Modeling Microlensing System with Deep Learning Methods

Zerui Liu, Bonan Pu, Wei Zhu, Shude Mao

August 30, 2021

Abstract

The gravitational lensing effect has a wide range of applications in cosmology and the search for exoplanets. And the study of microlensing effects caused by binary stars is of great value in the search for exoplanets. Estimating the physical parameters of lensing systems is an important task in the study of binary star lensing, and the deep learning methods developed in recent years offer the possibility of greatly increasing the efficiency of parameter estimation. We use convolutional neural networks (CNN) to investigate the possibility of using deep learning for parameter estimation and provide some assessment of the effectiveness and promise of this approach. To solve the common degeneracy problem in microlensing, we further explored the effectiveness of mixture density network (MDN) for modeling microlensing systems, and analyzed the advantages and problems faced. After extensive experiments, we found that deep learning methods have a fairly high processing speed in terms of modeling efficiency, but accuracy can be greatly limited on current low-quality datasets. In the future, we will explore more advanced deep learning methods and investigate the possibility of running deep learning methods on higher quality data.

Contents

1 Introduction

It is an important task in the field of gravitational lensing to find microlensing events from the observation data of astronomical telescopes and predict the parameters of the lensing system based on the lightcurves. Convolutional neural network (CNN) is the most common deep learning method in the field of image recognition and processing nowadays. If the lightcurves of microlensing events are considered as one-dimensional images, identifying the features of the lightcurves could be seen as a task of image recognition.

The traditional methods of detection of possible microlensing events rely on match filtering searching over a grid of models, and parameter estimation relies on grid search or Monte Carlo Markov Chain method (MCMC). We mainly pay our attention on the parameter estimation of the microlensing system. For parameter estimation, grid search or MCMC will still yield the most accurate possible result, but CNN will offer a close estimate of the true parameter in orders of magnitude less computation time.

In addition, the microlensing degeneracy, mainly the close-wide degeneracy, has a large impact on the output of the network. Traditional neural network, which only regress a set of single physical values, fails to give a accurate predicted results for events with microlensing degeneracy. To deal with this type of difficulty, mixture density network (MDN) is applied to our work. This kind of network would give a appromixation of the probability distribution in parameter space. When degeneracy occurs, the positions of the parameter space corresponding to several sets of solutions will have a large probability density distribution.

We generate simulation datasets using `Mulensmodel` [1],a python package, and use these datas to train our networks. The trained neural network will be applied to real data from the Korean Microlensing Telescope Network (KMTNet) project [2]. Details of dataset generation will be introduced in section 2. Several structures of CNN networks and corresponding results will be introduced in section 3. The result of MDN will be introduced in section 4 The comprehensive performance evaluation of CNNs and future work will be discussed in Section 5.

2 Generation of Simulation Datasets

2.1 Generation of Lightcurve

We generate mock lightcurves using `Mulensmodel`, a python package used for microlensing modeling, which uses the advanced contour integration algorithm. The probability of low-mass planets showing detectable signatures is very small. In order to increase the sampling efficiency, we use the empirical parameterization of [3].The main parameters of interest to us for modeling are as follows:

- q: The mass ratio of binary star system producing microlensing event. Initially the mass ratio q is chosen from a log-uniform distribution from 10^{-4} to 10^{0}.
- s: The projected separation between the lens primary and its companion as a fraction of the Einstein ring radius. The projected separation s is chosen from a log-uniform distribution between 0.3 and 3.
- u_0: The impact parameter between the source trajectory and the lens center of mass. For the impact parameter u_0, it is chosen to be uniform between 0 and some maximum impact parameter value (temporally chosen to be 1) for large mass-ratio ($q > 10^{-3}$) events; otherwise it is given by the method of [3].
- α: The angle between the source trajectory and the binary axis. Angle α is chosen from a uniform distribution on the interval [0, 2π].
- ρ: The radius of the source as a fraction of the Einstein ring. ρ is chosen from a log-uniform distribution between 10^{-4} and 10^{-2}.
- t_0: The time of the closest approach between the source and the lens.
- t_E: The Einstein radius crossing time.

These parameters enable the calculation of a binary magnification curve. The magnification curve is then converted to the light curve in magnitudes for given source baseline magnitude m_0 and blending fraction f_b (temporally chosen to be zero, namely no blending.

$$m(t) = m_0 - 2.5 \log_{10}\{(1 - f_b)[A(t) - 1] + 1\} \tag{1}$$

m_0 is drawn from a normal distribution with a mean of 18 and a standard deviation of 1.5.

For the lightcurves in simulation dataset, in order to quantify the detectability of the binary perturbation, we will need to know the single-lens model that best matches this binary light curve. At first we use the following parameters to generate the corresponding single-lens light curve:

- The PSPL peak time $t_{0,\mathrm{pspl}} = t_0 - \frac{q}{1+q} s t_E \cos\alpha$.

- The PSPL impact parameter $u_{0,\mathrm{pspl}} = u_0 - \frac{q}{1+q} s \sin\alpha$.
- The PSPL Einstein crossing time $t_{\mathrm{E,PSPL}} = t_E$.

The corrections to t_0 and u_0 take into account the different definitions of these parameters in the binary and PSPL models: For binary they are defined relative to the center of mass whereas for PSPL they are to the single lens.

The detectablility of the binary perturbation is quantified by

$$\Delta\chi^2 \equiv \sum_{\{t_i\}} \left(\frac{m_{\mathrm{binary}}(t_i) - m_{\mathrm{pspl}}(t_i)}{\sigma_m(t_i)} \right)^2 \tag{2}$$

In the traditional approach, a planet is assumed to be detected if $\Delta\chi^2 \sim 200$ [4]. Observationally, the criterion has been set to be higher in order to ensure the detection is genuine ($\Delta\chi^2 > 500$, [5]).

However, simply substituting the parameters to generate PSPL lightcurves instead of performing the time-consuming model fittings might cause some problems. For some lightcurves with deviations in the source trajectory and caustic, the microlensing features are less obvious. There is no difference in morphology between the PSPL lightcurve and the 1S2L lightcurve obtained by replacing the parameters, but there is a significant difference in the absolute magnitude values. It causes $\delta\chi^2$ to be falsely high. If lightcurves with distinct microlensing features are needed, we need to perform the time-consuming model fittings to obtain more accurate $\delta\chi^2$ values for the determination.

2.2 time series and Noise Model

The time series and noise models we initially used are idealized models, i.e., time series using equal time intervals, and a noise model using single-valued functions. When applied to real data, the real data with unequal time intervals will be linearly interpolated to match the simulated data.

But the real data are messier than we could have imagined. Real data often has many intervals with missing data points, as well as some bad data points with too much error. When interpolating, intervals with missing data points can cause significantly distorted intervals (e.g., straight line alignments that would not appear in real data). The noise model using a single-value function makes the resulting lightcurves too clean to produce realistic, non-negligible bad data points in the simulated data, or to simulate the effect of realistic factors on the measurement uncertainty of the data points. The two schemes we have used for the time series and noise models are as follows:

2.2.1 Ideal time series and Noise Model

The time range of lightcurve is chosen from $t_0 - 2t_E$ to $t_0 + 2t_E$. 200 points are inserted evenly and equally spaced into this time range. The characteristic time duration of the microlensing signal can be estimated according to the following equation:

$$\frac{\Delta T}{T_E} \propto \sqrt{q} \tag{3}$$

When 200 data points are taken, the minimum observable mass ratio q is estimated to be roughly of order 10^{-4}. In this case, simulated events with mass ratios greater than 10^{-4} are reliable.

The magnitude uncertainties are assumed to be Gaussian with a magnitude-dependent standard deviation (a noise model based on the simulation of KMTNet project; Figure 8 of [6])

$$\sigma_m = \begin{cases} 10^{0.34(m-16.372)-2} & m \leq 16.732 \\ 10^{0.17(m-16.372)-2} & m \geq 16.732 \end{cases} \tag{4}$$

2.2.2 Realistic time series and Noise Model

To solve the various problems of overly idealized simulation data, a realistic time series and noise model based on the real data of KMTNet is used. Real data from microlensing events that have been made public by KMTNet is downloaded for use. While generating the lightcurves, the required time series is cut directly from the real time data to the desired length. This method ensures that the measurement time of the simulated data matches the real data.

The noise information is also obtained directly from the real data. The uncertainty of data points is obtained directly from the real data. The magnitude and errorbar (σ) of the data points in KMTNet are recorded in advance, and the magnitude axis is divided into a series of subintervals in the magnitude-errorbar space at intervals of 0.1. When generating the simulation data, the number of errorbars is randomly selected within the subinterval according to the location of the subinterval where the magnitude of the data point is located.

Obtaining time series and noise information directly from the real data makes the characteristics of the simulated data more similar to the real data. We first use this method to generate a series of simulated light curves with 1000 data points. The entire time interval is then divided equally into 200 subintervals. The data points falling in each sub-interval are recorded, and the weighted average of magnitude is calculated according to the following equation:

$$m_i = \frac{\sum_j m_i, j/(\sigma_i, j^2)}{\sum_j (1/\sigma_i, j)^2} \tag{5}$$

Using the midpoint of the subinterval as the time value and the calculated magnitude as the magnitude value, a new equally spaced light curve with 200 data points is obtained. If a subinterval falls into no more than 1 data point, then the interval is not computed and is filled by linear interpolation.

This processing method turns unequally spaced time series into equally spaced time series and removes bad data points with large offsets by weighted averaging. This data pre-processing approach allows the neural network to not have to deal with intractable temporal information. And we hope that the method can be easily extended to preprocessing of real observation data. An example of simulated data is shown in Figure.1.

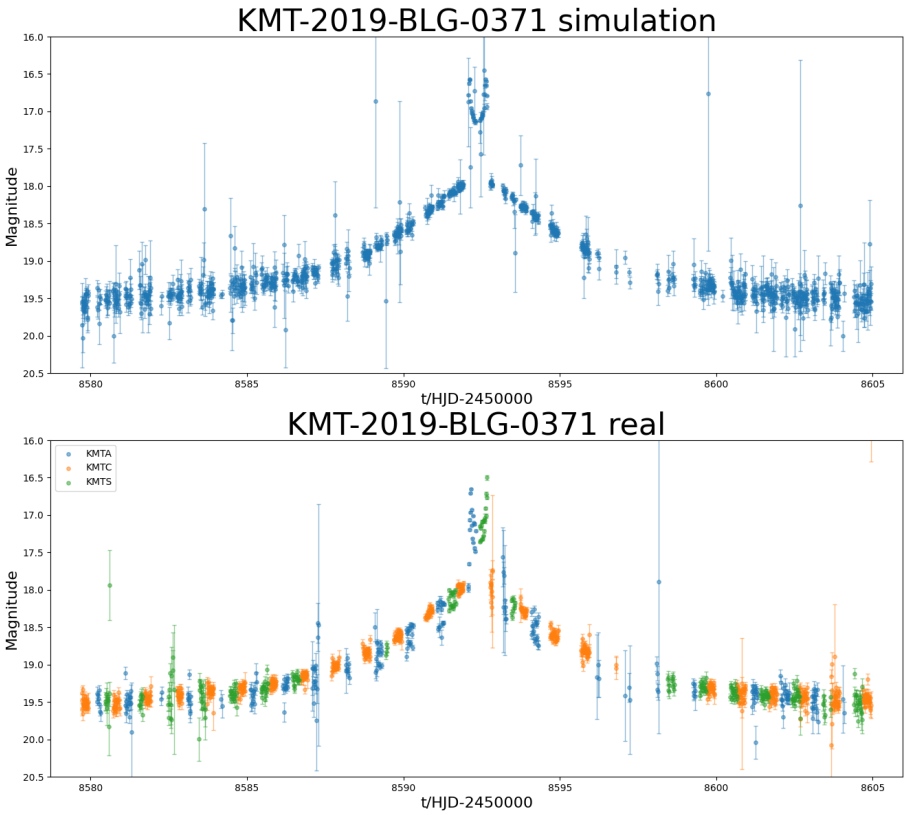


Figure 1: To demonstrate our data simulation, we selected the real binary lensing event KMT-2019-BLG-0371 [7], simulated it using its time sequence and the noisemodel we mentioned in the previous section, and compared it with its original data. It can be seen that they are very similar and our simulation is usable.

3 Structures of CNN Networks and Corresponding Results

Our work can be roughly divided into two parts; the network, which is used in the first part, is trained using the more ideal simulated dataset mentioned earlier in section 2.2.1 and the results are tested. The dataset in the second stage is replaced with the more realistic simulated dataset mentioned earlier in section 2.2.2 for training. We use *Pytorch* to build one-dimensional convolutional neural networks. The training of the neural network is performed on the GPU Cluster *Metis* of Tsinghua Astronomy Department. 80 GPU cores and 4 Nvidia Tesla-V100 GPUs can be called upon for computing.

3.1 Training With the Ideal Simulation Dataset

We use a Convolutional Neural Network with an input layer, 8 convolutional layers and 4 densely connected layers, followed by an output layer,which is used for regression and parameter estimation. The data input to the network is an equally spaced light curve with 200 data points as described earlier in section 2.2.1. The network outputs two parameters about the lensing system: mass ratio q and seperation s. At the time of actual training, the network output is $[\log_{10} q, s]$. The range of $\log_{10} q$ is from -5 to 0, and the range of s is from 0.3 to 3.

We generate 100,000 light curves as the training set and 10,000 light curves as the validation set in the training process according to section 2.2.1.

For parameter estimation, the output layer consists of N linear units, where N is the number of parameters to be estimated. The model is trained by minimizing the mean-square error:

$$MSE = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2, \tag{6}$$

where y is some parameter of interest and $\hat{y}_i$ is its predicted value for the i-th lightcurve.

Training is done using the Adam optimizer, which is a variant of Stochastic Gradient Descent with momentum.

After 500 epochs of training, the MSE loss function of the test set is reduced to about 0.1. The CNN algorithm attains an RMS error of ~ 0.3 for $\log q$ and ~ 0.18 for s. This provides a reference for the accuracy estimation of regression and parameter estimation based on deep learning method. Then we take 2,000 light curves from the validation set and make a scatter plot of the predicted-actual values for comparison.

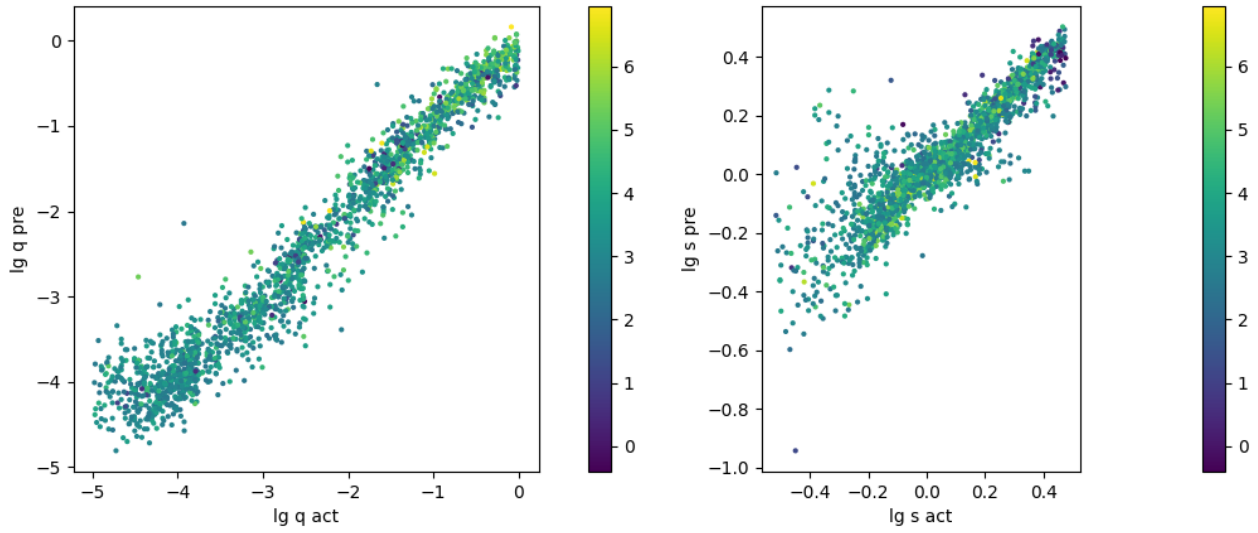


Figure 2: Scatter plot of the predicted-true values, which contains 2,000 simulated microlensing event. The colorbar represents the values of $\log_{10} \Delta\chi^2$ of these events

3.2 Training With the Realistic Simulation Dataset

3.2.1 Network Construction and Training Results

In this stage we train the network with the realistic simulated dataset, which is generated by the method mentioned in section 2.2.2. We generate 200,000 light curves as the training set and 10,000 light curves as the validation set in the training process according to section 2.2.2. We use a Convolutional Neural Network with an input layer, 10 convolutional layers and 4 densely connected layers, followed by an output layer,which is used for regression and parameter estimation.

The data input to the network is pre-processed equally spaced lightcurves with 200 data points each as described earlier in section 2.2.2. Based on the two lensing parameters of the first stage network output, we will add two more parameters to the network output. These two parameters are related to the trajectory of the background source. At first, the impact parameter u_0 and the angle α between the source trajectory and the binary axis are chosen. However, the effect of u_0 and α on the light curve morphology is so different that the network would be difficult to train if these two parameters are predicted directly. Therefore, $u_0 \cos\alpha$ and $u_0 \sin\alpha$ are chose as the two parameters describing source trajectory instead of u_0 and α. In the actual training, the parameters of the network output are $[\log_{10} q, \log_{10} s, u_0 \cos\alpha, u_0 \sin\alpha]$. To facilitate the training of network, these parameters are mapped to the $[0, 5]$ interval by linear deflation.

After 550 epochs of training, the MSE loss function of the test set is reduced to about 0.1. A sample of 1000 simulated events is selected for testing. And the noise is regenerated according to the errorbar of each sample, and 5 more samples with different noise are obtained from each sample. These 5000 lightcurves are fed into the network to get the output parameter prediction values. A scatter plot of the predicted-actual values is drawn to evaluate the network performance.

The MS errors and RMS errors of the predicted parameters attained by the CNN algorithm are shown in the following table.

Parameter	Mean Square Error	Root Mean Square Error
$\log_{10} q$	0.0846	0.291
$\log_{10} s$	0.0176	0.133
$u_0 \cos\alpha$	0.037	0.192
$u_0 \sin\alpha$	0.040	0.199

The MS errors and RMS errors of u_0 and α calculated from $u_0 \cos\alpha$ and $u_0 \sin\alpha$ are shown in the following table. This provides a reference for the accuracy estimation of regression and parameter estimation based on deep learning method.

Parameter	Mean Square Error	Root Mean Square Error
u_0	0.0176	0.133
α	$7441.306((^\circ)^2)$	$86.263(^\circ)$

3.2.2 Large Deviation Points and Degeneracy

The analysis of the scatter plot shows that the main source of MSE error is mainly from the data points that deviate from the midline. It can be seen that the data points are more loosely distributed where $\log_{10} q$ is smaller, and the deviation from the central axis is greater. And from the scatter plot, it can be seen that the predicted value of $\log_{10} s$ has a tendency to converge to 0 incorrectly. Also the predicted values of $u_0 \cos\alpha$ and $u_0 \sin\alpha$ have a tendency to converge to 0 incorrectly, making most of the incorrect predicted values of u_0 around 0.

The deviation of $\log_{10} q$ is easier to understand. It is consistent with the perception that the smaller the mass ratio, the less visible the signal and the more difficult the event is to detect. The deviation of $\log_{10} s$ is presumed to be related to close-wide degeneracy [8]. The effect of close-wide degeneracy is more pronounced when the value of s is close to 1 (the value of $\log_{10} s$ is close to 0).

Regarding the problem of u_0 erroneously converging to 0, we conjecture that this is also a kind of "degeneracy". The lightcurve, which is produced by the primary caustic near the center of mass of the lens, may be morphologically similar to the lightcurve, which is produced by the secondary caustic farther from the center of mass. This issue needs further research.

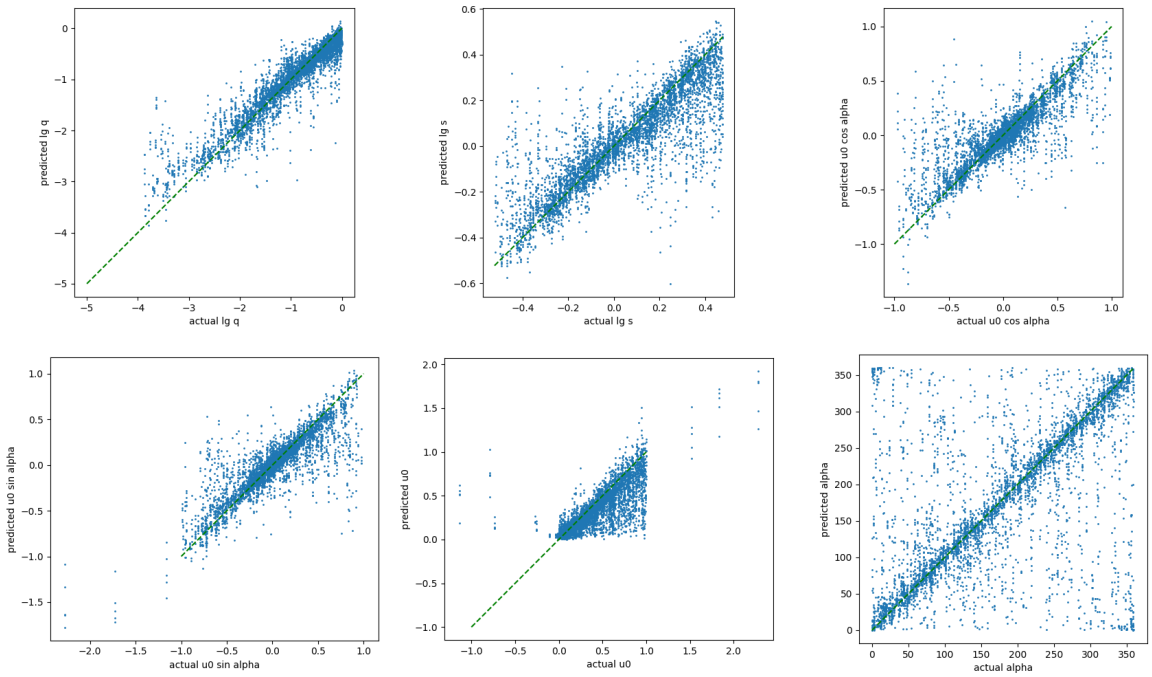


Figure 3: Scatter plot of the predicted-actual values. The parameters u_0 and α are calculated based on the two parameters u_x $(u_0 \cos\alpha)$ and u_y $(u_0 \sin\alpha)$ output directly from the network.

4 Structures of MDN Networks and Corresponding Results

4.1 Methodology of Mixture Density Network

To deal with the problems that traditional neural network with single value output has bad performance on events with microlensing degeneracy, we carry out the mixture density network method [9]. The central idea of the mixture density network is to replace the simple single value output with a probability density distribution in the parameter space of target regression value. We use gaussian mixture model to describe this distribution, for arbitrary probability distributions can theoretically be approximated more accurately by combining enough multiple Gaussian probability distributions. The mixture Gaussian distribution can be written in the following form.

$$\rho_{gm}(y|\omega_i, \sigma_i, \mu_i) = \sum_{i=1}^{N_{gs}} \omega_i \mathcal{N}(y|\sigma_i, \mu_i) \tag{7}$$

N_{gs} is the number of single Gaussian distributions used in the gaussian mixture model, σ_i and μ_i are the standard deviation and mean of single gaussian distributions. ω_i is the weight that each single Gaussian distribution occupies in the gaussian mixture model with the following constraint.

$$\sum_{i=1}^{N} \omega_i = 1 \tag{8}$$

The actual network we built is set to $N_{gaussian} = 12$. The backbone structure of the network is the same as the network mentioned in the previous article section, the difference is that the output is modified. This network outputs the parameters of individual gaussian mixture model:$[z_i, \sigma_i, mu_i]$. Due to the convenience in numerical optimization, ω_i is characterized by another set of parameters z_i.

$$\omega_i = \frac{\exp z_i}{\sum_{i=1}^{N_{gs}} \exp z_i} \tag{9}$$

The loss function being optimized is of the following log-likelihood form.

$$\begin{aligned} &Loss(\{y_j\},\{z_i\}_j,\{\sigma_i\}_j,\{\mu_i\}_j) \\ =&-\log_{10}(\prod_j \rho_{gm}(y_j|\{\omega_i\}_j,\{\sigma_i\}_j,\{\mu_i\}_j)) \\ =&-\sum_{j=1}^{N_{phy}} \log_{10}\{\sum_{i=1}^{N_{gs}} \frac{\exp z_{ij}}{\sum_{i=1}^{N_{gs}} \exp z_{ij}} \mathcal{N}(y_j|\sigma_{ij},\mu_{ij})\} \end{aligned} \tag{10}$$

N_{phy} represents the number of the physical parameters to be regressed. In our network, $N_{phy} = 4$, corresponding to the 4 physical parameters $[\log_{10} q, \log_{10} s, u_0 \cos\alpha, u_0 \sin\alpha]$. These physical parameters are approximated as independent of each other, so the loss function can be written in the form of four independent 1-dimensional mixture probability density logarithms summed together. Strictly speaking, a 4-dimensional mixture density model should be used, but due to computational constraints, a more rigorous model will be tested in the future.

4.2 Training With the Realistic Simulation Dataset

The network is trained with the realistic simulated dataset generated by the method mentioned in section 2.2.2. 200,000 lightcurves are generated as training set and 20,000 lightcurves are generated as validation set. The main structure of the network is constructed according to ResNet34, and the arguments of expected gaussian mixture model, $[z_i, \sigma_i, \mu_i]$, are obtained after the processing of 4 linear layers and 3 individual mapping layers. We focus on the following four physical parameters: $[\log_{10} q, \log_{10} s, u_0 \cos\alpha, u_0 \sin\alpha]$. Each physical parameter corresponds to an independent set of 1-dimensional probability density distribution in the parameter space.

After 500 epochs of training, the log-likelihood loss function of the validation set is reduced to about 0.1. Another sample of 1732 simulated events with smooth distribution of $\log_{10} q$ (as seen in Figure.4) is selected for testing. These 1732 lightcurves are fed into the network to get the probability distributions of physical parameters. We randomly select some examples of the results predicted by the network of these test events and show the results in Appendix.A.

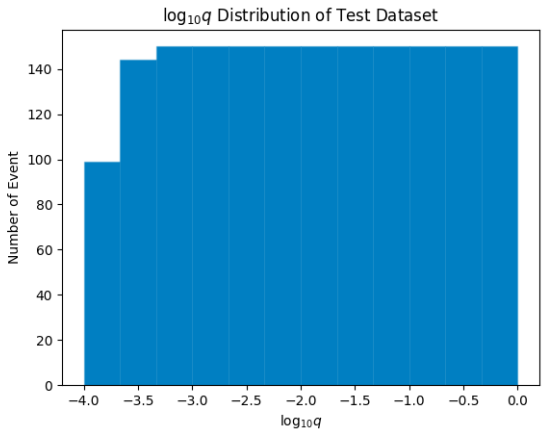


Figure 4: The distribution of $\log_{10} q$ in the test dataset of size 1732

After observing the results of the network output, we can find that the mixture density network is better at handling events with degeneracy, and the network can analyze the corresponding results for the events with degeneracy. In addition to the classical close-wide degeneracy, the network can find other kinds of degeneracy and give the corresponding results.

Neural networks also expose some problems. Due to the randomness within the neural network, it is still possible for the neural network to give completely wrong results at the time of prediction. And if the lightcurve has bad continuity or has some narrow peaks, the neural network is likely to discard this part of the features. In addition, if the signal is too weak or has a low signal-to-noise ratio, or missing data points exist, the neural network is likely not to detect these features. And these cases correspond to events with low mass ratio. It can

be concluded that the effectiveness of neural networks in improving the efficiency of detecting low mass ratio events is very limited and cannot replace human eye discrimination for the time being. This disadvantage can perhaps be solved by obtaining observations with higher quality. The noise of the simulated data can be easily removed, but the network trained with high quality data will not be applied to the current ground-based telescope observations. Due to the influence of the Earth's atmosphere, the current major ground-based microlensing survey projects, such as KMTNet, cannot obtain high quality data and have problems in time continuity. We place our hopes on the next generation of space telescopes, such as the Nancy Grace Roman Space Telescope. We expect that deep learning methods can be better applied on higher quality observational data.

In order to solve the problem of missing data points, we have made some additional attempts, such as two-dimensionalizing the data and adding the recurrent neural network layer before the convolutional neural network part. These attempts are still experimental and we will discuss the results of these approaches in the future.

To investigate the overall distribution of the prediction results, we take the parameter values corresponding to the places where the probability densities take maximum values in the parameter space as approximate regression solutions and record their corresponding probability density values as the basis for the confidence evaluation. A scatter plot of the predicted-actual values (Figure.5) is made for comparison.

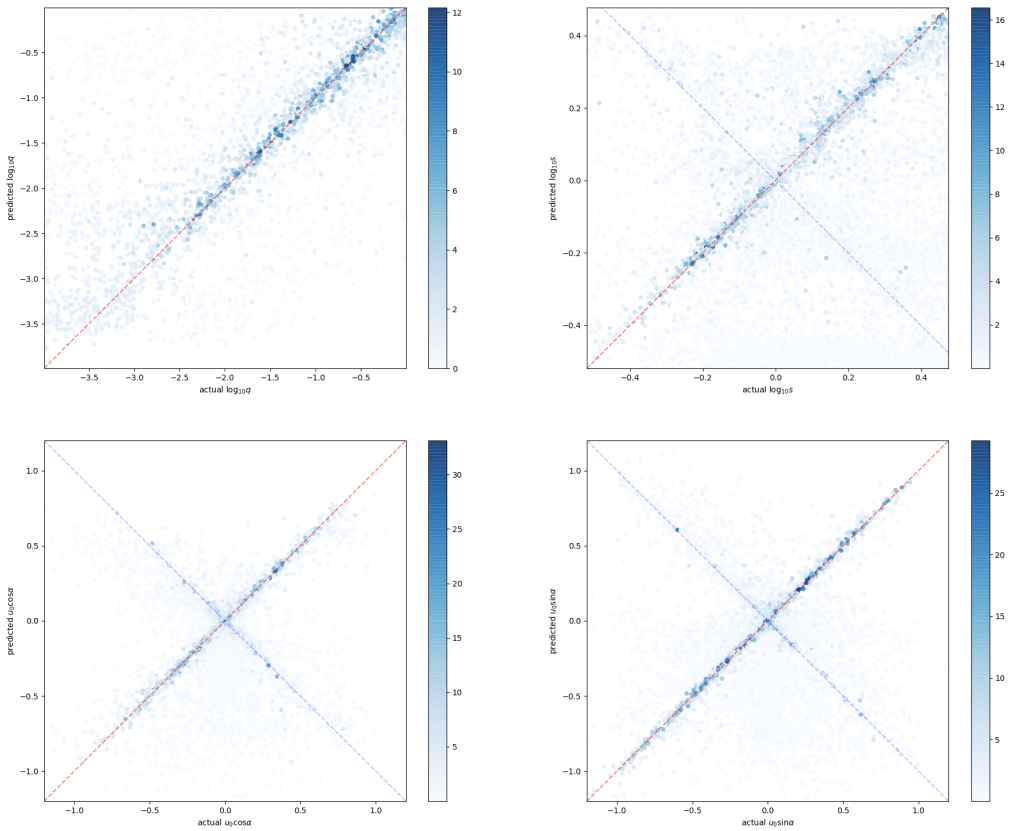


Figure 5: Scatter plot of the predicted-actual values. The parameters u_0 and α are calculated based on the two parameters $u_0 \cos\alpha$ and $u_0 \sin\alpha$ output directly from the network.The probability densities corresponding to the parameters in the parameter space are shown in the graph as color maps. The darker the blue color means the higher the probability density at that data point, and the higher the confidence level of that data point.

From the scatter plot, it can be seen that the prediction of $\log_{10} q$ is better when $\log_{10} q$ is larger (about ≥ -2.5). Predictions with higher confidence are closer to the center of the line representing the equality of the predicted and actual values (in the later section we will call this line predicted-actual-equal line). In the region where $\log_{10} q$ is smaller, it can be seen that the data points deviate from the predicted-actual line to a

greater extent and with less confidence. This indicates that our network is poor at detecting events with small mass ratio and weak binary lensing signal.

It can be seen that the overall predicted results for $\log_{10} s$ are relatively poor. The overall deviation of the data points from the predicted-actual line is large. The blue straight line perpendicular to the red predicted-actual line in the figure corresponds to a negative sign between the predicted and true values, which corresponds to close-wide degeneracy(in the later section we will call this line predicted-actual-opposite line). It can be seen that there are some points of higher confidence close to the predicted-actual-opposite line, but the number is not large enough to observe the expected more pronounced close-wide degeneracy feature from the scatter plot of $\log_{10} s$. Further examination of the probability of occurrence of close-wide degeneracy events for the current simulation data generation method is necessary.

The predictions for $u_0 \cos\alpha$ and $u_0 \sin\alpha$ are relatively good. It can be observed that more data points with higher confidence appear near the predicted-actual-opposite line. This may correspond to some kind of geometric degeneracy, stemming from the symmetry of the binary lensing system about the principal axis of symmetry. The occurrence of this kind of degeneracy in the $u_0 \cos\alpha$ and $u_0 \sin\alpha$ scatterplots helps to discuss and further explore the temporal inversion symmetry of the neural network with respect to the input time series.

4.3 Test on Real Events from KMTNet: KMT-2019-BLG-0371

We have selected some known binary lensing events to examine the functionality of the network. This event is confirmed as a microlensing system, and is explained by a model with a mass ratio between the host star and planet of $q \sim 0.08$ [7]. The regression of the traditional best fitting method indicates that this event has degeneracy, so we use this event to evaluate both regression accuracy and degeneracy determination ability. The results of the network output are shown in Figure.6 and Figure.7.

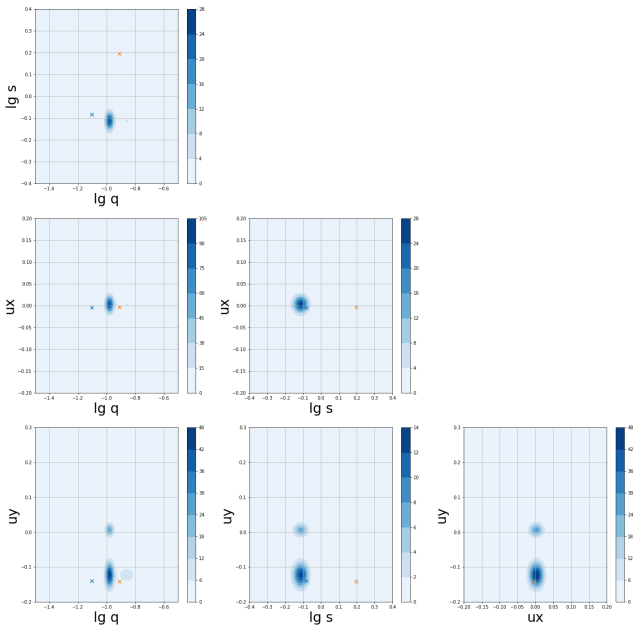


Figure 6: The probability density in the parameter space of network predictions. The two points drawn in the figure correspond to the two solutions given by best fitting. It can be seen that the prediction results show a more serious deviation when the network is applied to real data. For $\log_{10} q$, the network gives only order-of-magnitude estimates. For $\log_{10} s$, the network fails to give a degeneracy prediction.

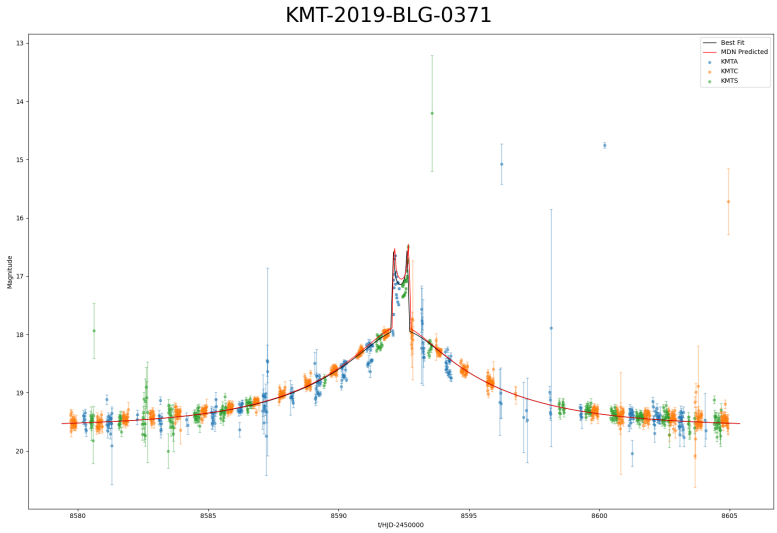


Figure 7: It can be seen that although the parameter prediction of the network is not accurate, the lightcurve reduced from the predicted parameters is more consistent with the original lightcurve morphological characteristics. It is conjectured that the network is more sensitive to the morphology of the lightcurve compared to the exact values of the physical parameters.

It can be seen that the accuracy of the parameters predicted by the network when applied to real data is not satisfactory. This is related to the low signal-to-noise ratio of the real data and the missing data points. This reminds us of the need to complicate the noise model of the simulated data and to find new isochronization methods to make the simulated data match the real data as much as possible. In addition, we can find that although the parameters regressed by the network are not accurate, the reduced lightcurve matches more consistently with the original lightcurve. This reminds us that neural networks may be more sensitive to the intuitive shape of the input data than the exact physical parameters. This inspires us to think about the possibility of applying unsupervised learning to this problem in the future. By unsupervised learning methods such as Auto encoder&decoder, the output lightcurve of the network is made to match the original lightcurve as closely as possible, and then the machine features are extracted from the intermediate layer for further analysis to obtain the physical parameters. Perhaps the introduction of unsupervised learning methods can improve the accuracy of parameter prediction, which is one of the directions we explore in the future.

5 Conclusion

The biggest advantage of deep learning algorithms such as CNN over traditional methods is speed. For a light-variable curve, a neural network can complete modeling in a few milliseconds, giving predicted values of the parameters of microlensing system. But the accuracy of neural network prediction is not at all comparable to traditional methods such as MCMC. Neural network algorithms are usually used for classification problems, and there are always large errors in the parameters when dealing with such regression problems. During the training process, the network randomly discards some of the parameters, for example, the ReLU activation function removes all the information of negative elements, as well as the dropout layer set to avoid overfitting discards some information randomly. It may cause some subtle features of lightcurve to be discarded, so that the parameters cannot be regressed exactly. In addition, our CNN neural network, which only outputs single values, cannot handle complex degeneracy problems. MDN solves the problem of degeneracy to some extent. However, the MDN we currently use assumes that the probability distribution of each physical parameter in the parameter space is independent, which may pose some problems. In the future, we will further modify the form of MDN to replace the current discrete probability density model with a mixed density model in a high-dimensional space and test the effect. In addition MDN still makes frequent errors when regressing parameters, which may require us to further search for ways to improve the network structure and find better data pre-processing methods and higher quality data.

At present, the significance of using neural networks for parameter estimation is that neural network algorithms, such as CNN, can quickly give the range where the physical parameters are located within certain

error limits, thus reducing the range of parameters that need to be searched by traditional methods such as MCMC, and thus reducing the computational cost of traditional methods.

Future applications of deep learning algorithms for modeling microlensing system can be improved in the following ways:

- Improvements for CNNs that only output single values: Instead of outputting individual parameter values, algorithms such as Bayesian neural network can be used to output probability distributions in the parameter space. These method provide the possibility of outputting multiple possible solutions, then partially lifting the degeneracy.
- Speeding up the step of generating lightcurve from the given parameters: Traditional methods, such as MCMC, consume a lot of computation time in the step of getting the light curve given the parameters. It is now common practice to calculate the magnification at each moment in the lightcurve by solving polynomial equations. This is extremely time consuming when there are many data points in the lightcurves. This step can probably be replaced by deep learning. Using algorithms such as auto-encoder&decoder or GAN, it may be possible to generate lightcurve from given parameters quickly. We are currently looking for information to investigate the possibility of this idea.

In the future we will test the efficiency of the more advanced deep learning methods, such as Bayesian neural network and auto encoder&decoder. The precision of parameter estimation using neural networks has proven difficult to catch up with traditional methods, so it may make more sense to turn attention to the speed advantage of neural networks.

References

[1] R. Poleski and J. Yee, "MulensModel: Microlensing light curves modeling," Mar. 2018.

[2] C. B. Henderson, B. S. Gaudi, C. Han, J. Skowron, M. T. Penny, D. Nataf, and A. P. Gould, "OPTIMAL SURVEY STRATEGIES AND PREDICTED PLANET YIELDS FOR THE KOREAN MICROLENSING TELESCOPE NETWORK," *The Astrophysical Journal*, vol. 794, p. 52, sep 2014.

[3] M. T. Penny, "Speeding up Low-mass Planetary Microlensing Simulations and Modeling: The Caustic Region Of INfluence," *apj*, vol. 790, p. 142, Aug. 2014.

[4] W. Zhu, M. Penny, S. Mao, A. Gould, and R. Gendron, "Predictions for Microlensing Planetary Events from Core Accretion Theory," *apj*, vol. 788, p. 73, June 2014.

[5] J. C. Yee, Y. Shvartzvald, A. Gal-Yam, I. A. Bond, A. Udalski, S. Kozłowski, C. Han, A. Gould, J. Skowron, D. Suzuki, F. Abe, D. P. Bennett, C. S. Botzler, P. Chote, M. Freeman, A. Fukui, K. Furusawa, Y. Itow, S. Kobara, C. H. Ling, K. Masuda, Y. Matsubara, N. Miyake, Y. Muraki, K. Ohmori, K. Ohnishi, N. J. Rattenbury, T. Saito, D. J. Sullivan, T. Sumi, K. Suzuki, W. L. Sweatman, S. Takino, P. J. Tristram, K. Wada, MOA Collaboration, M. K. Szymański, M. Kubiak, G. Pietrzyński, I. Soszyński, R. Poleski, K. Ulaczyk, Ł. Wyrzykowski, P. Pietrukowicz, OGLE Collaboration, W. Allen, L. A. Almeida, V. Batista, M. Bos, G. Christie, D. L. DePoy, S. Dong, J. Drummond, I. Finkelman, B. S. Gaudi, E. Gorbikov, C. Henderson, D. Higgins, F. Jablonski, S. Kaspi, I. Manulis, D. Maoz, J. McCormick, D. McGregor, L. A. G. Monard, D. Moorhouse, J. A. Muñoz, T. Natusch, H. Ngan, E. Ofek, R. W. Pogge, R. Santallo, T. G. Tan, G. Thornley, I. G. Shin, J. Y. Choi, S. Y. Park, C. U. Lee, J. R. Koo, and μFUN Collaboration, "MOA-2011-BLG-293Lb: A Test of Pure Survey Microlensing Planet Detections," *apj*, vol. 755, p. 102, Aug. 2012.

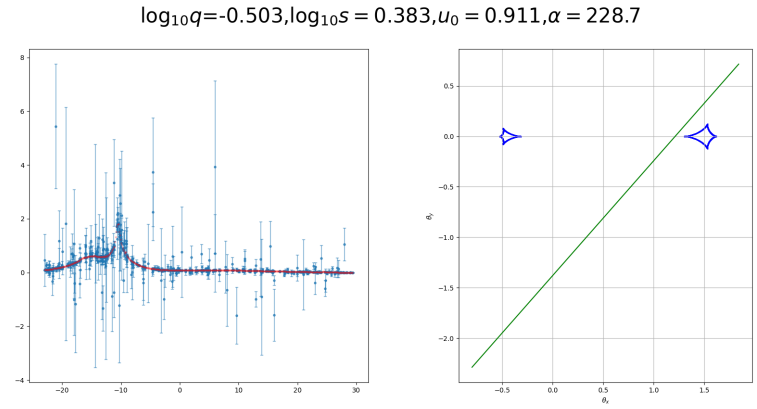
[6] C. B. Henderson, B. S. Gaudi, C. Han, J. Skowron, M. T. Penny, D. Nataf, and A. P. Gould, "Optimal Survey Strategies and Predicted Planet Yields for the Korean Microlensing Telescope Network," *apj*, vol. 794, p. 52, Oct. 2014.

[7] Y. H. Kim, S.-J. Chung, J. C. Yee, A. Udalski, I. A. Bond, Y. K. Jung, A. Gould, M. D. Albrow, C. Han, K.-H. Hwang, and et al., "Kmt-2019-blg-0371 and the limits of bayesian analysis," *The Astronomical Journal*, vol. 162, p. 17, Jun 2021.

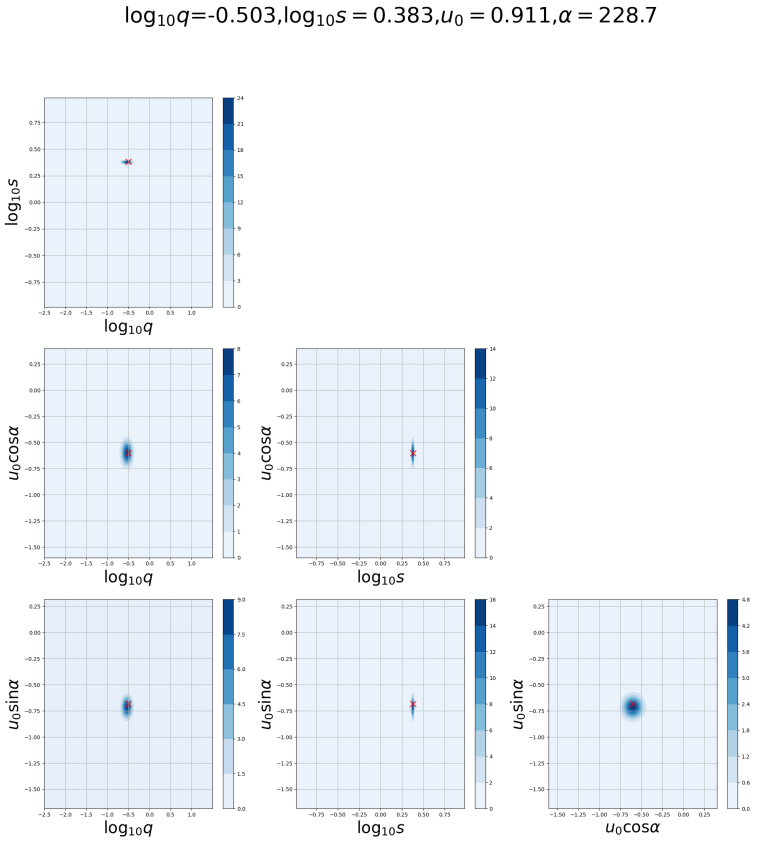
[8] K. Griest and N. Safizadeh, "The use of high-magnification microlensing events in discovering extrasolar planets," *The Astrophysical Journal*, vol. 500, p. 37–50, Jun 1998.

[9] C. Bishop, “Mixture density networks,” workingpaper, Aston University, 1994.

A Appendix: Examples of MDN output

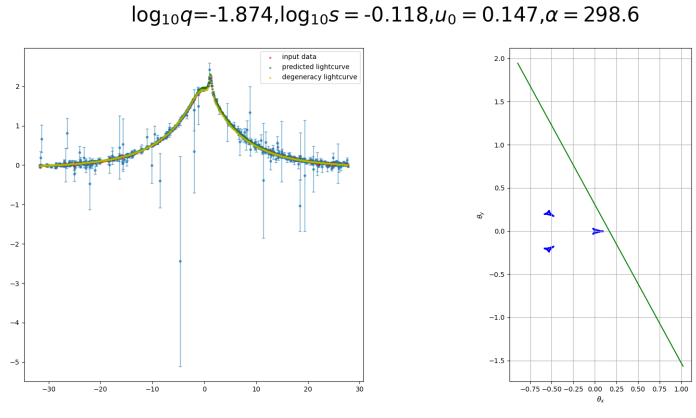


(a) Lightcurve of event

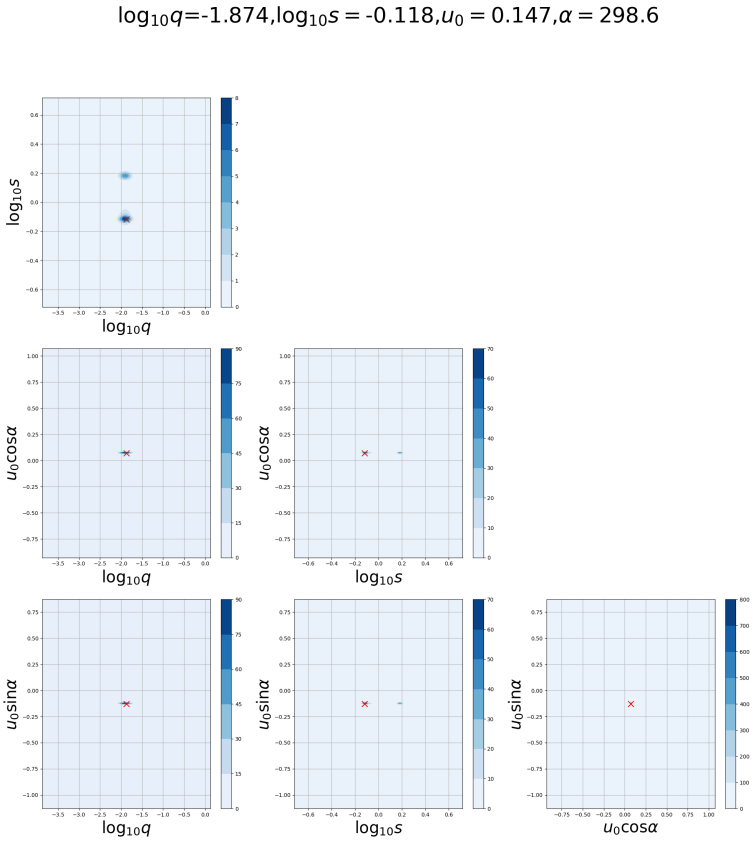


(b) Probability density in parameter space

Figure 8: A classic simulation event that does not have degeneracy. The red data points are the lightcurve input to the network after data preprocessing. figure (b) shows the probability density distribution plotted by pairing the parameters two by two.

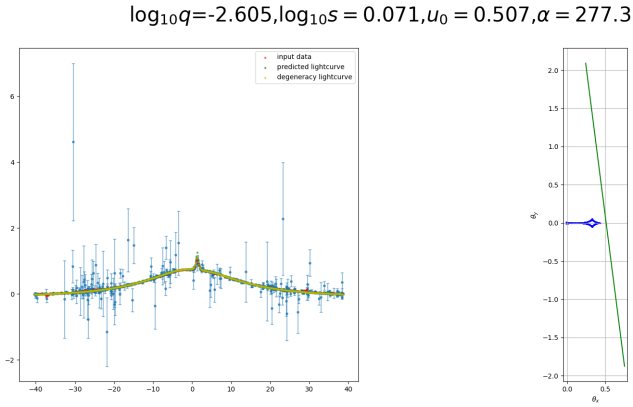


(a) Lightcurve of event

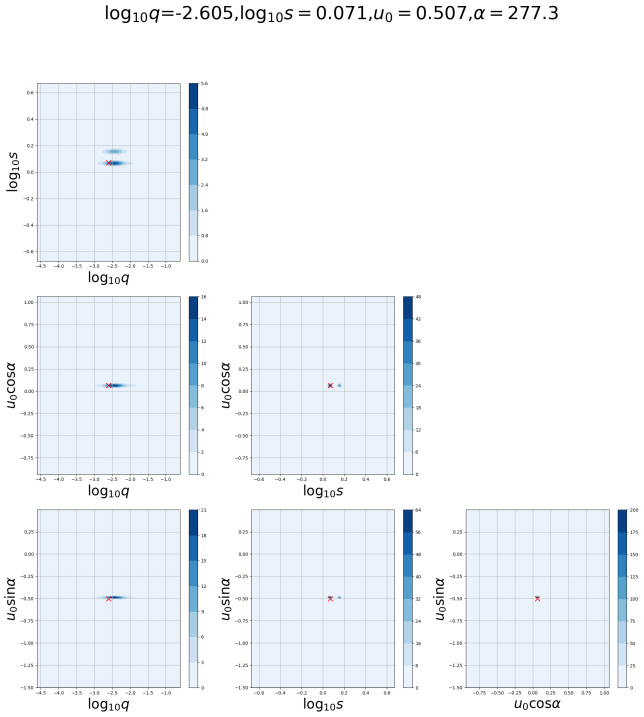


(b) Probability density in parameter space

Figure 9: An event with the classic close-wide degeneracy. We used the maximum values of the probability density distribution function as parameters for the re-modeling verification. Since $\log_{10} s$ appears degeneracy, there are two lightcurves (green predicted lightcurve and yellow degeneracy lightcurve) that are re-modeled according to the parameters. It can be seen that the two lightcurves obtained by re-modeling according to the regression parameters overlap well with the known lightcurve.

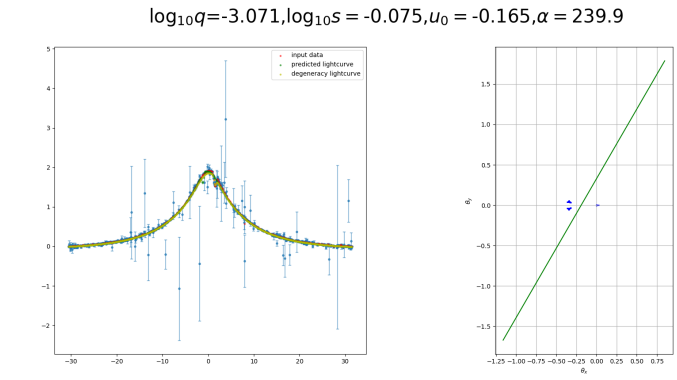


(a) Lightcurve of event

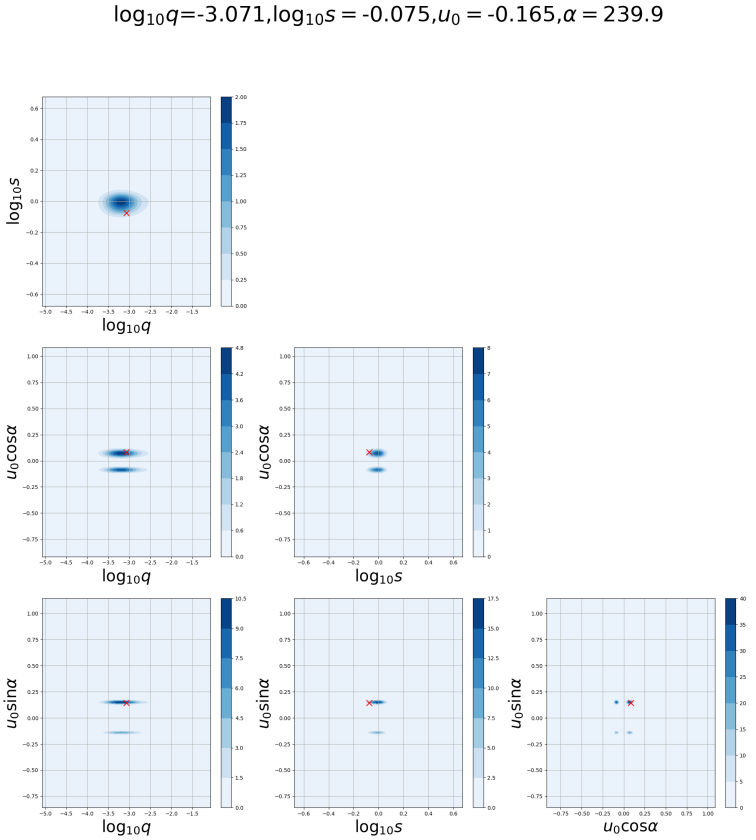


(b) Probability density in parameter space

Figure 10: Another event with $\log_{10} s$ degeneracy. This degeneracy does not seem to be close-wide degeneracy. it can be seen that the predicted lightcurve and the degeneracy lightcurve fit the original curve more closely. This event provides evidence that the neural network has the ability to find other possible degeneracy

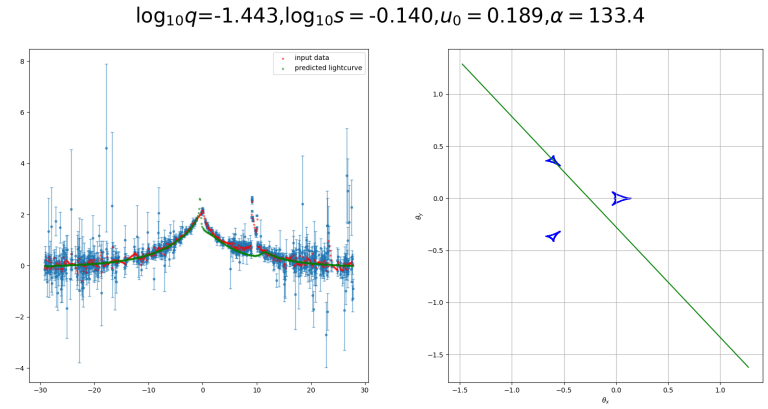


(a) Lightcurve of event

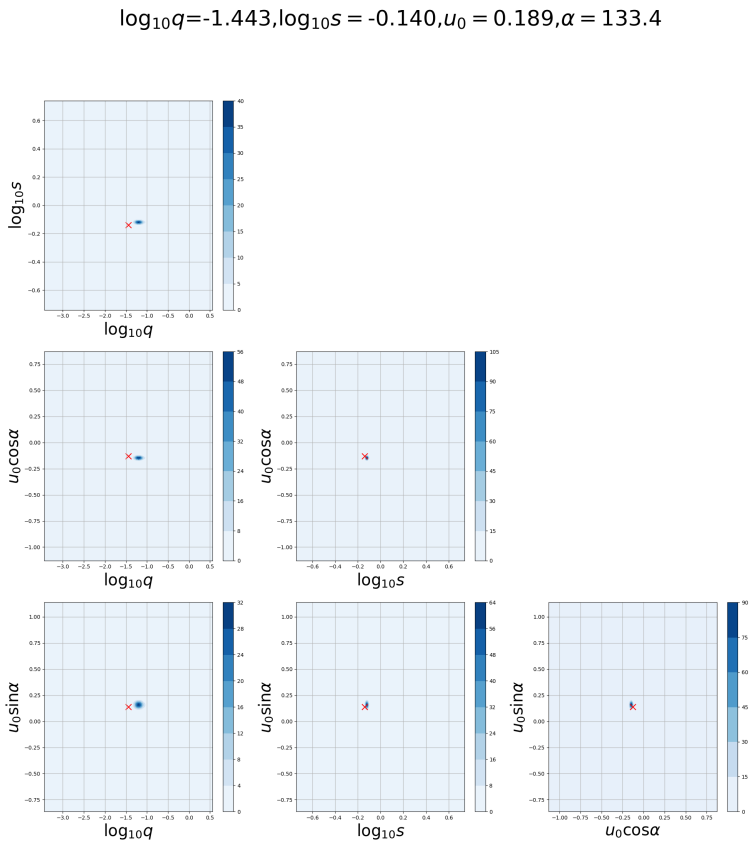


(b) Probability density in parameter space

Figure 11: An event with $u_0 \cos\alpha$ degeneracy. This degeneracy corresponds to the symmetry of the microlensing system to the principal axis. The $u_0 \cos\alpha$ inverse sign corresponds to the rotation of the background source's trajectory around the origin to a position that is mirror-symmetric with respect to the principal axis. The transformed lightcurve is theoretically the time inverse of the original lightcurve. The $u_0 \cos\alpha$ inverse sign corresponds to the rotation of the background source's trajectory around the origin to a position that is mirror-symmetric with respect to the principal axis. The transformed lightcurve is theoretically the time inversion of the original lightcurve. It can be seen in Figure.(a) that the two reconstructed lightcurves are morphologically identical, differing by one time inversion. This indicates that the neural network is inclusive of data with symmetric time inversion. In addition, the degeneracy of $u_0 \sin\alpha$ in this example is verified as an incorrect prediction of the neural network.



(a) Lightcurve of event



(b) Probability density in parameter space

Figure 12: An example of a neural network prediction gone wrong. The neural network completely ignores the steeper and narrower peak at $t \sim 10d$.