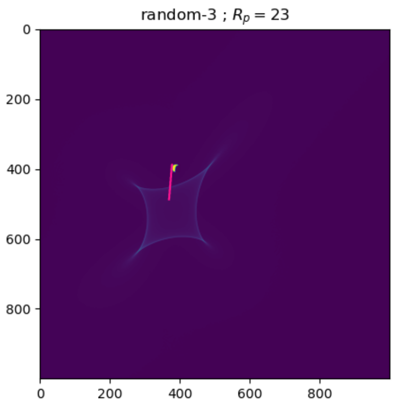
(a)



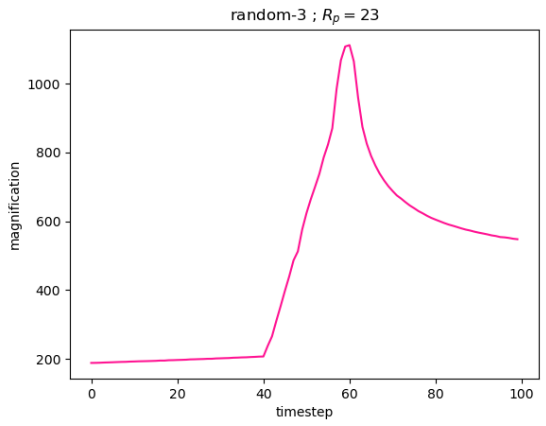
(b)

Figure 39: (a) Example for random-2 source moving along a path in magnification map A. (b) The resulting lightcurve.

Example of a Random-3 Source in Magnification Map A



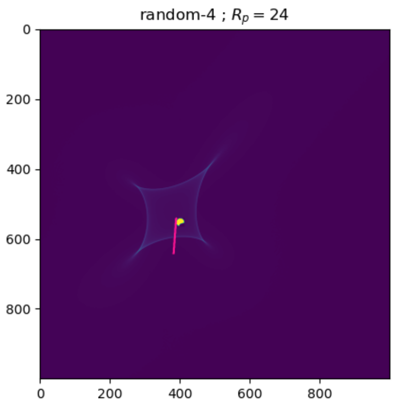
(a)



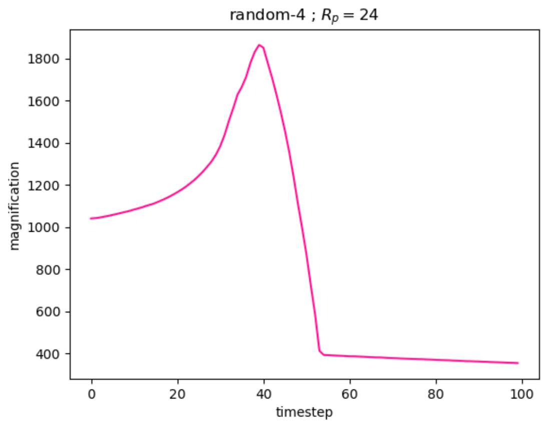
(b)

Figure 40: (a) Example for random-3 source moving along a path in magnification map A. (b) The resulting lightcurve.

Example of a Random-4 Source from Dataset A



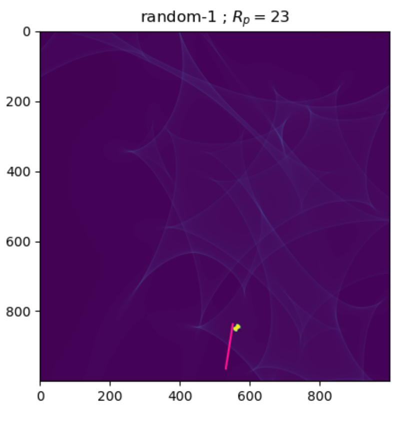
(a)



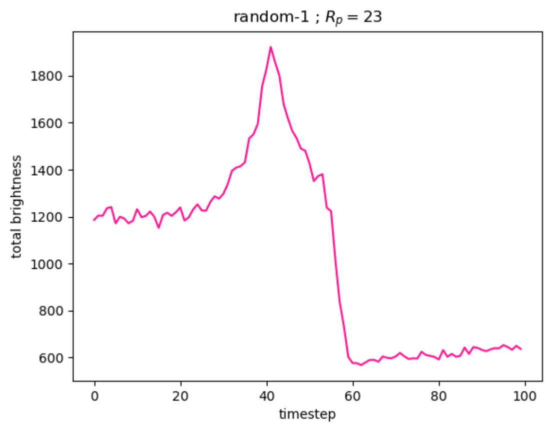
(b)

Figure 41: (a) Example for random-4 source moving along a path in magnification map A. (b) The resulting lightcurve.

Example of a Random-1 Source from Dataset B



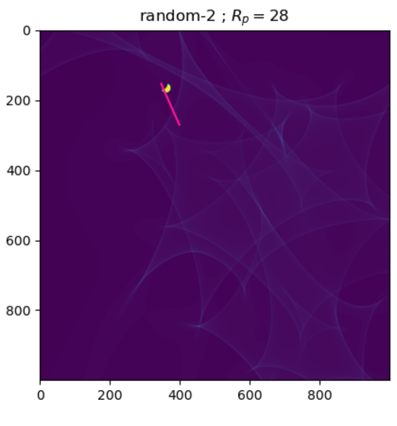
(a)



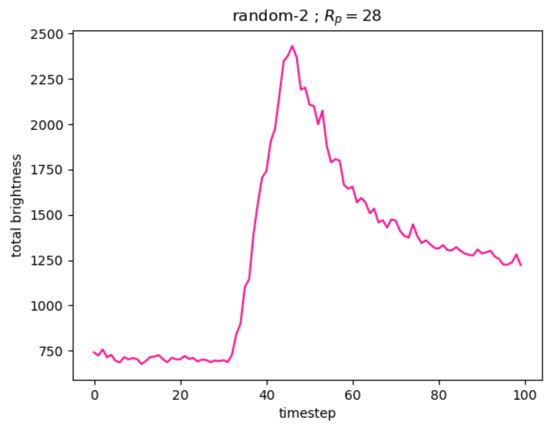
(b)

Figure 42: (a) Example for random-1 source moving along a path in magnification map B. (b) The resulting lightcurve with added noise.

Example of a Random-2 Source from Dataset B



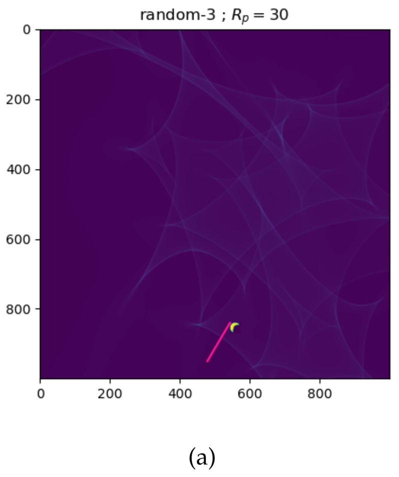
(a)



(b)

Figure 43: (a) Example for random-2 source moving along a path in magnification map B. (b) The resulting lightcurve with added noise.

Example of a Random-3 Source from Dataset B



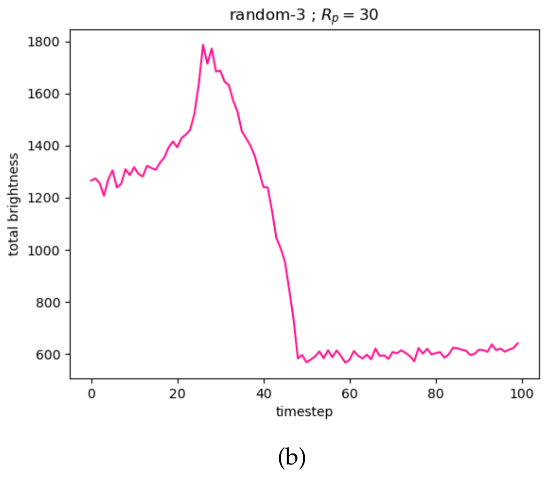
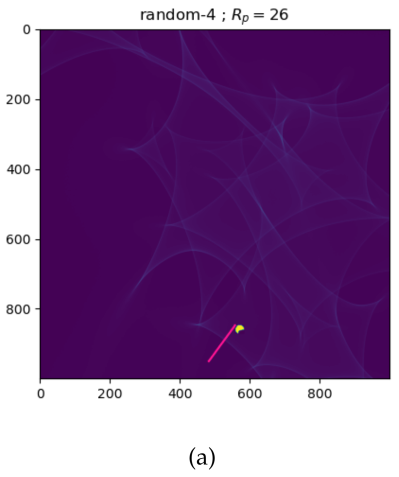


Figure 44: (a) Example for random-3 source moving along a path in magnification map B. (b) The resulting lightcurve with added noise.

Example of a Random-4 Source from Dataset B



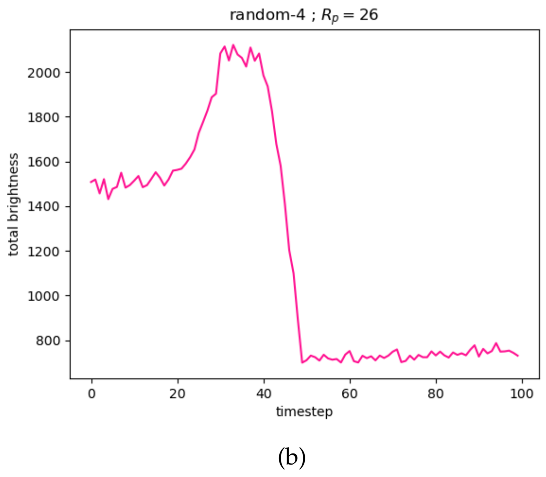
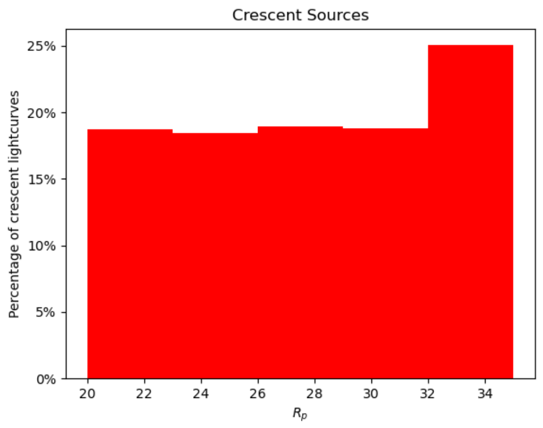


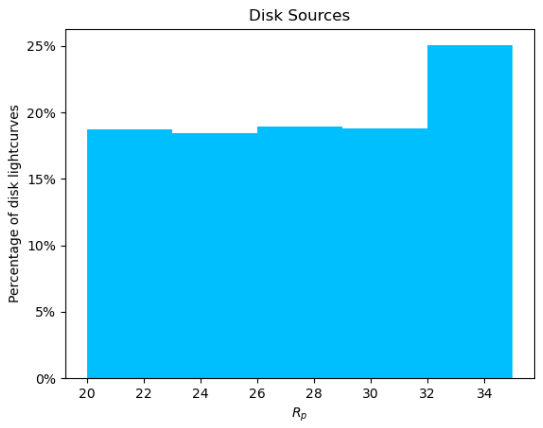
Figure 45: (a) Example for random-4 source moving along a path in magnification map B. (b) The resulting lightcurve with added noise.

Lightcurves from Dataset-B by Source Properties

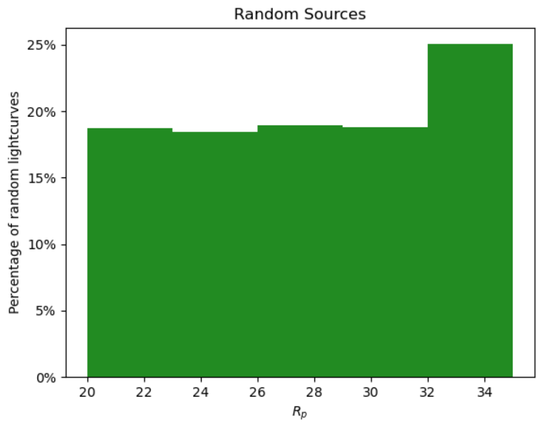
Lightcurves by Source Radius R_p



(a)



(b)



(c)

Figure 46: Percentage of lightcurves by source radius for the three source labels (a) crescent (b) disk (c) random .

Crescent Lightcurves by Small Disk Radius R_n

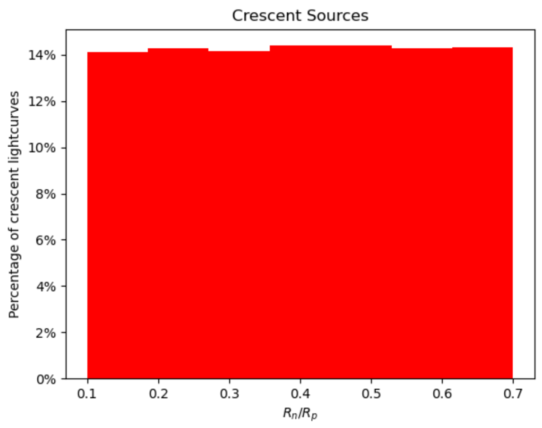


Figure 47: Percentage of crescent lightcurves by small disk radius R_n.

Random Lightcurves by Subtypes

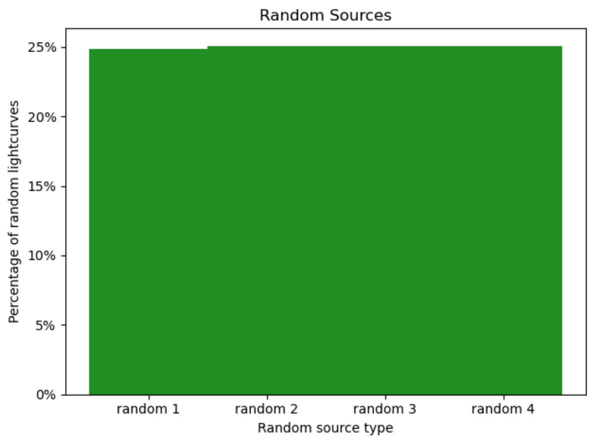


Figure 48: Percentage of random lightcurves by subtypes.