Application of Machine Learning to Detecting Gravitational-Wave Signals

Kyungmin Kim
The Chinese University of Hong Kong

December 7, 2018 @ KASI, Korea

Contents

- What is Machine Learning?
- Applications
  - New ranking method with machine learning for low-latency detection of gravitational waves from compact binary mergers
  - Identification of lensed gravitational waves using machine learning

What is Machine Learning?

- Introduction to Machine Learning for Newbies - Part I
    - Conceptual Definitions, Basic Principles, Kinds, and Jargons
- Introduction to Machine Learning for Newbies - Part II
    - Practical considerations for Input Data Preparation

New Ranking Method with Machine Learning for Low-Latency Detection of Gravitational-Waves from Compact Binary Mergers

[LIGO-P1800253]

Kyungmin Kim,[1] Tjonnie G. F. Li,[1] Rico K. L. Lo,[1] Surabhi Sachdev,[2] Robin S. H. Yuen[1]

[1] The Chinese University of Hong Kong
[2] California Institute of Technology

Overview

- The nature, multivariate structure, of candidate events of searches for gravitational waves (GWs) from compact binary mergers seamlessly leads to consider machine learning (ML).
- Aims at enhancing the performance of GstLAL-based inspiral pipeline with supervised ML.

- Investigates feasibility of application of ML with the MDC data for GW150914.
- Implements supervised learning with Random Forest and Neural Network.
- Analyzes the classification performance through three figure-of-merits; confusion matrix, 2-D histogram, and receiver operation characteristic curve.
- Computes search sensitivity in terms of range against FAR.
- Compares classification performance and search sensitivity to those obtained with log-likelihood ratio, the conventional ranking statistic of GstLAL inspiral pipeline.

Tools

- **GstLAL inspiral pipeline**
  - Generates MDC triggers and time-slide background data.

- **Machine learning:** *2-class supervised learning*
  - **Random Forest with scikit-learn**
    - Hyperparameter optimization:"GridSearchCV" (module of scikit-learn)
  - **Neural Network with TensorFlow**
    - 4 hidden layers (32 nodes per layer) with L2 regularization (for connection weight), dropout (to remove node dependency), ReLU (as activation function between nodes), Cross Entropy (as cost function)
    - No ad-hoc method for searching optimal hyperparameters

Data Preparation

- Signal samples: mock data of GW150914 from MDC of GstLAL inspiral pipeline (~ 5 000 samples)
- Background samples: time-slide data around the GPS times of injections of the MDC (~ 172 000 samples)
- Features: mass1, mass2, spin1z, spin2z, snr, and chisq (6 features)
- Train/Test data: 75%/25% of shuffled samples (no validation data)

TABLE I. Number of signal and background samples for training and test data.

		Signal	Background
H1	Train	3,641	129,405
	Test	1,220	43,129
L1	Train	3,623	129,423
	Test	1,238	43,111

Training

- Time for training (w/ ~ 122 000 samples of 6 features) on MacBook Pro
  - Random Forest: **~ 6—7 hrs** for running GridSearchCV with 288 combinations
  - Neural Network: **~ 7—10 mins**

TABLE II. Tested entries for hyperparameters of RF in running `GridSearchCV`.

Hyperparameter	Entry
`n_estimators`	50, 100, 200
`criterion`	gini, entropy
`max_features`	2, 4, 6
`min_samples_split`	2, 3, 4, 5
`max_depth`	None, 10, 30, 50

TABLE III. Determined optimal hyperparameters of RF by running `GridSearchCV` for each data. One can see that some hyperparameters are the same for different data.

Hyperparameter	Data	Optimal
`n_estimators`	BBH (H1)	50
	BBH (L1)	50
`criterion`	BBH (H1)	entropy
	BBH (L1)	entropy
`max_features`	BBH (H1)	4
	BBH (L1)	4
`min_samples_split`	BBH (H1)	3
	BBH (L1)	4
`max_depth`	BBH (H1)	30
	BBH (L1)	10

TABLE IV. Hyperparameters for NN.

Hyperparameter	Value
layers (nodes)	1 input (6 nodes)
	4 hidden (32 nodes for each)
	1 output (1 node)
learning rate	0.01 %
regularization	$L2$ with 0.01 %
dropout	10 %
activation function	ReLU
cost function	Cross entropy with Softmax
batch size	1024

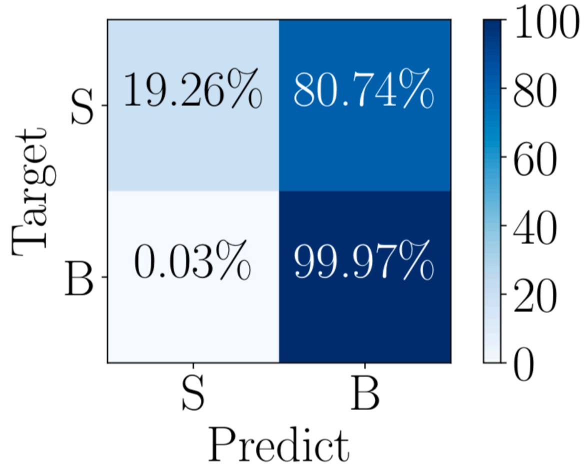
Evaluation

- Time for evaluation (w/ ~ 45 000 samples of 6 features): **~ O(100) ms**
- Output: probabilistic prediction between 0 and 1 → **rank**
- For the performance test of the evaluation result, 3 figure-of-merits were used:
  - Confusion matrix,
  - 2-D histogram: ln L vs. rank of ML,
  - Receiver Operation Characteristic (ROC) curve.

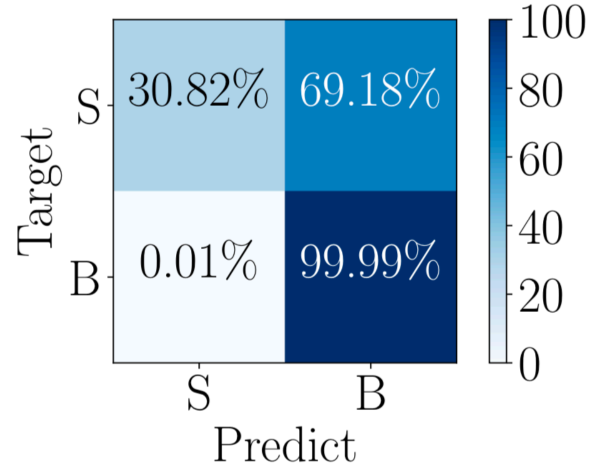
Confusion Matrix

- Cut-threshold
  - $\ln L$: 3σ of all $\ln L$ values in test data
  - rank: 0.5 (c.f. 3σ of rank: 0.26 for H1, 0.28 for L1)

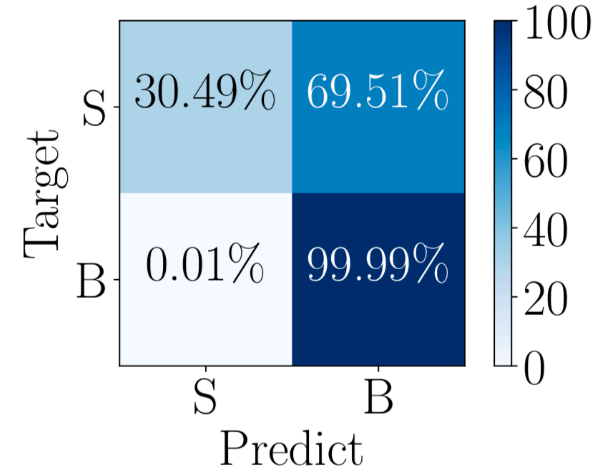
True Pos	False Neg
False Pos	True Neg



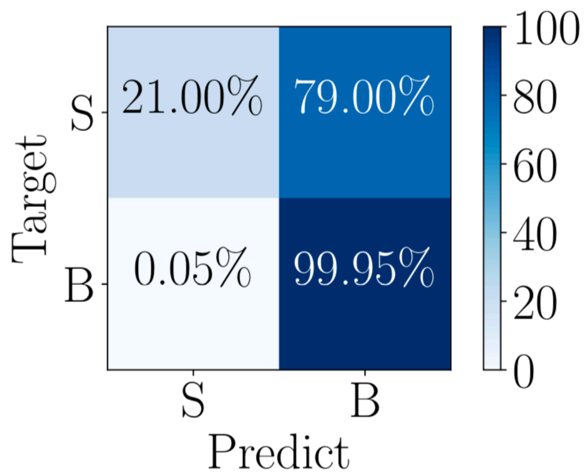
(a) $\ln L$ on H1 data



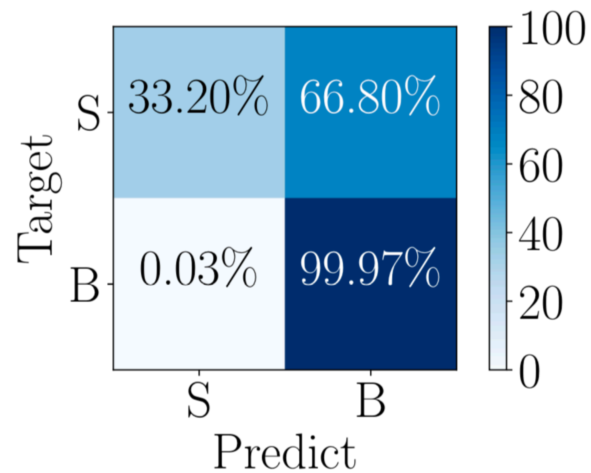
(b) RF on H1 data



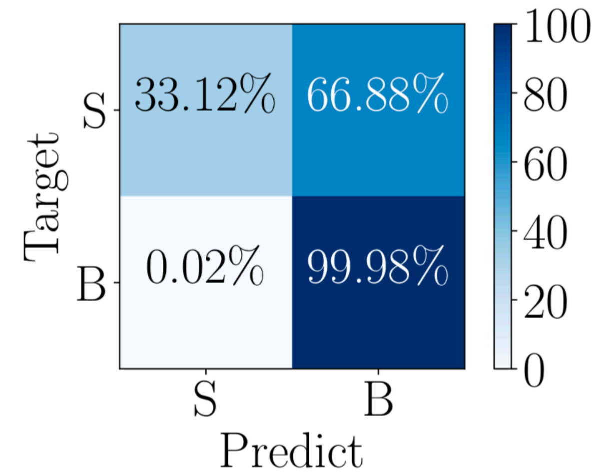
(c) NN on H1 data



(d) $\ln L$ on L1 data

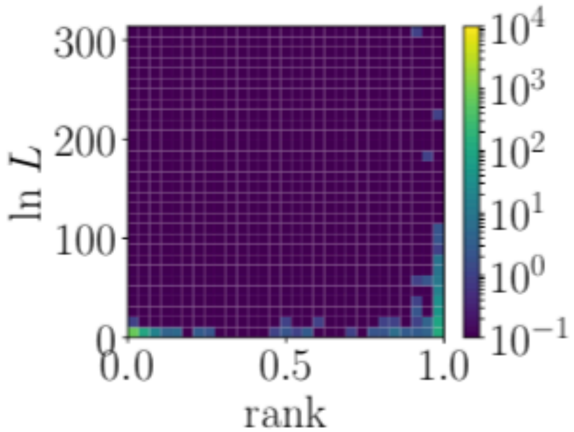


(e) RF on L1 data

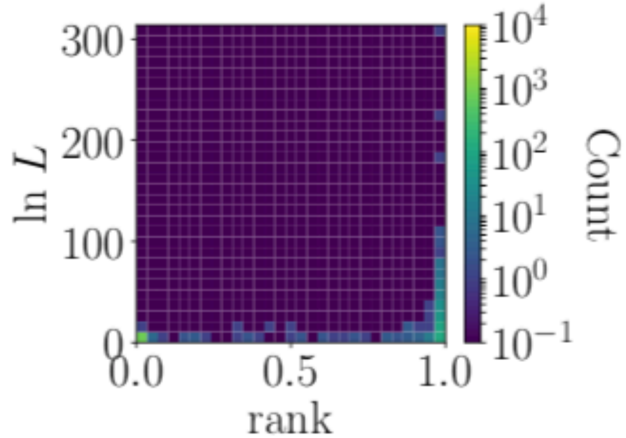


(f) NN on L1 data

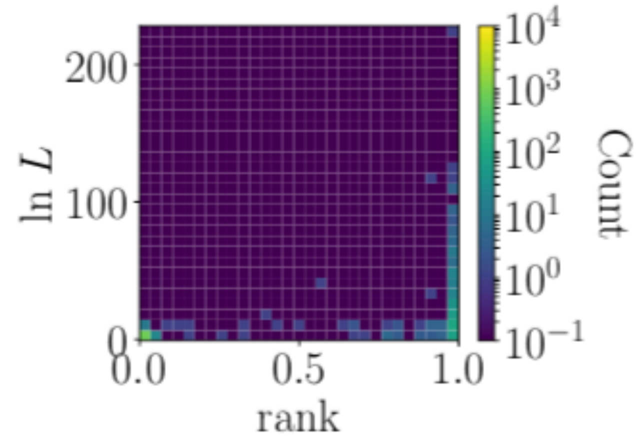
2-D Histogram



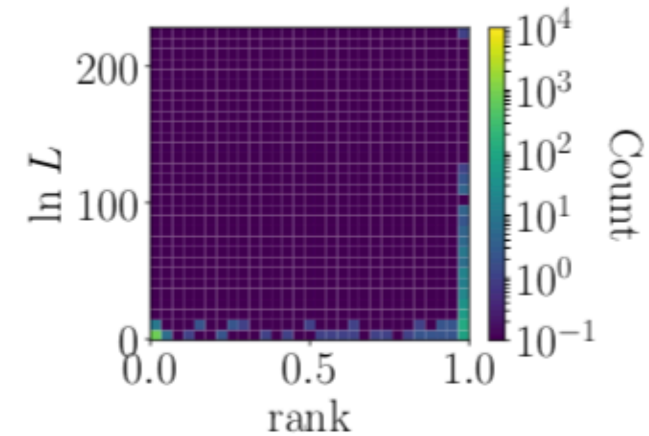
(a) RF on H1 data - Signal



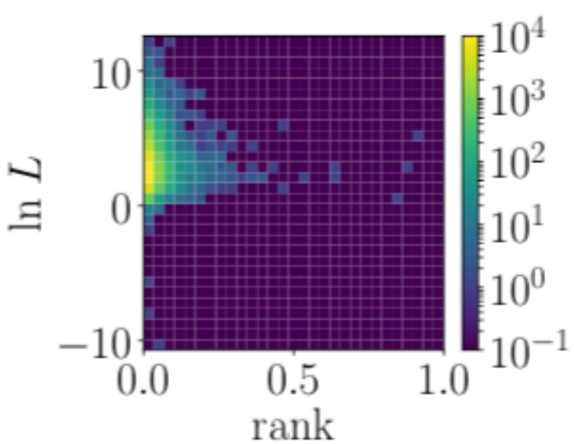
(b) NN on H1 data - Signal



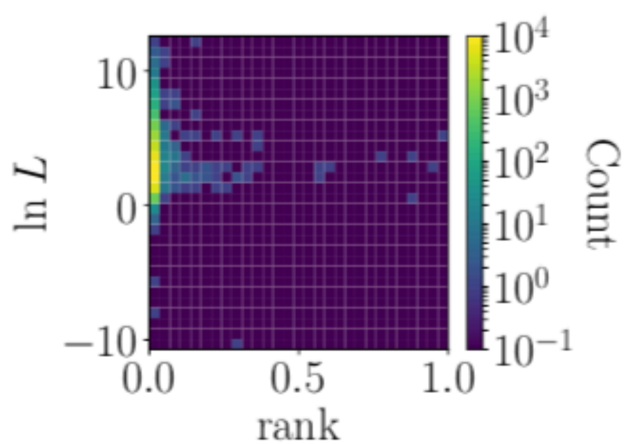
(e) RF on L1 data - Signal



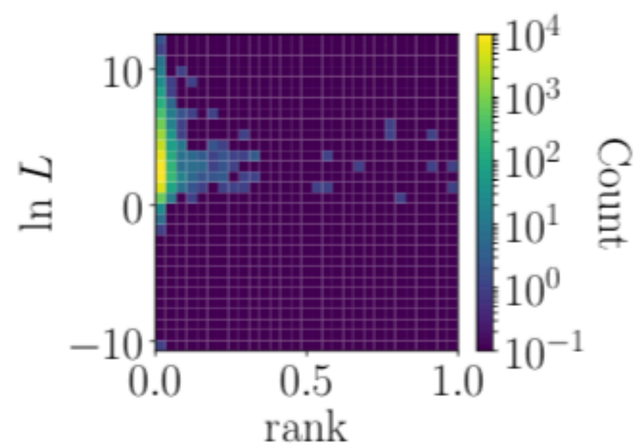
(f) NN on L1 data - Signal



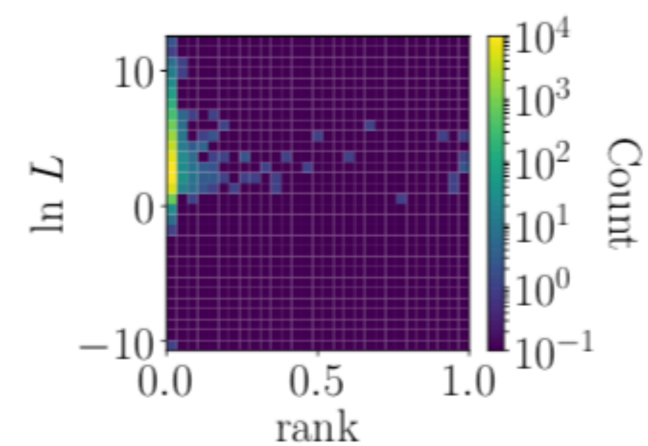
(c) RF on H1 data - Background



(d) NN on H1 data - Background



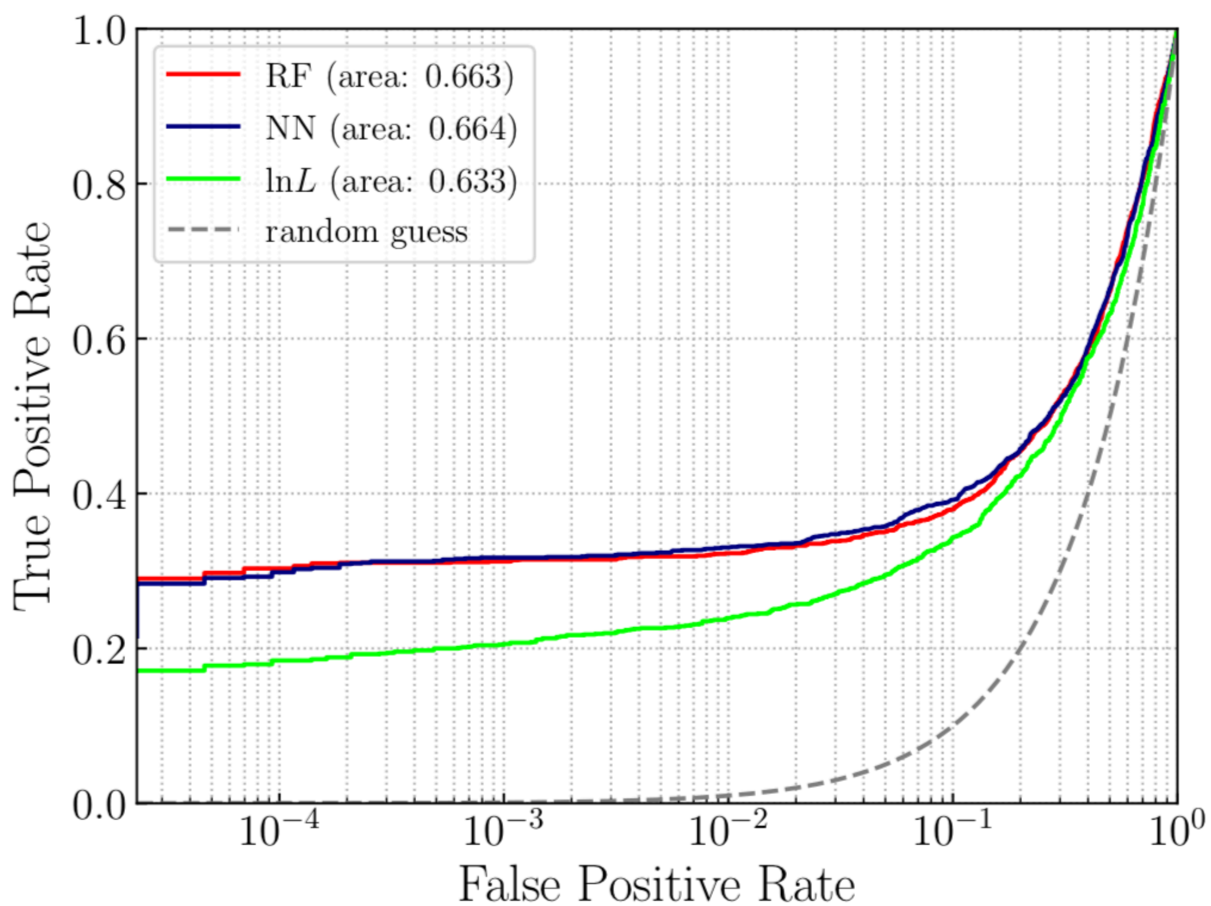
(g) RF on L1 data - Background



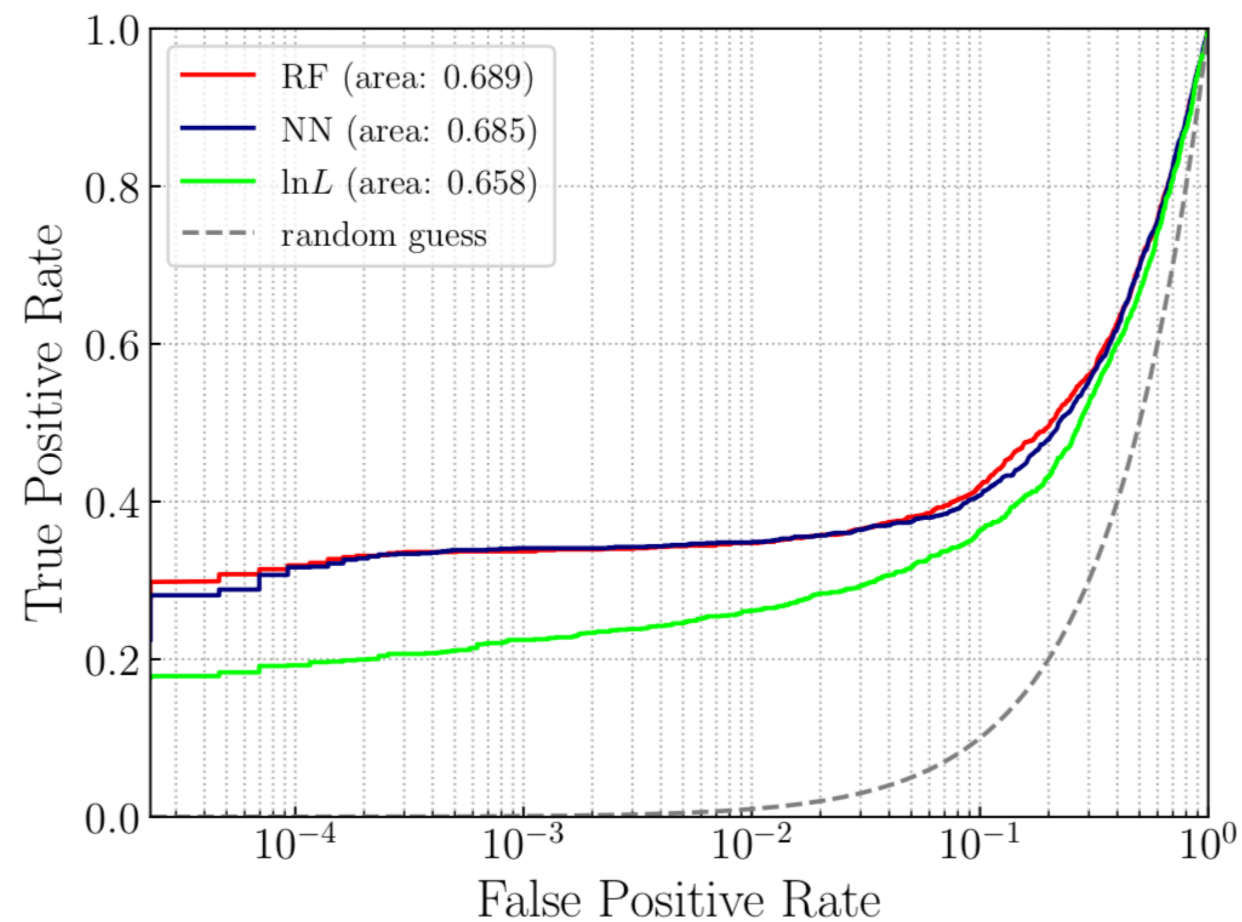
(h) NN on L1 data - Background

ROC Curve

- True Positive Rate := (# of signal samples($\rho \geq \rho$th)) / (total # of sig samples)
- False Positive Rate := (# of bg samples($\rho \geq \rho$th)) / (total # of bg samples)
- Area under Curve: probability that a ranking method will score higher rank on an arbitrary signal instance than the rank of an arbitrary background instance



(a) H1 data



(b) L1 data

Search Sensitivity

- **Computing FAR**
  - FAP = FPR
    - FAP := # of background samples(ln L > ln L_{th}) / total # of background samples
    - FPR := # of background samples($r \geq r_{\mathrm{th}}$) / total # of background samples (where r = either ln L or rank)
    - complementary cumulative distribution function (ccdf)
    - build interpolator with scipy.interpolate.interp1d(r, ccdf)
  - FAR := FAP / length of segment

Search Sensitivity

- **Computing Efficiency**
  - # of found sample: FAR < $\mathrm{FAR}_{\mathrm{fiducial}}$
  - Wilson binomial confident w/ continuity correction
    - center = $(p + z^2/2y) / (1+z^2/y)$,
      - y: number of total signal samples
      - p: fraction of found samples
      - z: probit via scipy.stats.norm.isf (inverse survival function)
    - upper/lower bounds

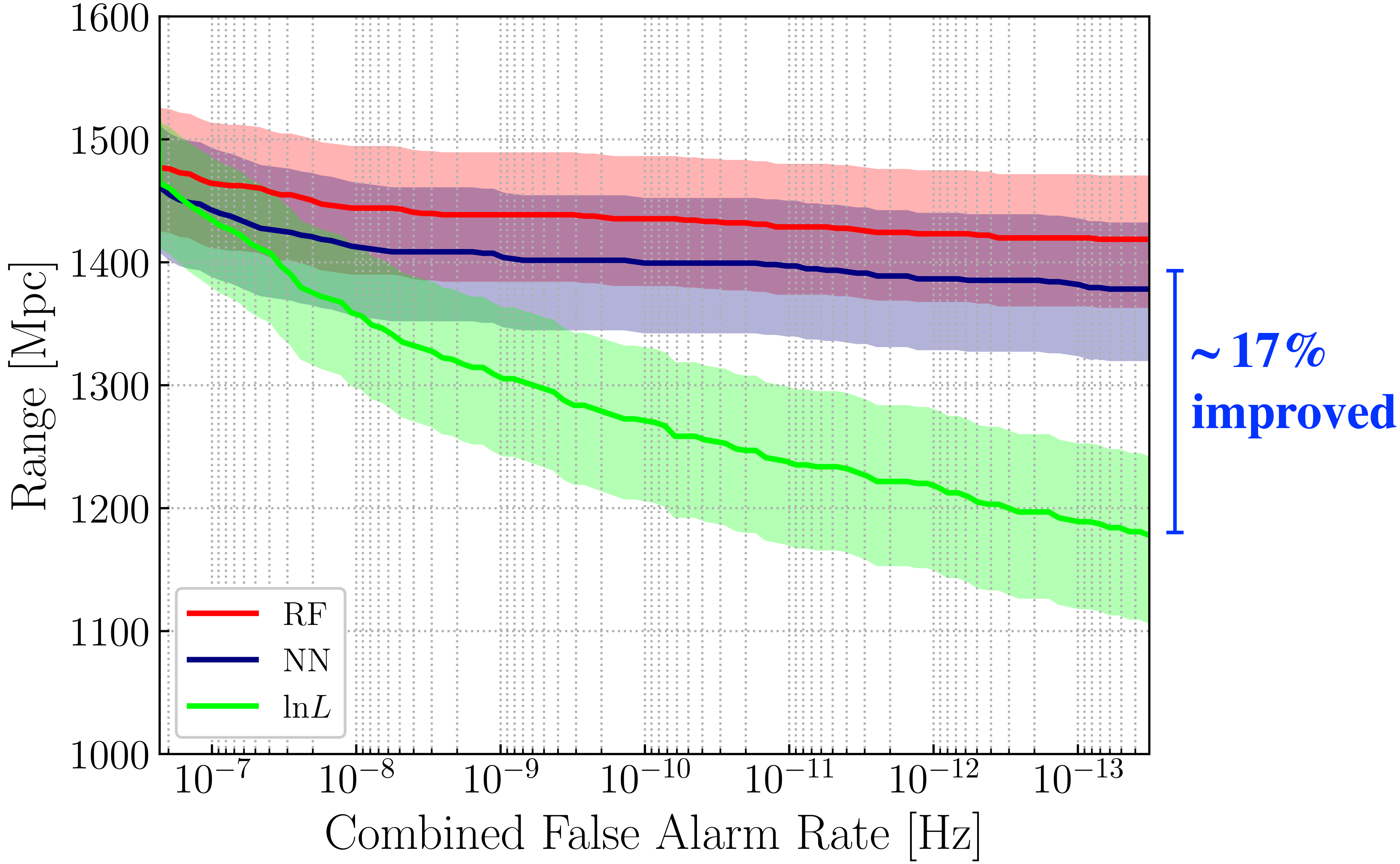
$$\omega^- = \max\left\{0, \frac{2yp + z^2 - \left[z\sqrt{z^2 - 1/y + 4yp(1-p) + (4p-2)} + 1\right]}{2(y+z^2)}\right\}$$

$$\omega^+ = \min\left\{1, \frac{2yp + z^2 + \left[z\sqrt{z^2 - 1/y + 4yp(1-p) + (4p-2)} + 1\right]}{2(y+z^2)}\right\}$$

Search Sensitivity

- **Computing Range**
  - compute search volume w/ efficiency and injection distance via trapezoidal formula
  - volume *= livetime → *VT*
  - range := (*VT* / (4π*livetime/3))**(1/3)

Search Sensitivity



Remarks

- Concludes that
  - ML can be an alternative ranking method to ln L.
  - it is worth to consider ML as a new ranking method for future low-latency searches.
- Directions of future work
  - including DQ information into the input data for training / evaluation
  - building up the strategy / framework for online search

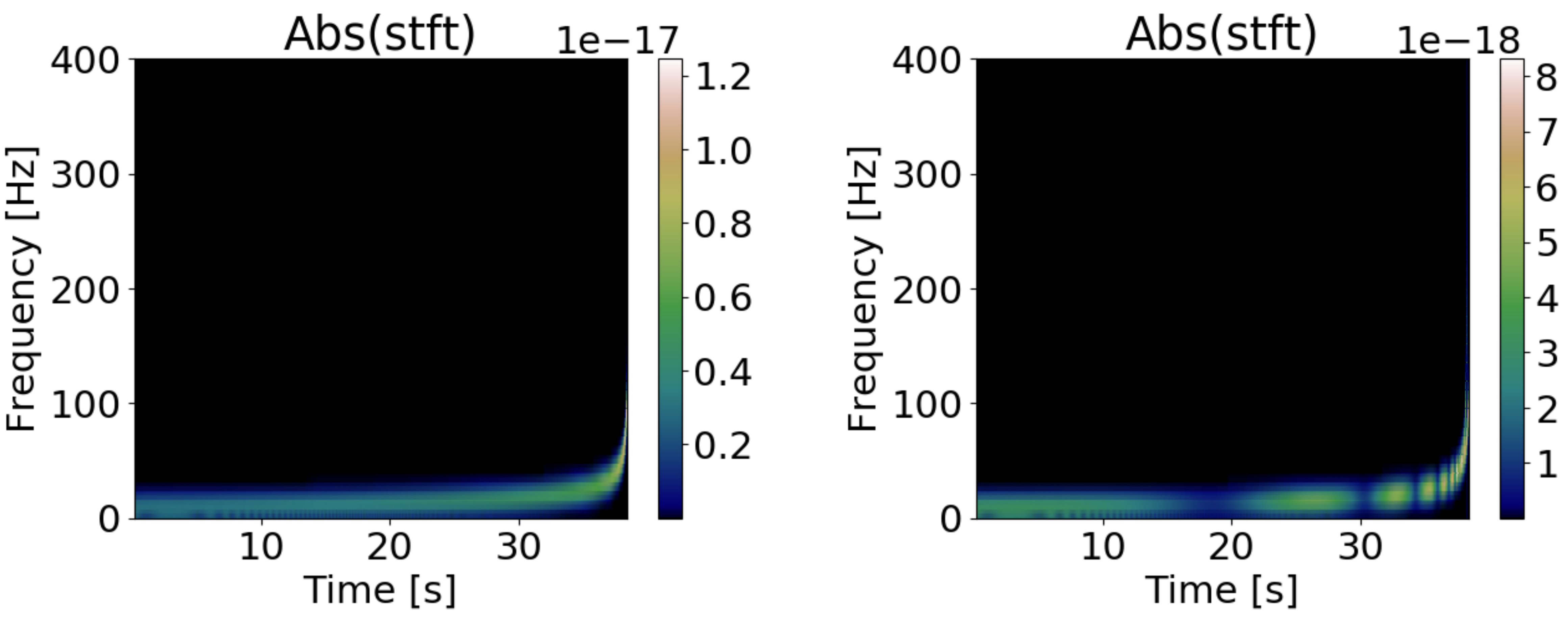
Identification of Lensed Gravitational-Waves

Kyungmin Kim,[1] Ivan S. C. Li,[2] Tjonnie G. F. Li,[1]
Amit Jit Singh,[1] Robin S. H. Yuen[1]

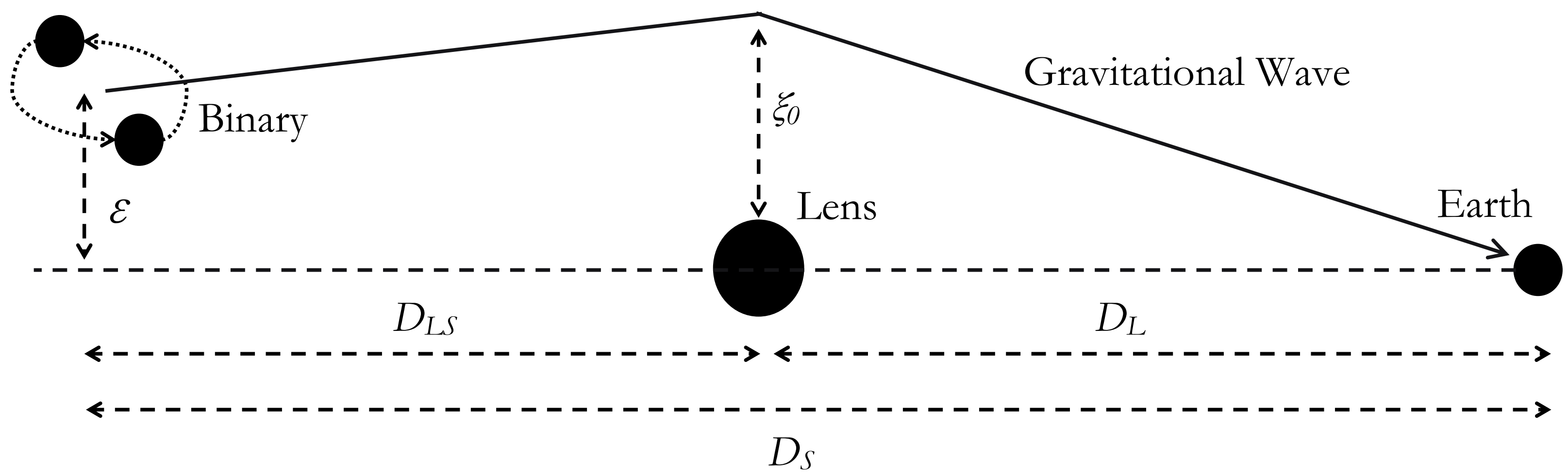
[1] The Chinese University of Hong Kong
[2] Imperial College London

Overview

- Can expect "gravitational lensing" effect on gravitational waves (GWs) like gravitational lensing on astronomical observations.
- May identify lensed GW from spectrogram, time-frequency map, of the chirp signal of GWs from compact binary mergers by using ML.



Lensing Model



[arXiv:1810.07888]

- Lensed GW in frequency domain

$$h_{\text{lens}}(f) = F(f)h(f)$$

where $F(f)$ is the amplification factor which is written as

$$F(f) = \sum_j |\mu_j|^{1/2} e^{2\pi i f \Delta t_{d,j} - i\pi n_j}$$

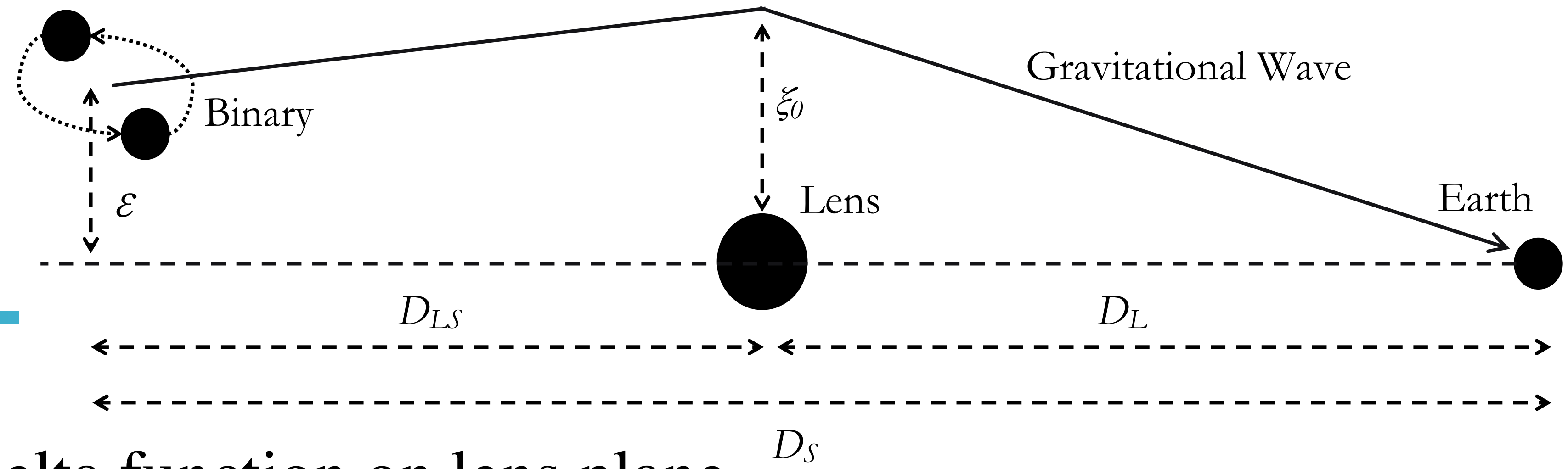
model dependent number

time delay of j-th image

magnification factor

[Takahashi & Nakamura, ApJ 2003]

Lensing Model



- Point-Mass (PM) Lens
  - Lens mass: 2-dim Dirac delta function on lens plane

$$F(f) = |\mu_+|^{1/2} - i\,|\mu_-|^{1/2} e^{2\pi i f \Delta t_d}$$

$$\mu_\pm = \frac{1}{2} \pm \frac{(y^2+2)}{(2y\beta)} \qquad \Delta t_d = 4M_{Lz}\left[\frac{y\beta}{2} + \ln\left(\frac{\beta+y}{\beta-y}\right)\right]$$

$$M_{Lz} = M_L(1+z)$$

$$\beta = \sqrt{y^2+4}$$

$$y = \frac{\varepsilon D_L}{\xi_0 D_S}$$

- Singular Isothermal Sphere (SIS)
  - Lens mass: circular and symmetric distribution on lens plane

$$F(f) = \begin{cases} |\mu_+|^{1/2} - i|\mu_-|^{1/2} e^{2\pi i f \Delta t_d} & \text{if } y < 1, \\ |\mu_+|^{1/2} & \text{if } y \geq 1 \end{cases}$$

$$\mu_\pm = \frac{1}{y} \pm 1 \qquad \Delta t_d = 8M_{Lz}y \qquad y = \frac{\varepsilon D_L}{\xi_0 D_S}$$

Strategy

- Classification
  - classifies lensed signal from unlensed signal.
  - w/ Scikit-Learn and TensorFlow

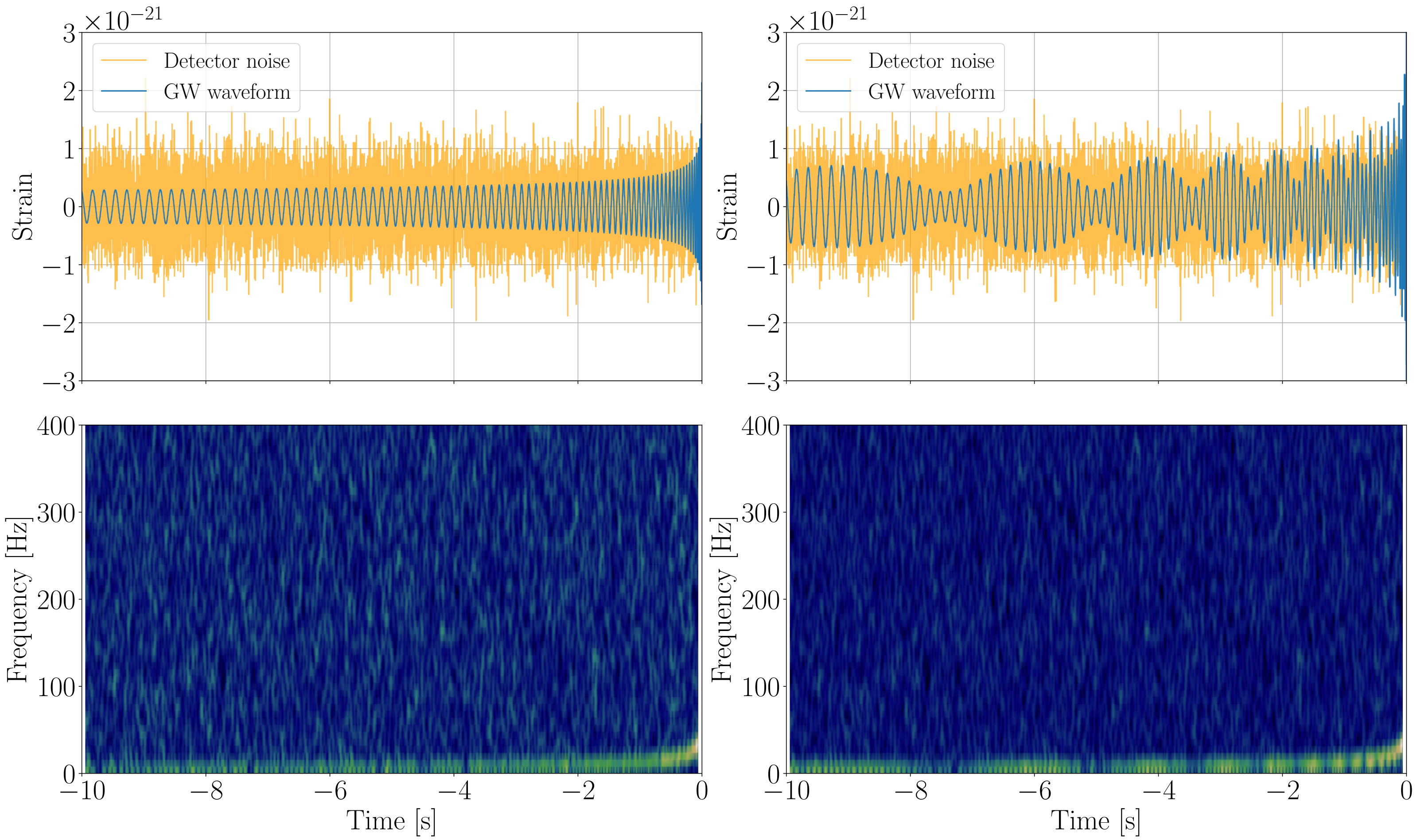
for successfully classified spectrograms

- Regression
  - estimates lensing parameters such as source mass, lens mass, distance, and so on.
  - w/ TensorFlow

reconstructs lensed GWs and compares to input lensed GWs

Classification [arXiv:1810.07888]

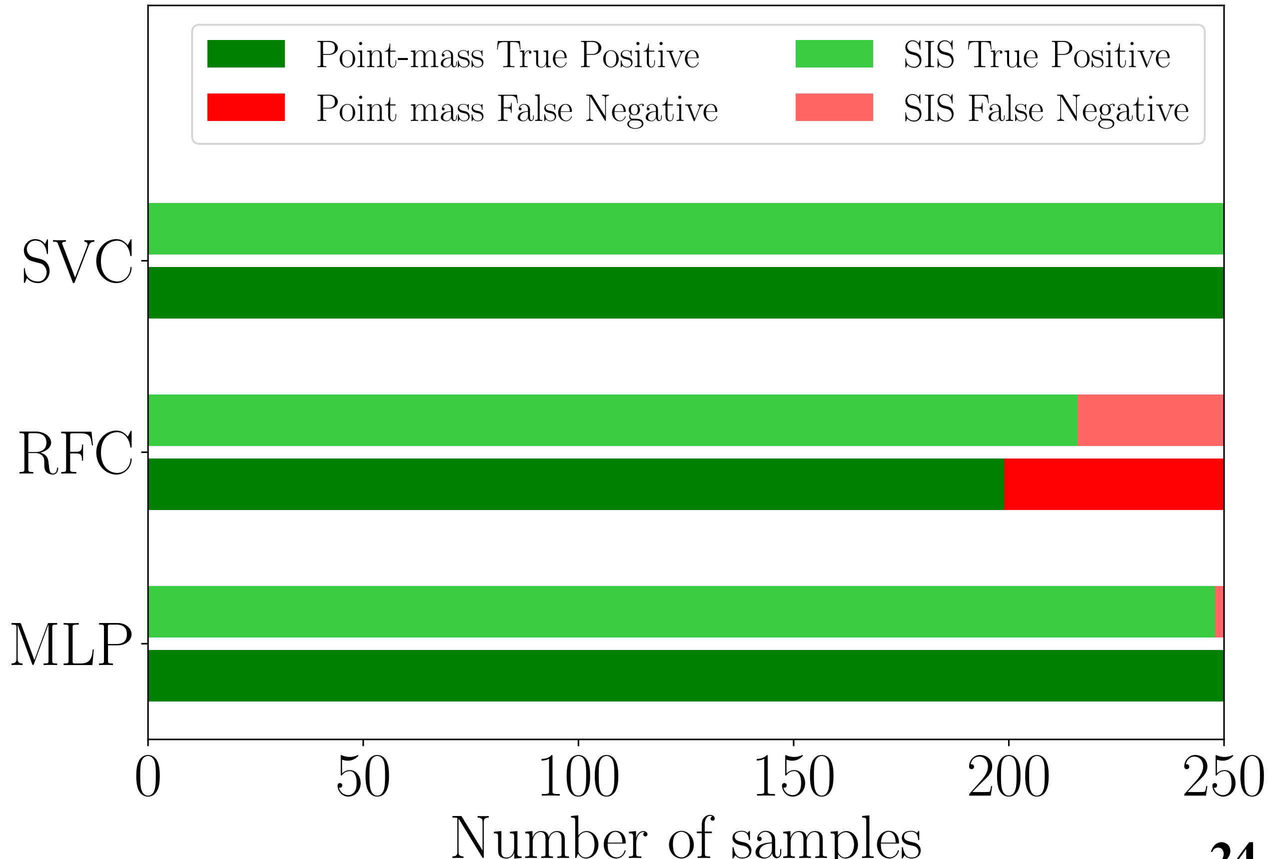
- Input data: 2-dim spectrogram images
  - 2 classes: unlensed vs. lensed
  - # of samples: 1000 samples for unlensed / 1000 samples for lensed (PM or SIS)
  - Parameters
    - m_1, m_2: 4-35 solar mass
    - D_L: 10-1000 Mpc
    - D_{LS}: 10-1000 Mpc
    - M_L: 10-10^7 solar mass
    - ε: 0-0.5 pc
    - redshift factor, z: 0-2
  - Noise: random normal

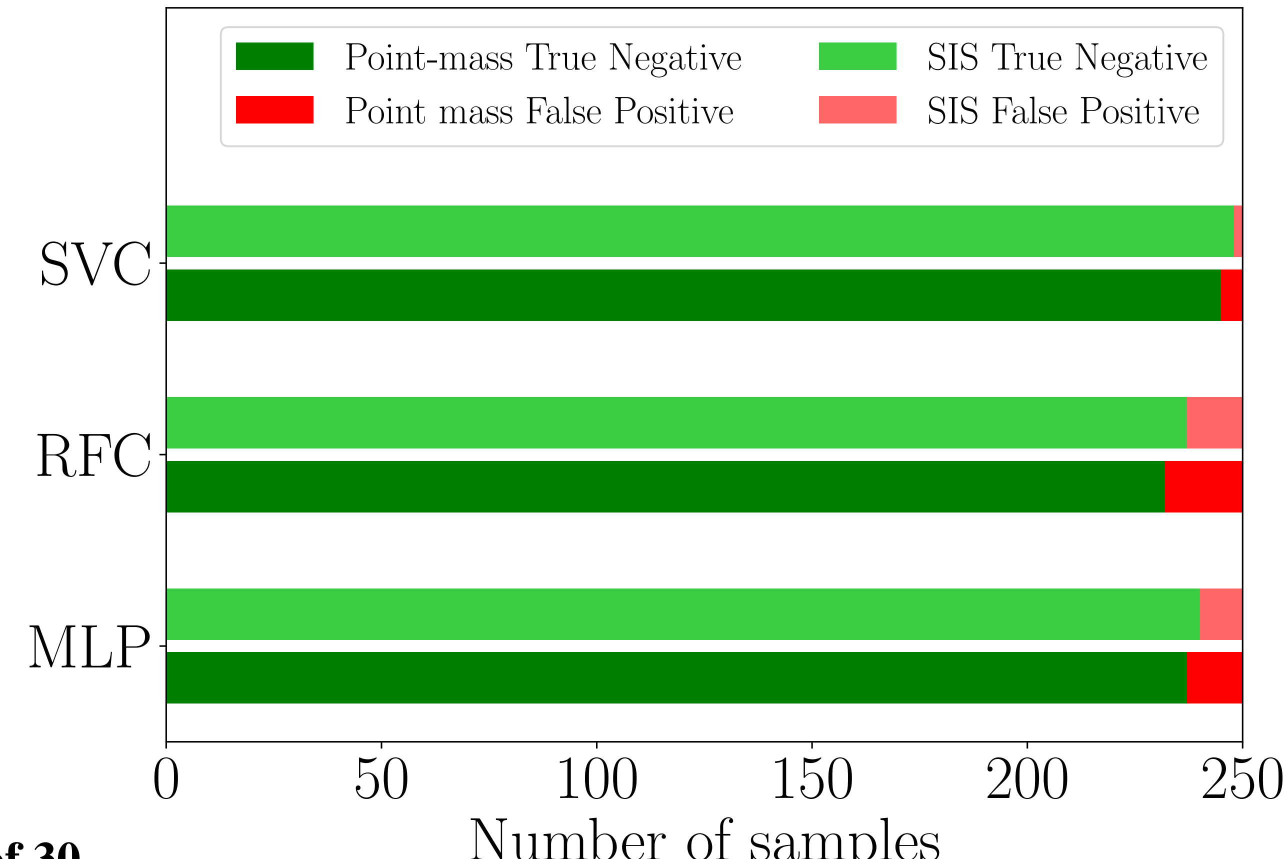


Classification Performance

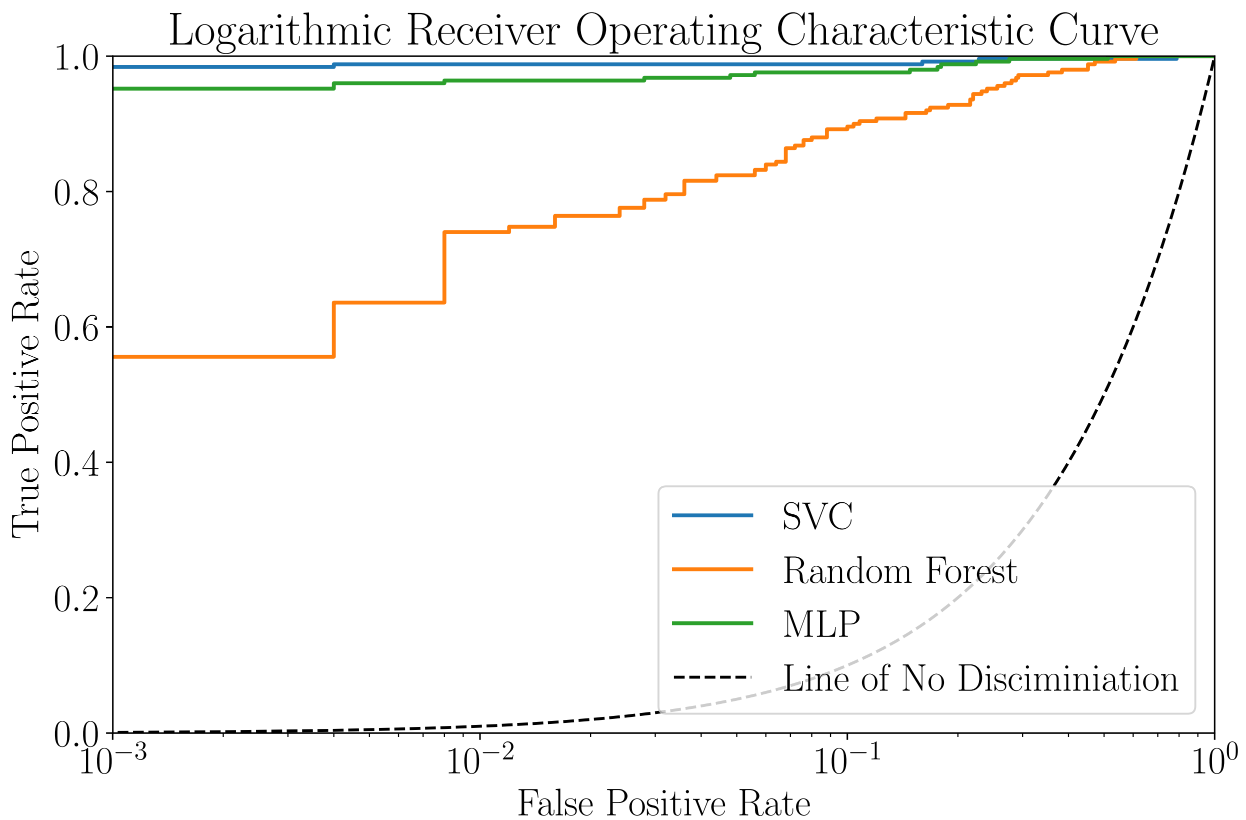
- ML algorithms
  - Support Vector Machine (SVC)
  - Random Forest (RF)
  - Multi-Layer Perceptron (MLP) ↔ Neural Network
- Confusion Matrix
  - Good performance
    - True Positive ~ 1, False Negative ~ 0, True Negative ~ 1, False Positive ~ 0

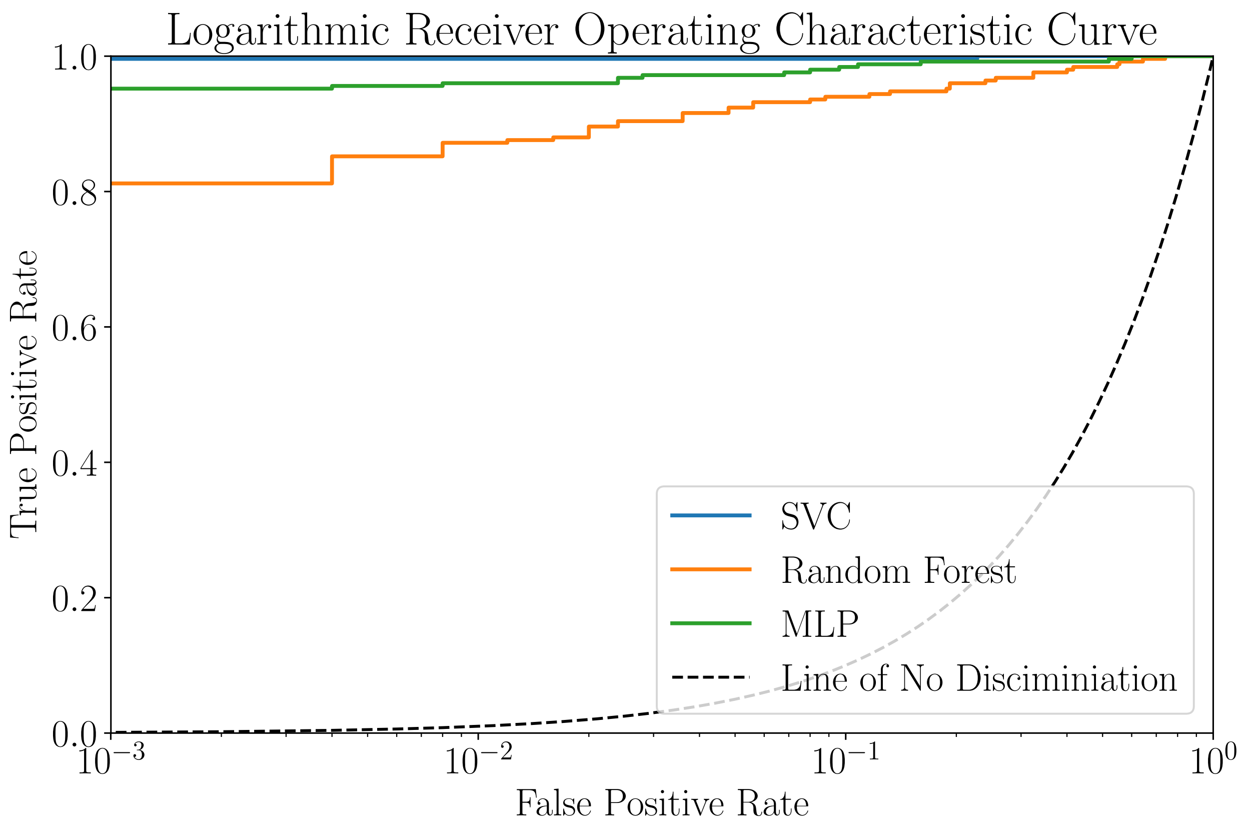
Classifiers	Hyper-parameters	Value
SVC	C	1000
	γ	5×10^{-7}
RFC	N_t	1000
	N_{mss}	2
MLP	α	0.72
	$N_{layer\_neurons}$	1000





ROC Curve



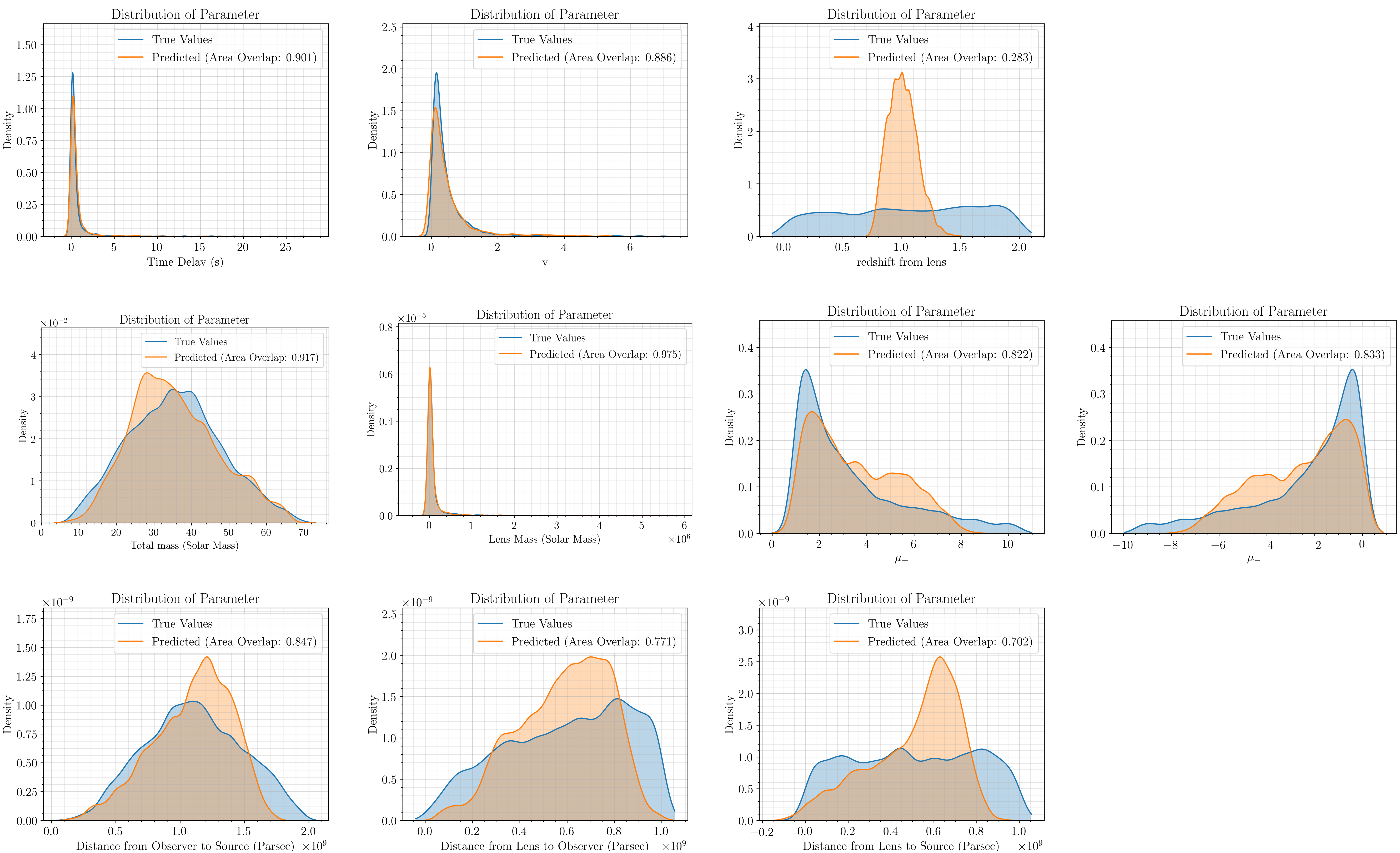


- Area under Curve
  - PM
    - SVC: 0.995 / RF: 0.962 / MLP: 0.993
  - SIS
    - SVC: 0.999 / RF: 0.976 / MLP: 0.993

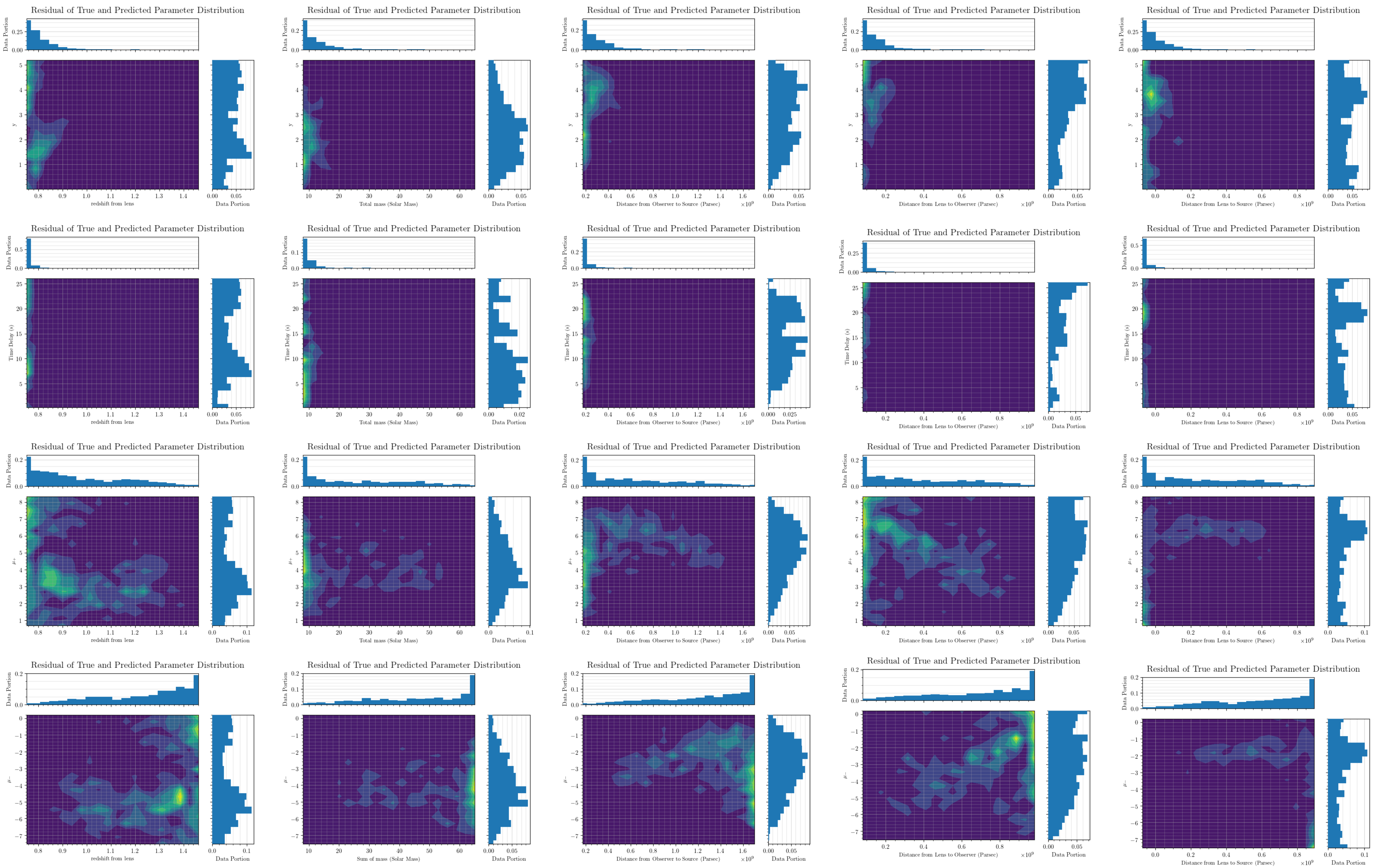
Regression

- Input data
  - 33 699 unlensed spectrograms
  - 40 440 lensed spectrograms (PM lens model only)
  - Parameters: the same with classification
- ML algorithm: neural network (in specific, convolutional neural network)
  - Error measurement: root mean square error

Comparison between True and Predicted Parameters

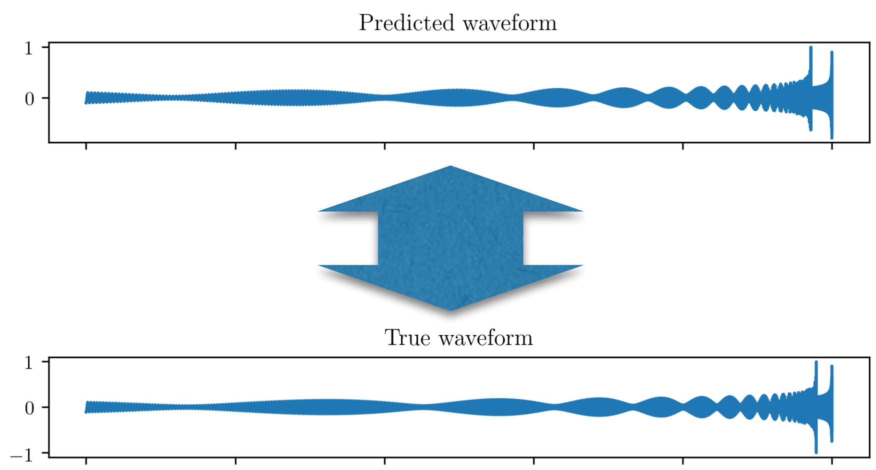


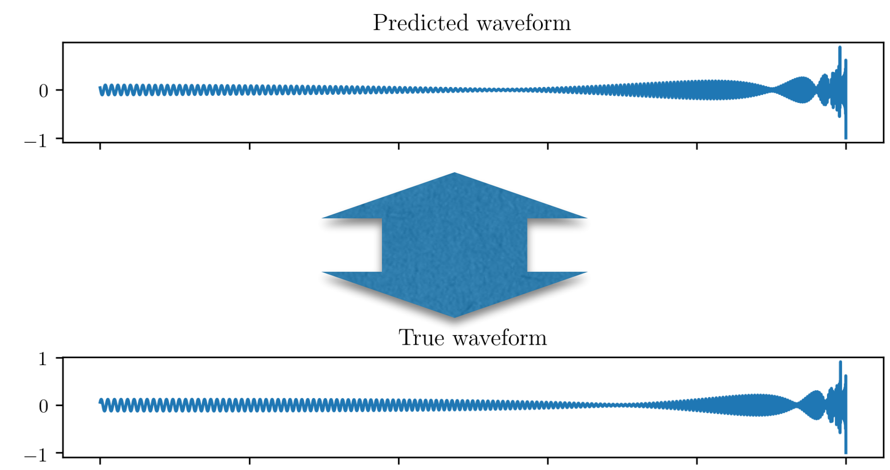
Comparison between True and Predicted Parameters



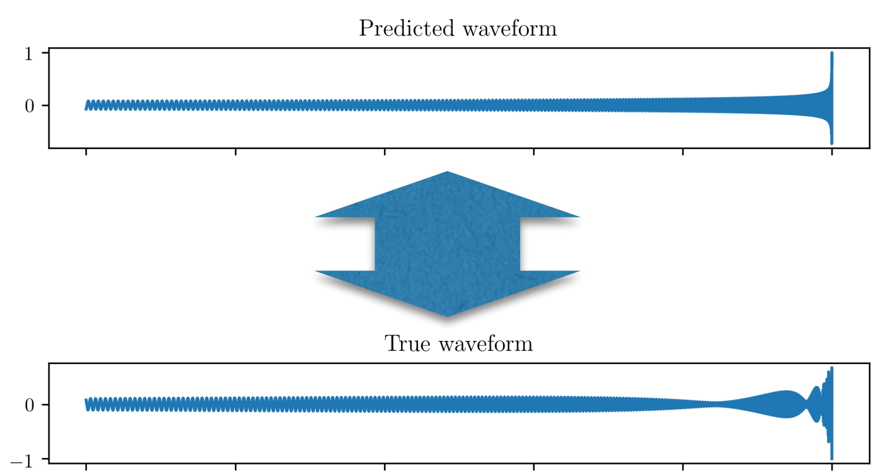
Waveform Reconstruction

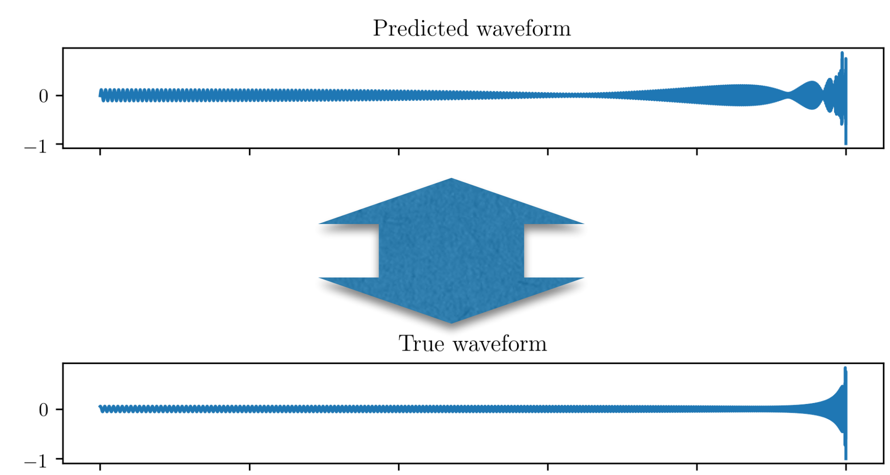
- w/ well estimated parameters





- w/ poorly estimated parameters





Remarks

- Concludes that it
  - Seems promising to classify lensed GWs from spectrograms of unlensed GWs.
    - Still needs more rigorous study on finding the most suitable ML algorithm and its optimal hyperparameters.
  - Seems possible to estimate lensing parameters from regression.
    - Still needs to increase estimation power on poorly estimated parameters.
- Directions of future work
  - Uses LIGO/Virgo noise profile w/ proper restriction on observable SNR in the generation of input image samples.
  - Builds a mature end-to-end framework from classification to regression.

Thank you
for listening!