Lecture Recording

- **Note: These lectures will be recorded and posted onto the IMPRS website**

- Dear participants,
- We will record all lectures on "*Making sense of data: introduction to statistics for gravitational wave astronomy*", including possible Q&A after the presentation, and we will make the recordings publicly available on the IMPRS lecture website at:
    - https://imprs-gw-lectures.aei.mpg.de/2023-making-sense-of-data/
- By participating in this Zoom meeting, you are giving your explicit consent to the recording of the lecture and the publication of the recording on the course website.

Making sense of data: introduction to statistics for gravitational wave astronomy

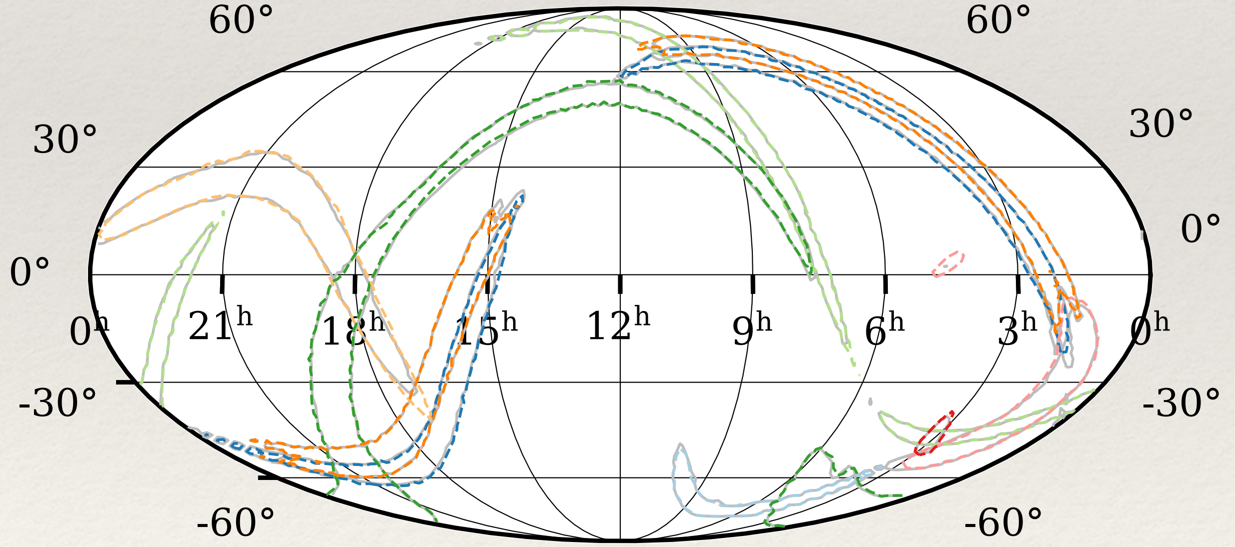
Part III: Machine Learning

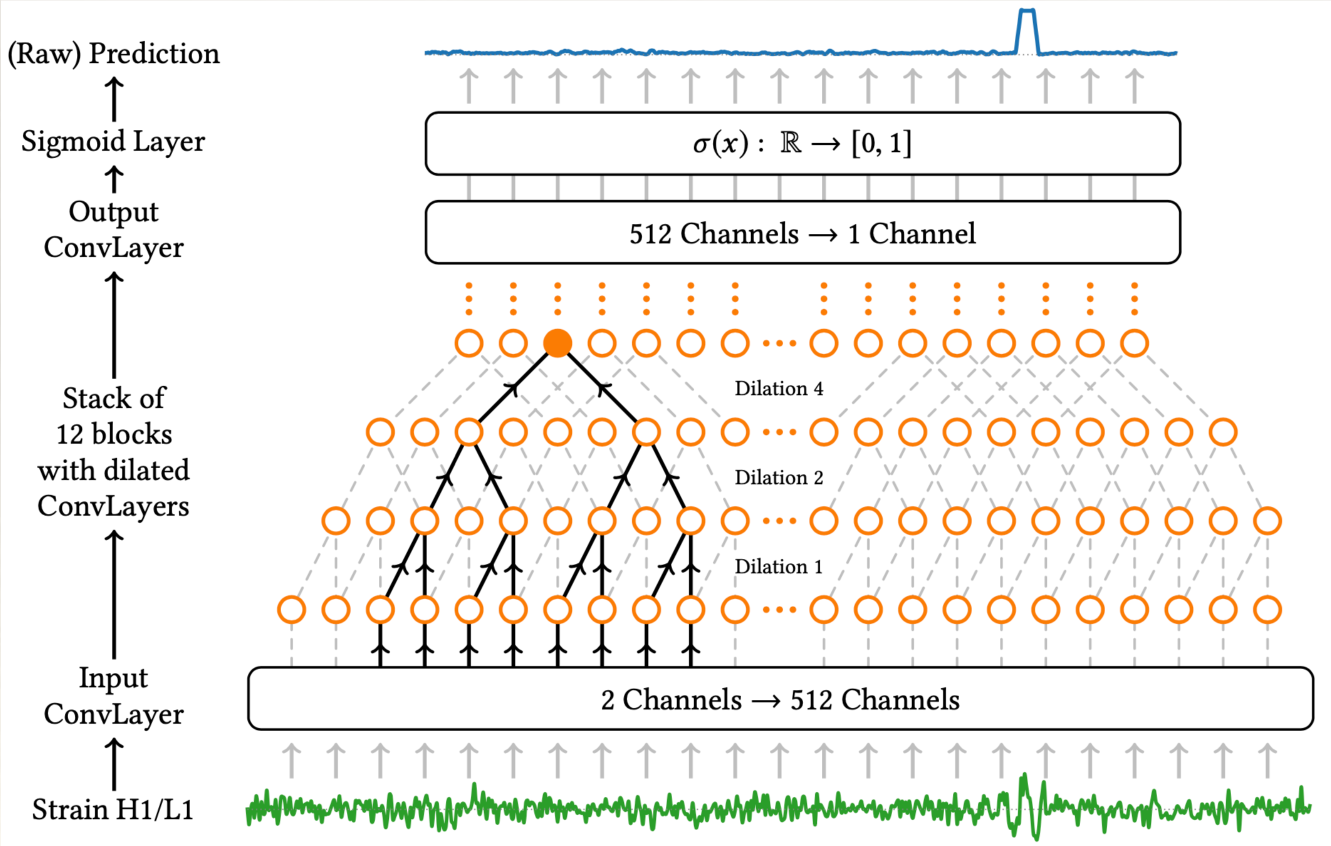
Lecture 3: Machine learning for GW astronomy

AEI IMPRS Lecture Course

Jonathan Gair jgair@aei.mpg.de

Thanks to **Stephen Green** for producing much of the material for this course in 2021!





Introduction

- Many **applications** so far of machine learning to gravitational waves:
  - searches, parameter estimation, waveform modeling, glitch classification, population inference, noise reduction, …

(somewhat biased presentation)

- **Objectives** of these applications:
  - Usually **speed of calculations**
  - Increasingly (and I believe in the future) **previously-intractable analyses**

Searches

- George and Huerta (2017), **Gabbard et al (2018)**, Gebhard et al (2019), …
- Approach is to train a **classifier** to distinguish **signal + noise** vs **noise only.**
- **Dataset:**
  - time domain data, $T = 1\ \mathrm{s}, f_s = 8192\ \mathrm{Hz}$
  - 5×10^5 samples; half with signal, half without; whitened
  - IMRPhenomD, $5\ \mathrm{M}_\odot \leq m_{1,2} \leq 95\ \mathrm{M}_\odot$, zero spin, $0.65\ \mathrm{s} \leq t_c \leq 0.85\ \mathrm{s}$
- **Probability model:** 2 softmax outputs representing $p(y = i \,|\, \boldsymbol{x})$
- **Loss:** Binary cross-entropy

$$f(\theta) = -\sum_{i\in S} \log(\theta_i^S) - \sum_{i\in N} \log(\theta_i^N)$$

Searches

- **1D convolutional architecture** captures time-translation invariance of the data

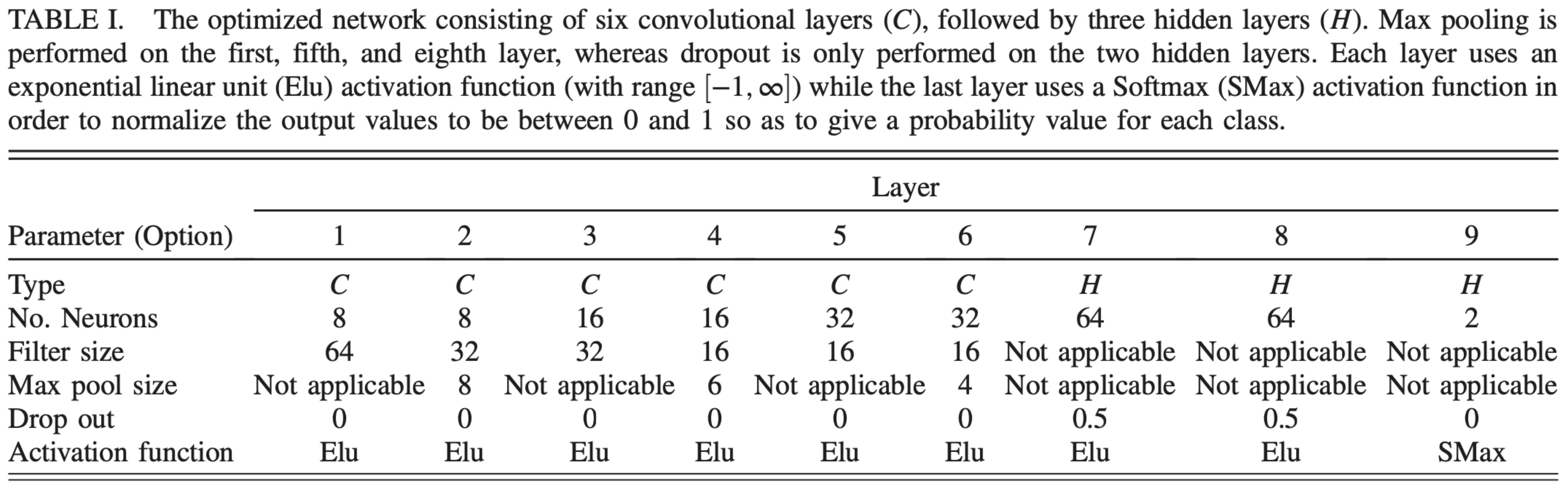


TABLE I. The optimized network consisting of six convolutional layers (C), followed by three hidden layers (H). Max pooling is performed on the first, fifth, and eighth layer, whereas dropout is only performed on the two hidden layers. Each layer uses an exponential linear unit (Elu) activation function (with range $[-1, \infty]$) while the last layer uses a Softmax (SMax) activation function in order to normalize the output values to be between 0 and 1 so as to give a probability value for each class.

	Layer								
Parameter (Option)	1	2	3	4	5	6	7	8	9
Type	C	C	C	C	C	C	H	H	H
No. Neurons	8	8	16	16	32	32	64	64	2
Filter size	64	32	32	16	16	16	Not applicable	Not applicable	Not applicable
Max pool size	Not applicable	8	Not applicable	6	Not applicable	4	Not applicable	Not applicable	Not applicable
Drop out	0	0	0	0	0	0	0.5	0.5	0
Activation function	Elu	Elu	Elu	Elu	Elu	Elu	Elu	Elu	SMax

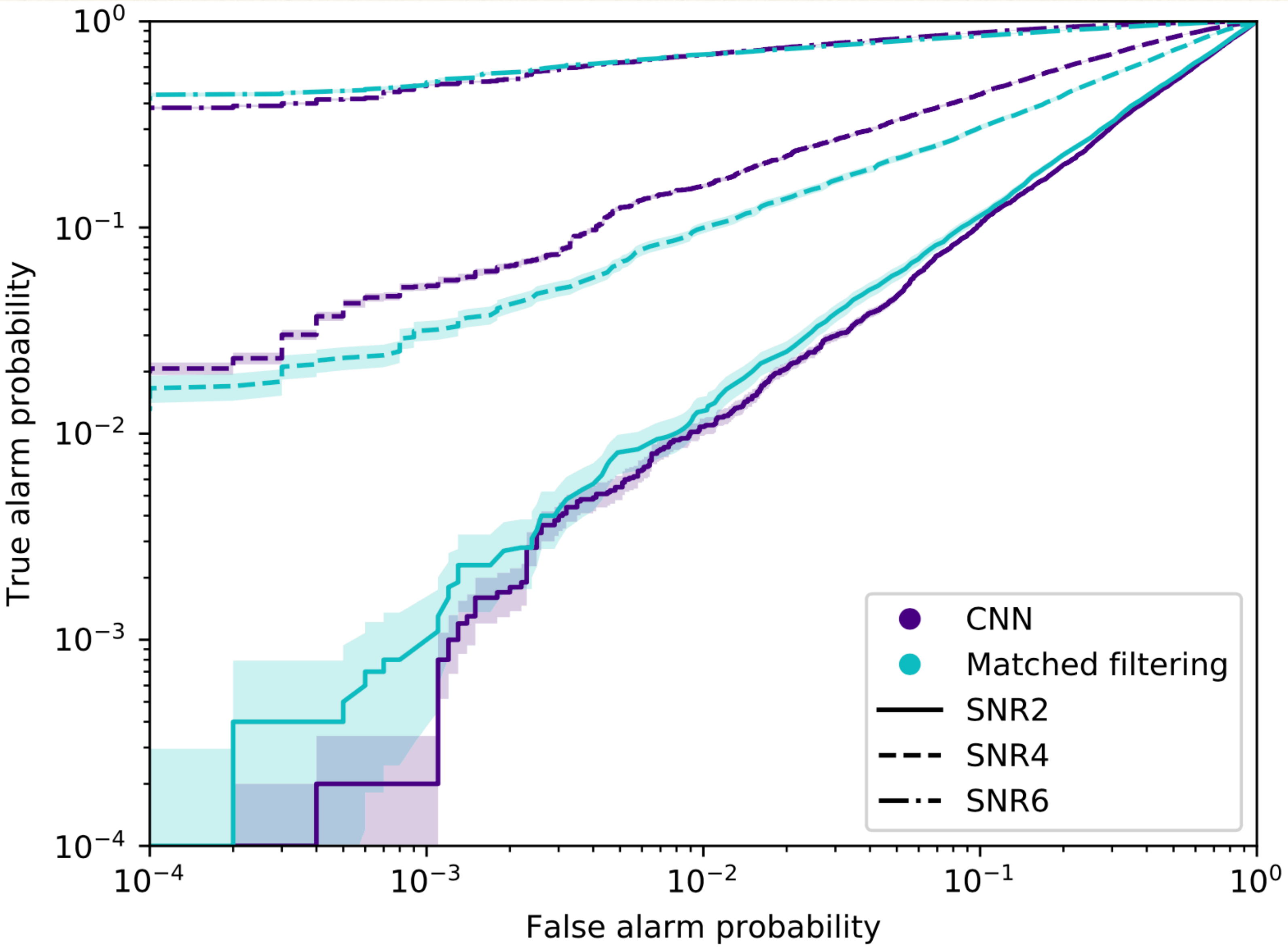
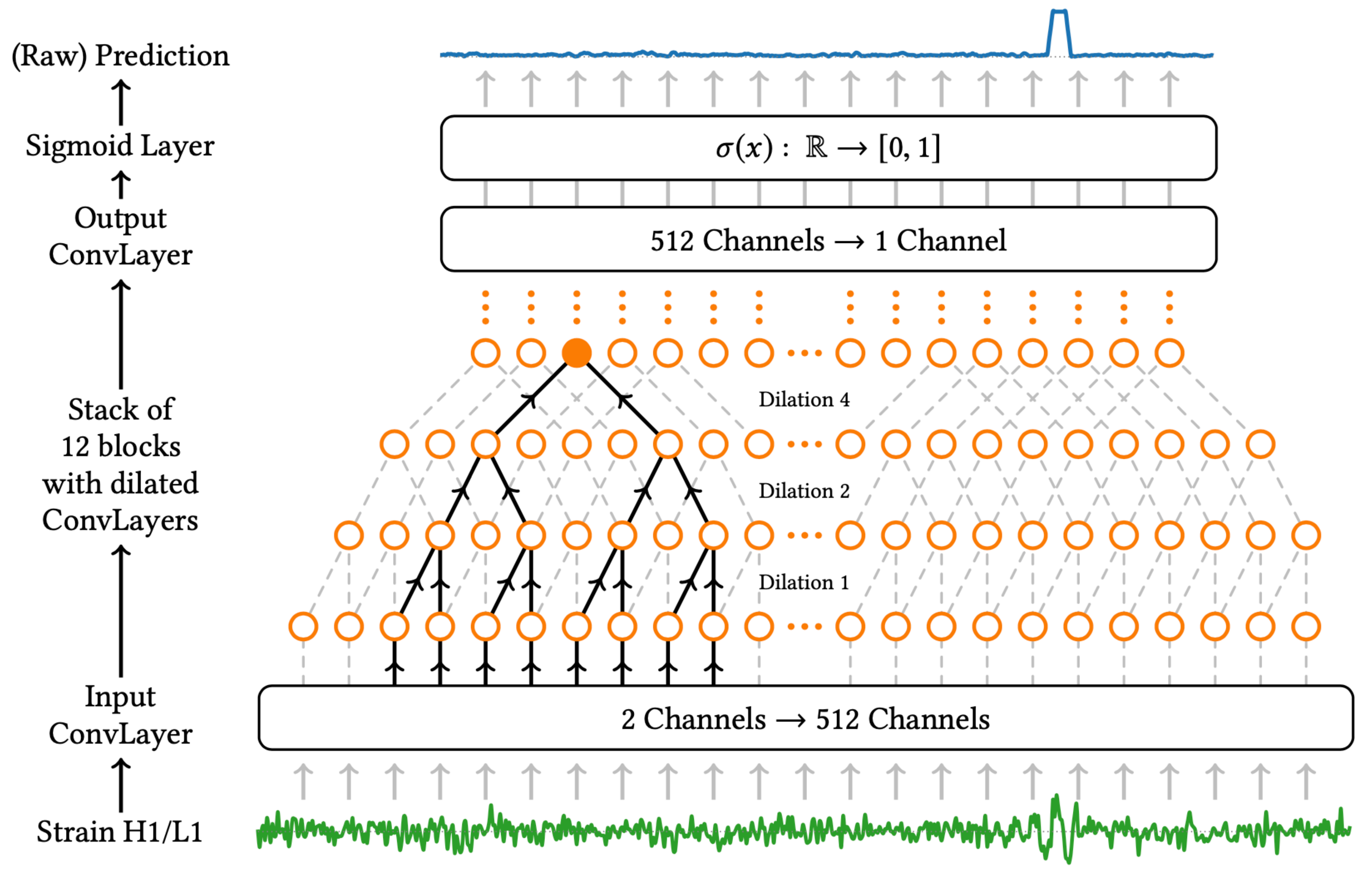


FIG. 2. The ROC curves for test data sets containing signals with optimal SNR, $\rho_{\text{opt}} = 2, 4, 6$. We plot the true alarm probability versus the false alarm probability estimated from the output of the CNN (purple) and matched-filtering (cyan) approaches. Uncertainties in the true alarm probability correspond to 1-σ bounds assuming a binomial distribution.

Other search approaches

- Gebhard et al (2019) used **dilated convolutions** for longer data sets.



Waveform modeling

- **Chua, Galley, Vallisneri (2019):** Reduced Order Modeling with Artificial Neurons (ROMAN)

- Expand waveforms in reduced basis ($\sim 10^2$ elements) and interpolate with neural network

$$h(\theta) = \sum_i \langle h(\theta) | e_i \rangle e_i := \sum_i \alpha_i(\theta) e_i \equiv \alpha(\theta)$$

- **Training data:** 6×10^5 pairs $\{\theta_n, \alpha(\theta_n)\}$

- **Loss function:**

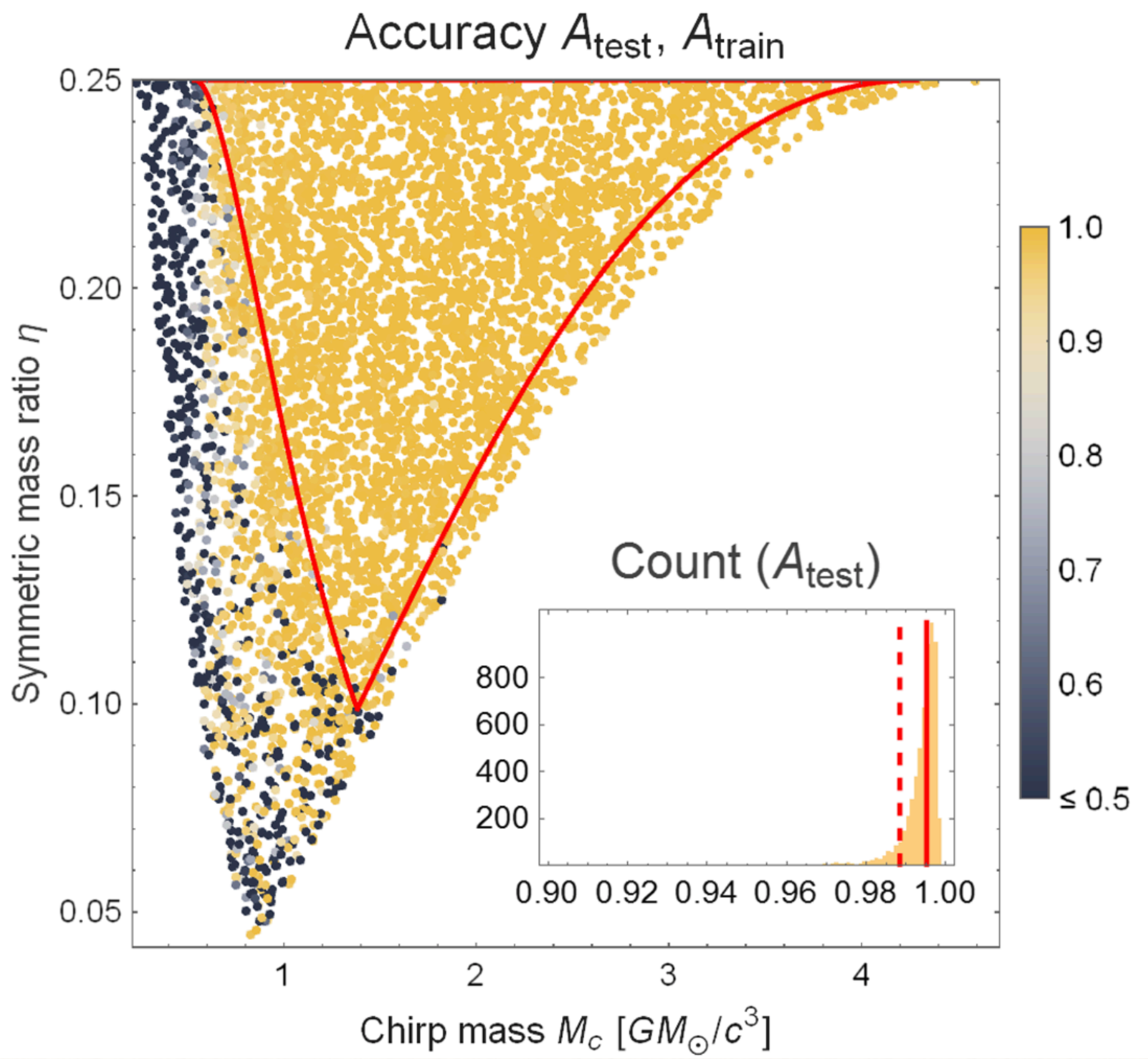
$$L := \frac{\langle |\alpha - \hat{\alpha}|^2 \rangle}{\sqrt{\langle |\hat{\alpha}|^2 \rangle}}$$

Engineered to give more weight to later basis elements

- **Fully-connected network:** 25 hidden layers x 256 units

Waveform modeling

- **Initial work:** TaylorF2; four parameters $\theta = (m_1, m_2, \chi_1, \chi_2)$



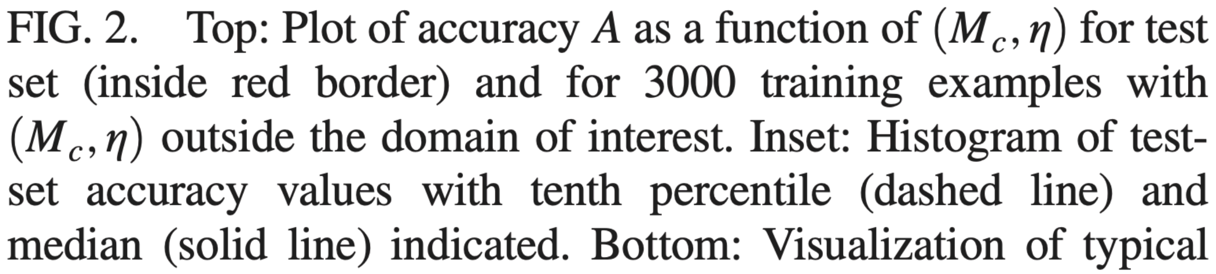
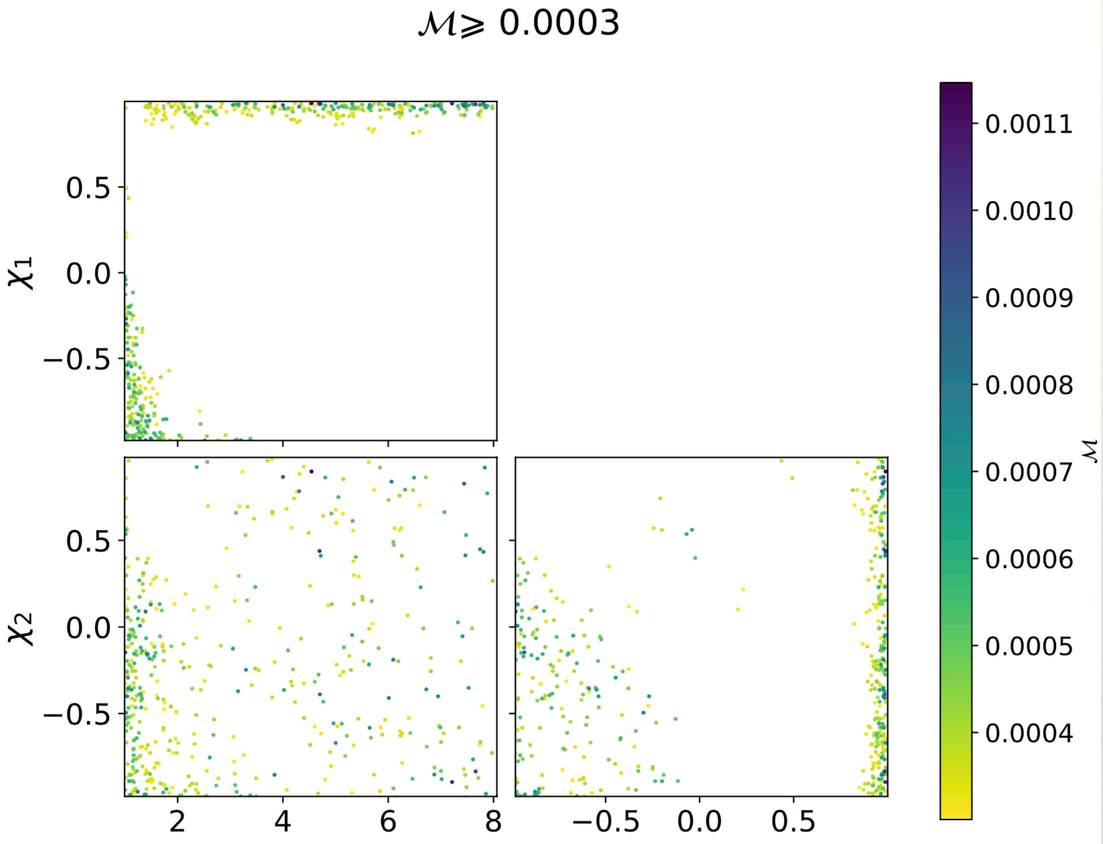


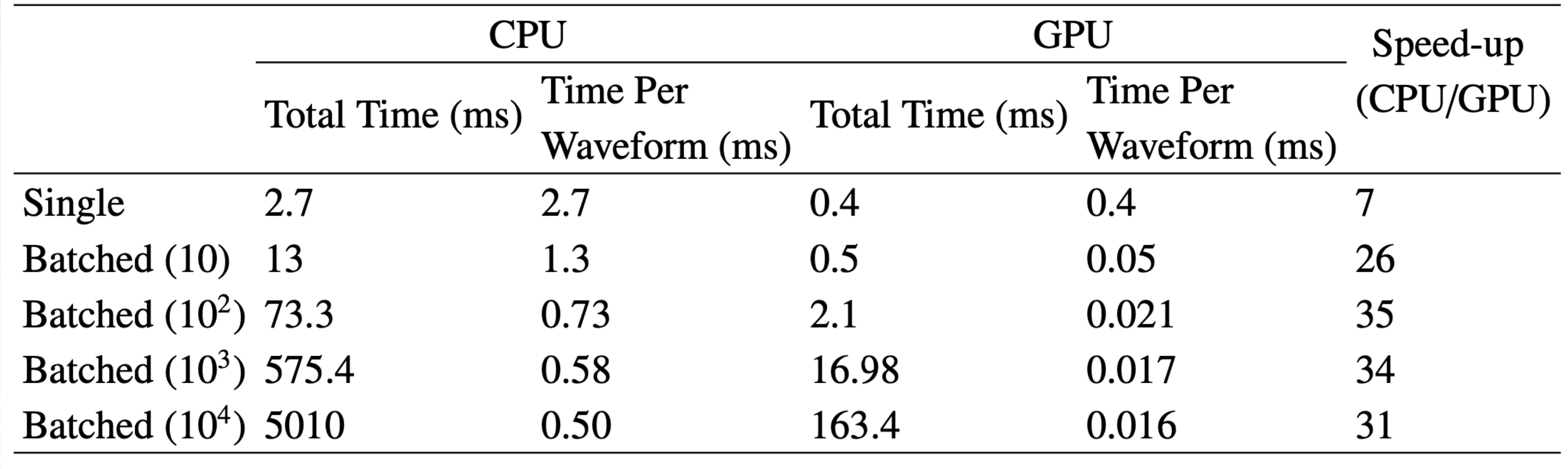
FIG. 2. Top: Plot of accuracy A as a function of (M_c, η) for test set (inside red border) and for 3000 training examples with (M_c, η) outside the domain of interest. Inset: Histogram of test-set accuracy values with tenth percentile (dashed line) and median (solid line) indicated. Bottom: Visualization of typical

- Automatic differentiation enables gradient-based sampling methods (e.g., Hamiltonian Monte Carlo)

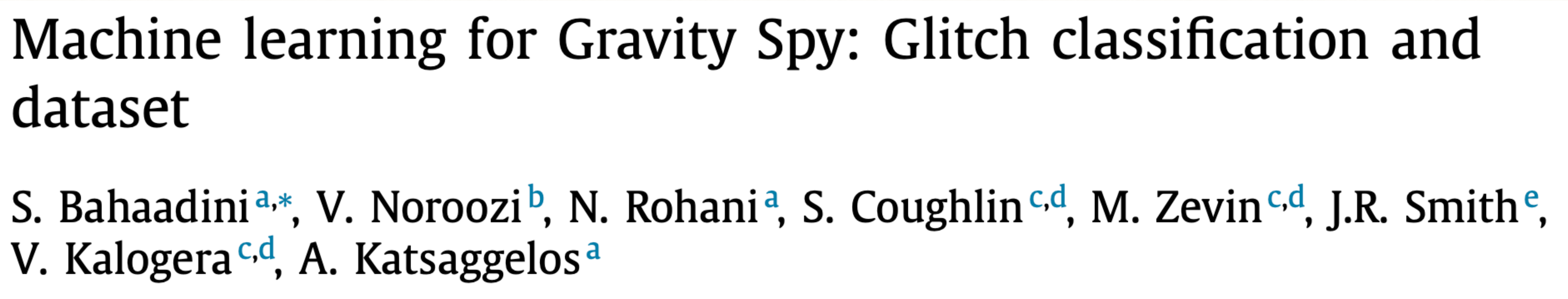
Waveform modeling

- More recent work **(S. Khan and R. Green, 2021)** uses similar techniques applied to SEOBNRv4.
    - Fit to amplitude and phase



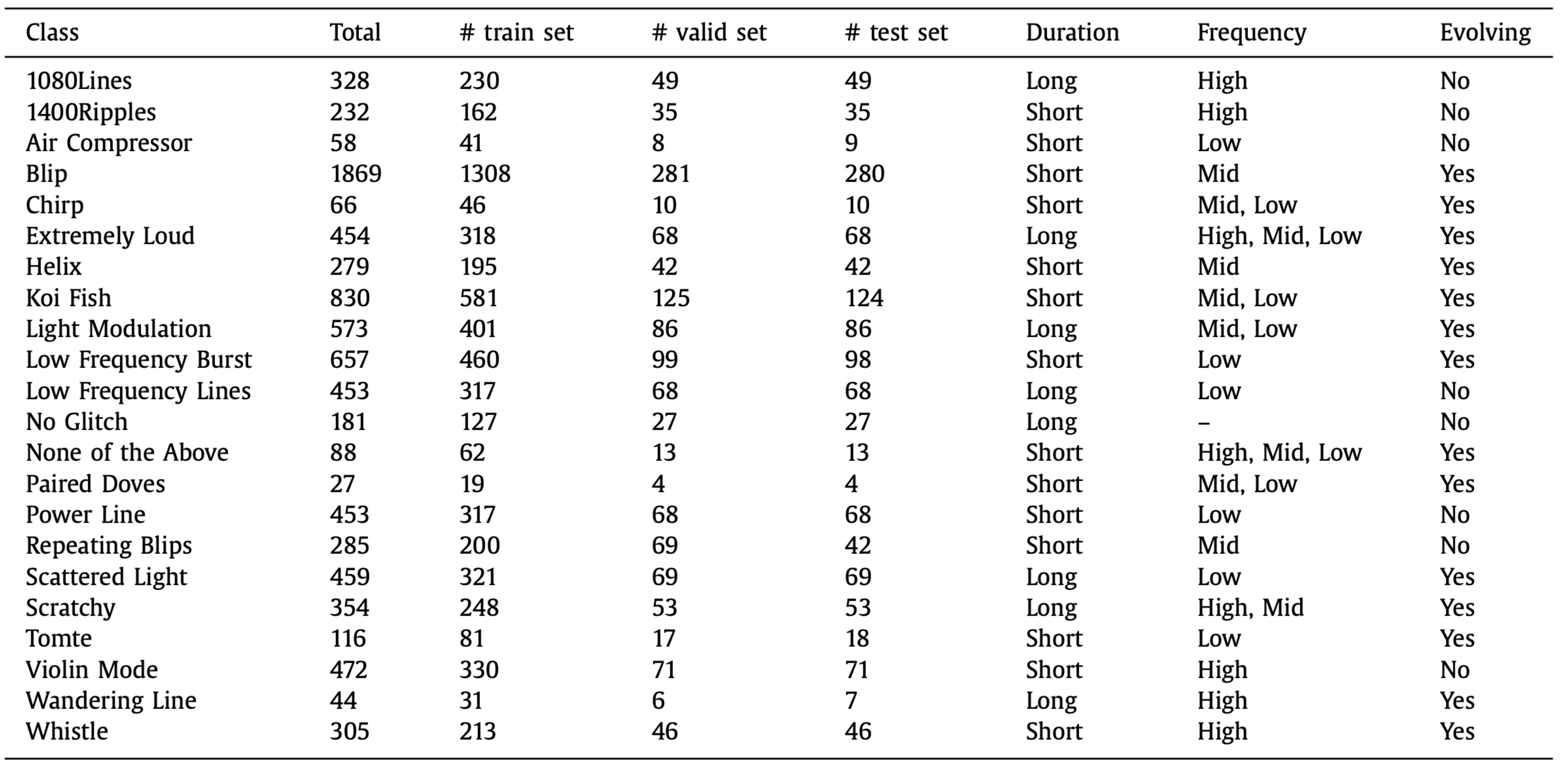


	CPU		GPU		Speed-up (CPU/GPU)
	Total Time (ms)	Time Per Waveform (ms)	Total Time (ms)	Time Per Waveform (ms)	
Single	2.7	2.7	0.4	0.4	7
Batched (10)	13	1.3	0.5	0.05	26
Batched (10^2)	73.3	0.73	2.1	0.021	35
Batched (10^3)	575.4	0.58	16.98	0.017	34
Batched (10^4)	5010	0.50	163.4	0.016	31



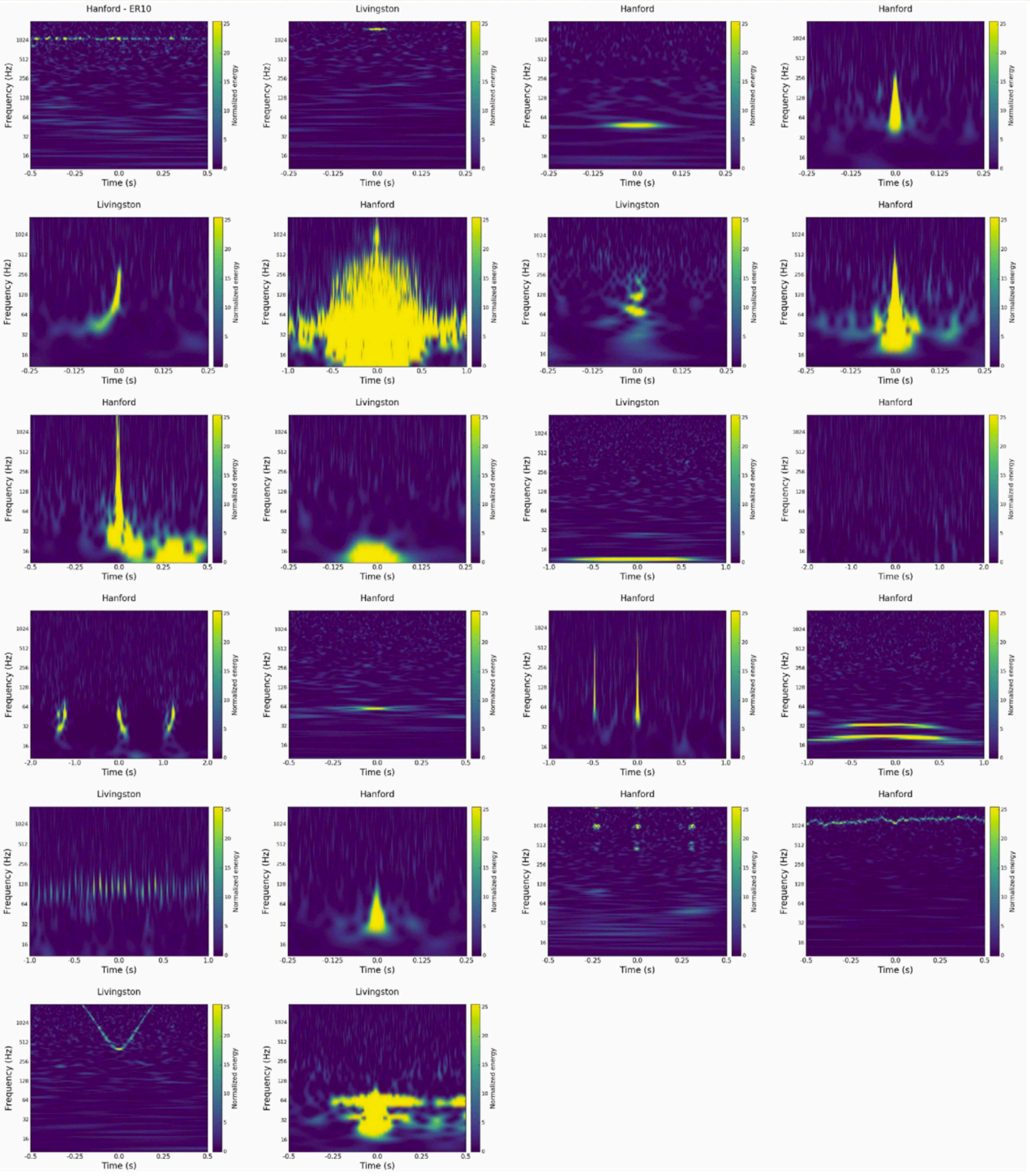
Machine learning for Gravity Spy: Glitch classification and dataset

S. Bahaadini[a,*], V. Noroozi[b], N. Rohani[a], S. Coughlin[c,d], M. Zevin[c,d], J.R. Smith[e], V. Kalogera[c,d], A. Katsaggelos[a]



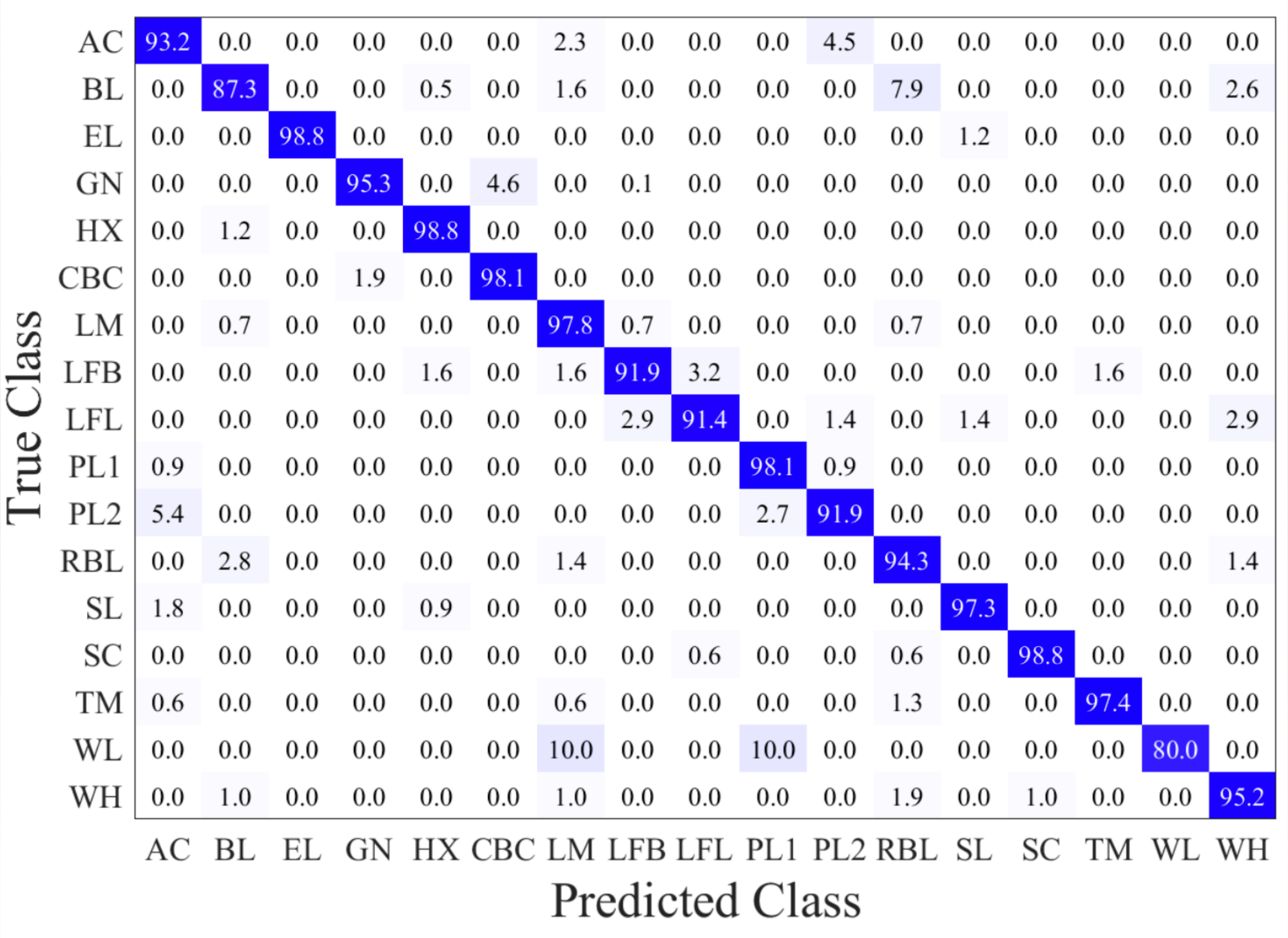
Class	Total	# train set	# valid set	# test set	Duration	Frequency	Evolving
1080Lines	328	230	49	49	Long	High	No
1400Ripples	232	162	35	35	Short	High	No
Air Compressor	58	41	8	9	Short	Low	No
Blip	1869	1308	281	280	Short	Mid	Yes
Chirp	66	46	10	10	Short	Mid, Low	Yes
Extremely Loud	454	318	68	68	Long	High, Mid, Low	Yes
Helix	279	195	42	42	Short	Mid	Yes
Koi Fish	830	581	125	124	Short	Mid, Low	Yes
Light Modulation	573	401	86	86	Long	Mid, Low	Yes
Low Frequency Burst	657	460	99	98	Short	Low	Yes
Low Frequency Lines	453	317	68	68	Long	Low	No
No Glitch	181	127	27	27	Long	–	No
None of the Above	88	62	13	13	Short	High, Mid, Low	Yes
Paired Doves	27	19	4	4	Short	Mid, Low	Yes
Power Line	453	317	68	68	Short	Low	No
Repeating Blips	285	200	69	42	Short	Mid	No
Scattered Light	459	321	69	69	Long	Low	Yes
Scratchy	354	248	53	53	Long	High, Mid	Yes
Tomte	116	81	17	18	Short	Low	Yes
Violin Mode	472	330	71	71	Short	High	No
Wandering Line	44	31	6	7	Long	High	Yes
Whistle	305	213	46	46	Short	High	Yes

Types of glitches in database



Other search approaches

- Jadhav et al (2021) used **transfer learning** with InceptionV3 network



True Class

	AC	BL	EL	GN	HX	CBC	LM	LFB	LFL	PL1	PL2	RBL	SL	SC	TM	WL	WH
AC	93.2	0.0	0.0	0.0	0.0	0.0	2.3	0.0	0.0	0.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0
BL	0.0	87.3	0.0	0.0	0.5	0.0	1.6	0.0	0.0	0.0	0.0	7.9	0.0	0.0	0.0	0.0	2.6
EL	0.0	0.0	98.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0	0.0	0.0
GN	0.0	0.0	0.0	95.3	0.0	4.6	0.0	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
HX	0.0	1.2	0.0	0.0	98.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CBC	0.0	0.0	0.0	1.9	0.0	98.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LM	0.0	0.7	0.0	0.0	0.0	0.0	97.8	0.7	0.0	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0
LFB	0.0	0.0	0.0	0.0	1.6	0.0	1.6	91.9	3.2	0.0	0.0	0.0	0.0	0.0	1.6	0.0	0.0
LFL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.9	91.4	0.0	1.4	0.0	1.4	0.0	0.0	0.0	2.9
PL1	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	98.1	0.9	0.0	0.0	0.0	0.0	0.0	0.0
PL2	5.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.7	91.9	0.0	0.0	0.0	0.0	0.0	0.0
RBL	0.0	2.8	0.0	0.0	0.0	0.0	1.4	0.0	0.0	0.0	0.0	94.3	0.0	0.0	0.0	0.0	1.4
SL	1.8	0.0	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	97.3	0.0	0.0	0.0	0.0
SC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.6	0.0	98.8	0.0	0.0	0.0
TM	0.6	0.0	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	1.3	0.0	0.0	97.4	0.0	0.0
WL	0.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	0.0	80.0	0.0
WH	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.9	0.0	1.0	0.0	0.0	95.2

Predicted Class

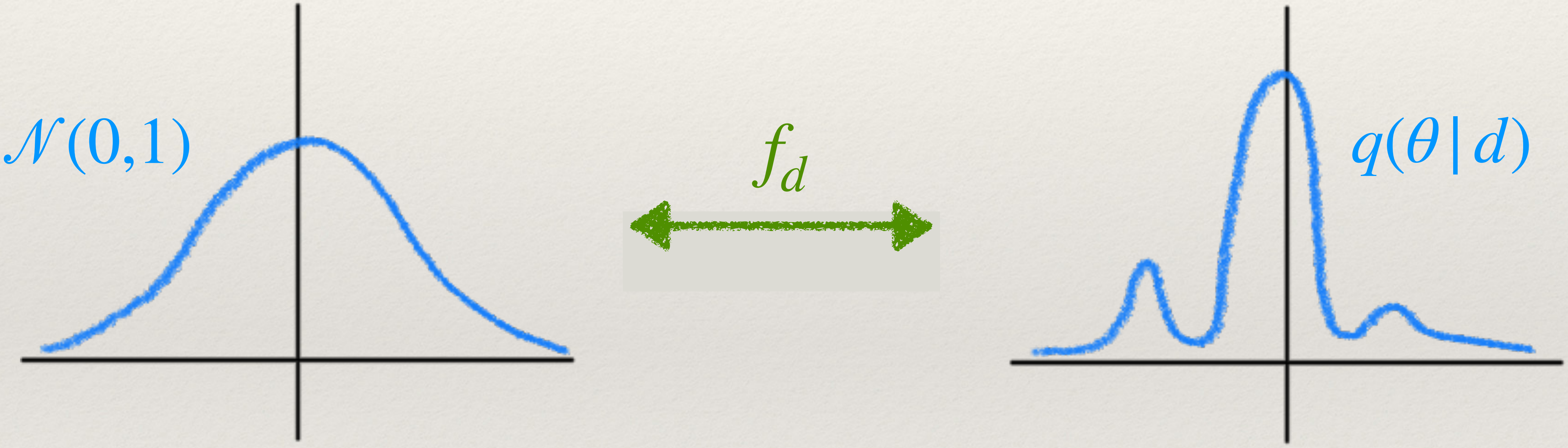
Parameter estimation

- George and Huerta (2018): point estimate
- Gabbard et al (2019): variational autoencoder
- Chua and Vallisneri (2020): Gaussian with learned covariance, histogram
- **Dax et al (2021):** normalizing flow

method to build very
complicated distributions

Normalizing flow

- A **normalizing flow** $f_d : u \mapsto \theta$ defines a complex distribution in terms of a simple one



$$q(\theta|d) = \mathcal{N}(0,1)^D\left(f_d^{-1}(\theta)\right)\left|\det J_{f_d}^{-1}\right|$$

Normalizing flow

$$q(\theta\,|\,d) = \mathcal{N}(0,1)^D\big(f_d^{-1}(\theta)\big)\,\left|\det J_{f_d}^{-1}\right|$$

- **Requirements:**
  1. Invertible
  2. Simple Jacobian determinant
- Parametrize f_d using **neural network.**

Fast to **evaluate** and **sample** from $q(\theta\,|\,d)$

needed for computing
cross-entropy loss

Normalizing flow

- **Requirements:**

  1. Invertible ✓
  2. Simple Jacobian determinant ✓

$$\det J_{f_d} = \prod_{i=\frac{D}{2}+1}^{D} c_i'\left(u_i; u_{1:\frac{D}{2}}, d\right)$$

- Use a sequence of "coupling transforms":

$$c_{d,i}(u) = \begin{cases} u_i & \text{if } i \leq D/2 \\ c_i\left(u_i; u_{1:\frac{D}{2}}, d\right) & \text{if } i > D/2 \end{cases}$$

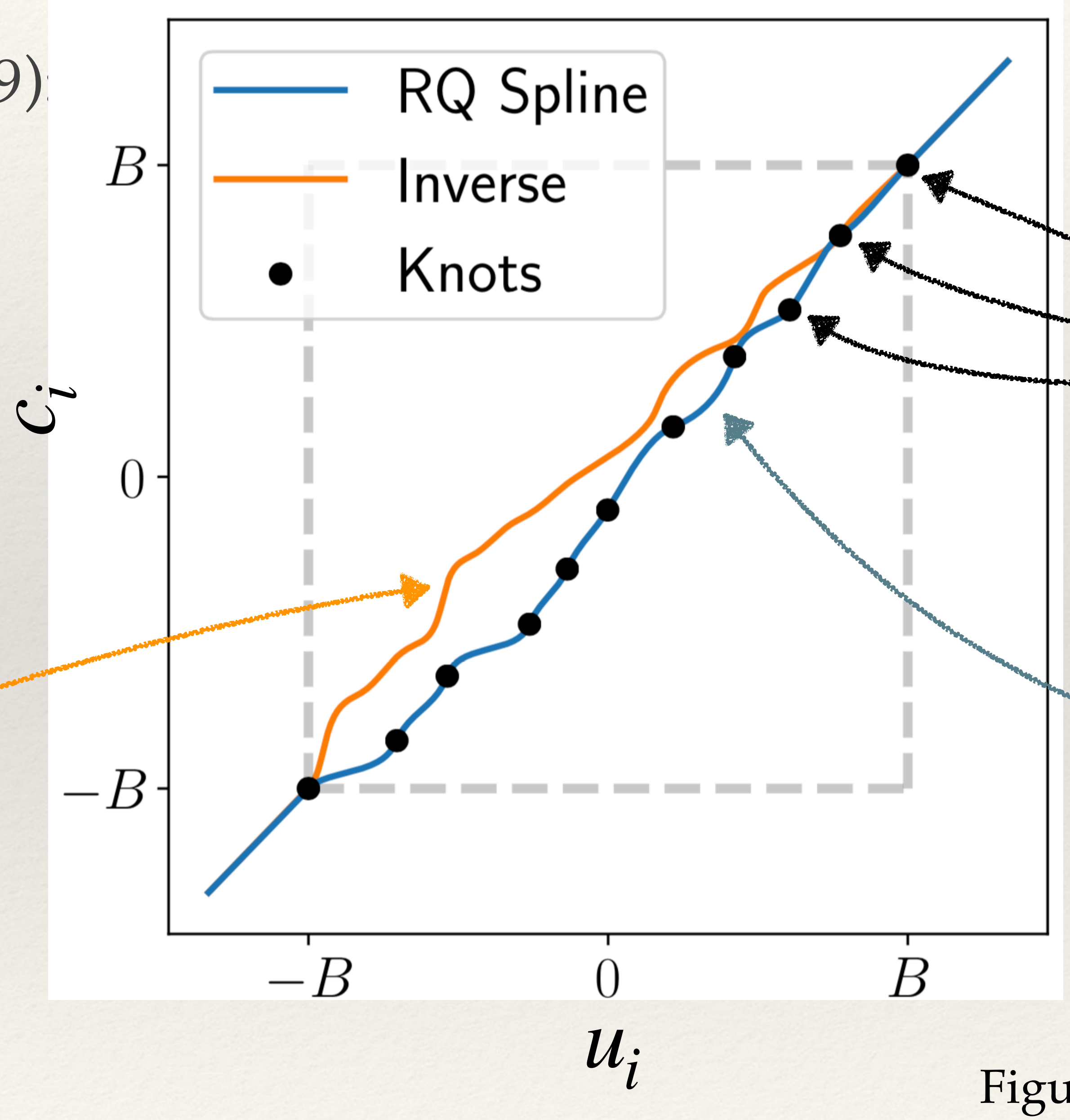
Hold fixed half of the components

Transform remaining components element-wise, conditional on other half and *s*.

- c_i should be **differentiable** and have **analytic inverse** with respect to u_i.

Normalizing flow

- **Spline flow** (Durkan et al, 2019):



knots and **derivatives** output of neural network; input $(u_{1:\frac{D}{2}}, d)$

rational-quadratic spline interpolation

analytic inverse

Figure: Durkan *et al* (2019)

Normalizing flow

- Sequence of flows can give very complicated distribution

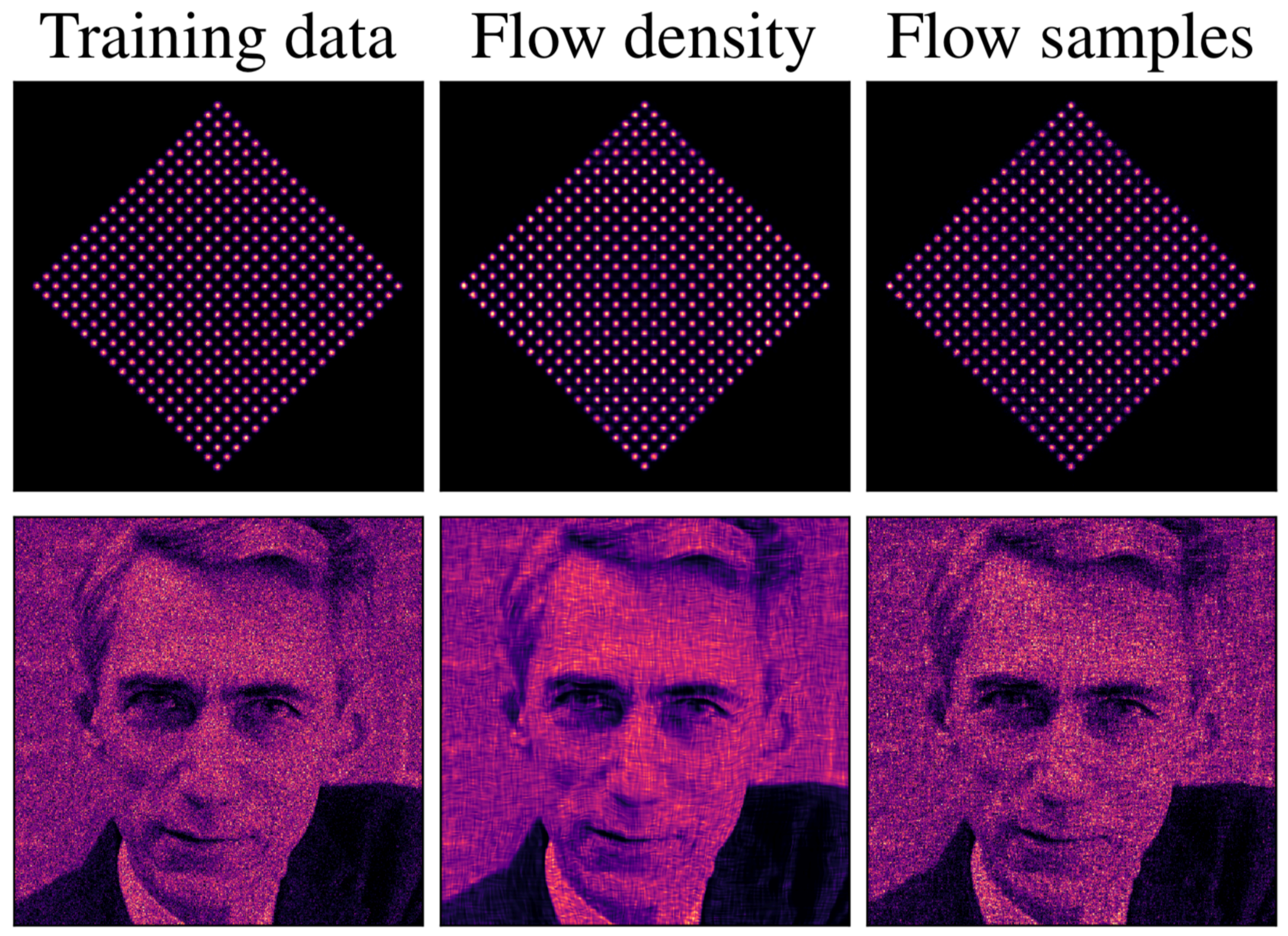
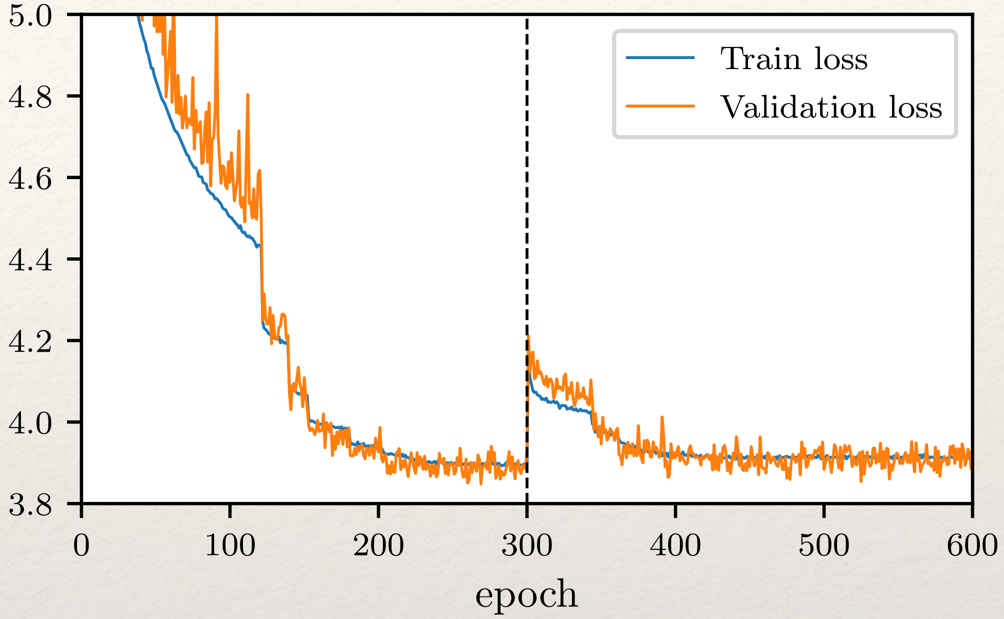


Image: Durkan *et al* (2019)

Training

- 5×10^6 training waveforms
  - IMRPhenomPv2
  - $T = 8$ s, $f_{\rm min} = 20$ Hz, $f_{\rm max} = 1024$ Hz
  - 15D parameter space
  - $m_1, m_2 \in [10,80]$ $\mathrm{M}_\odot$



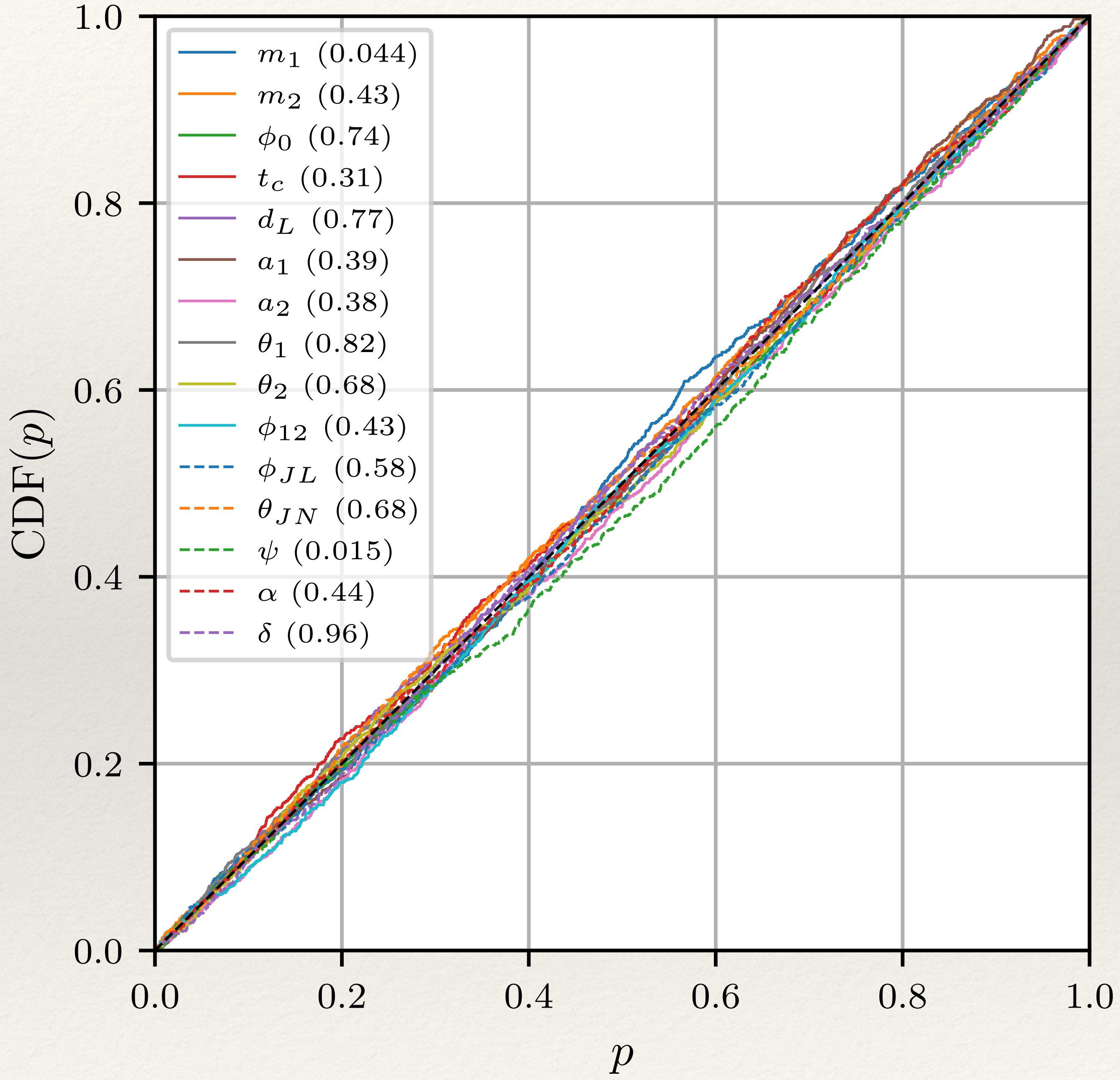
- + stationary Gaussian noise realizations

- Train several neural networks based on different noise level / number of detectors / distance range:

Observing run	Detectors	Distance range [Mpc]
O1	HL	[100, 2000]
O2	HL	[100, 2000] [100, 6000]
	HLV	[100, 1000]

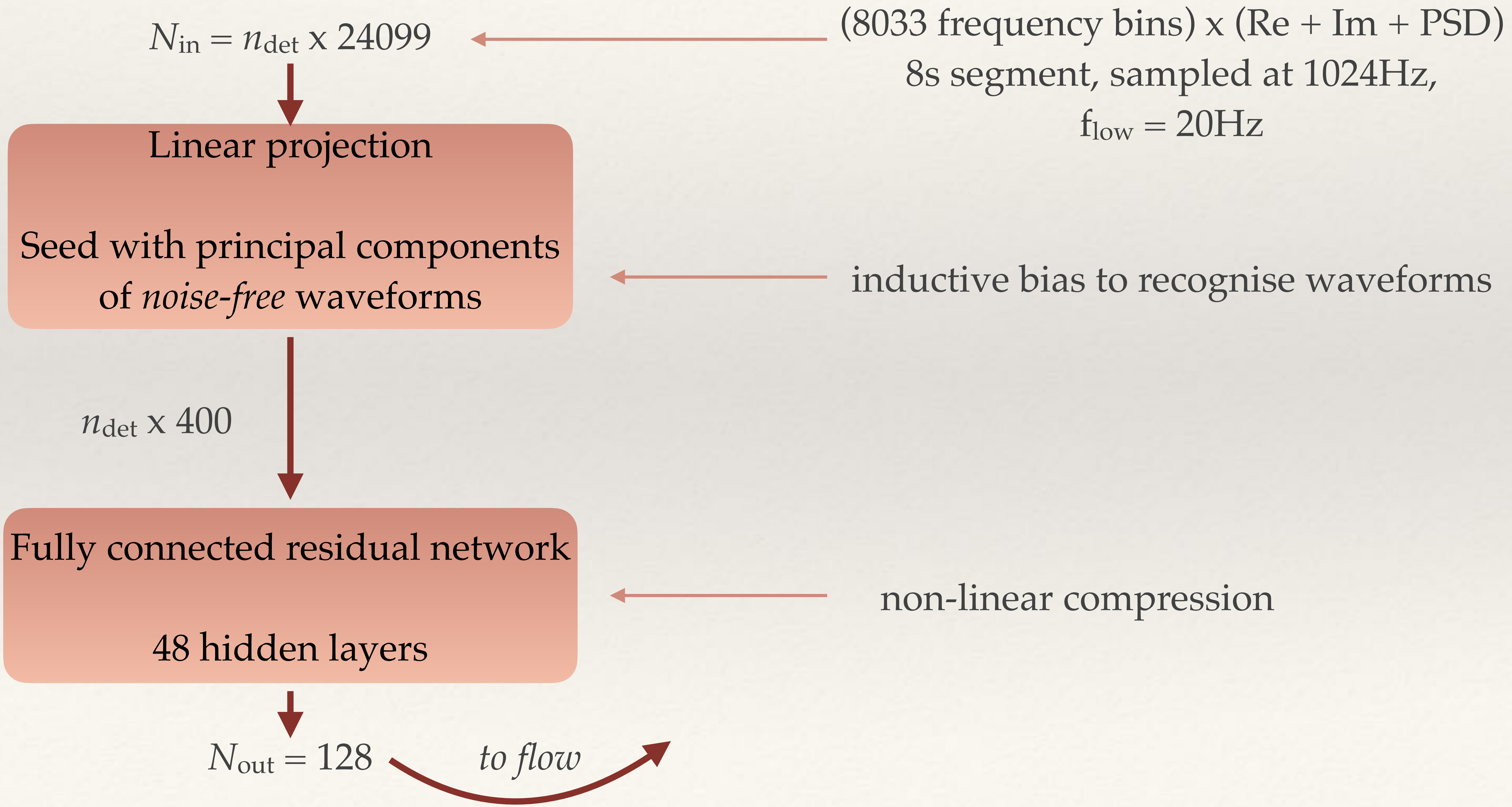
Method validation

- Uncertainty estimates allow for **consistency checks**
  - “within-distribution”
- On individual events, **compare posteriors** against standard tools.
  - “out-of-distribution”



NPE refinements: embedding network

- The existence of reduced bases shows that waveform bases can be compressed. Could impose this by hand, but more robust to learn this using an *embedding network*.



$N_{\mathrm{in}} = n_{\mathrm{det}}$ x 24099

(8033 frequency bins) x (Re + Im + PSD)
8s segment, sampled at 1024Hz,
f_{low} = 20Hz

Linear projection

Seed with principal components of *noise-free* waveforms

inductive bias to recognise waveforms

n_{det} x 400

Fully connected residual network

48 hidden layers

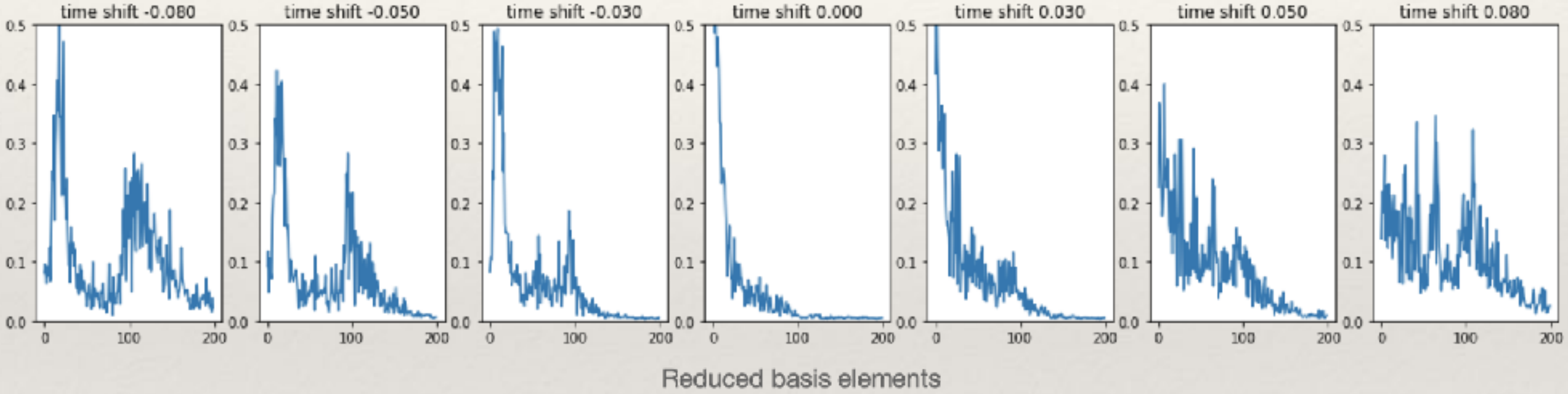
non-linear compression

N_{out} = 128

to flow

NPE refinements: group equivariant NPE

- Representing the time of coalescence, t_I, requires many reduced basis elements. Uses up a lot of training resources and freedom within the network.



- A change in time of coalescence in a single detector corresponds to a (trivial) transformation of the data and template. If the time shift is known, the waveform can be aligned and the learning process significantly simplified.

- Don’t know this *a priori* and not an exact symmetry for a detector network.

NPE refinements: group equivariant NPE

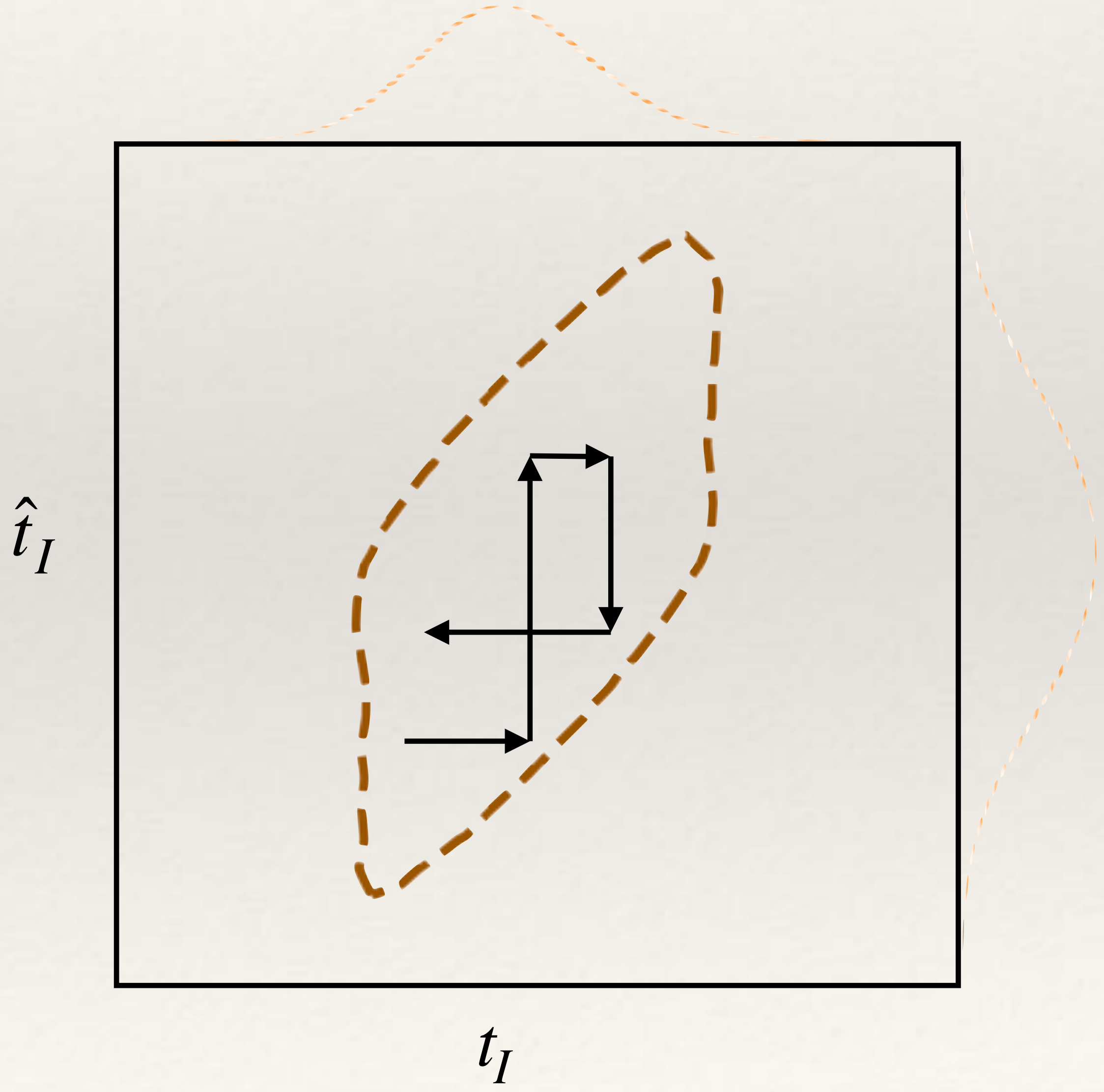
- Introduce a blurred estimate of t_I, $\hat{t}_I$, into the parameter space.
- In training and inference, follow a Gibbs sampling procedure
  1. Align data based on $\hat{t}_I$

     $$\theta \sim q(\theta | T_{-\hat{t}_I}(d), \hat{t}_I)$$

  2. Sample $\hat{t}_I$ from a fixed kernel

     $$\hat{t}_I \sim p(\hat{t}_I | t_I)$$

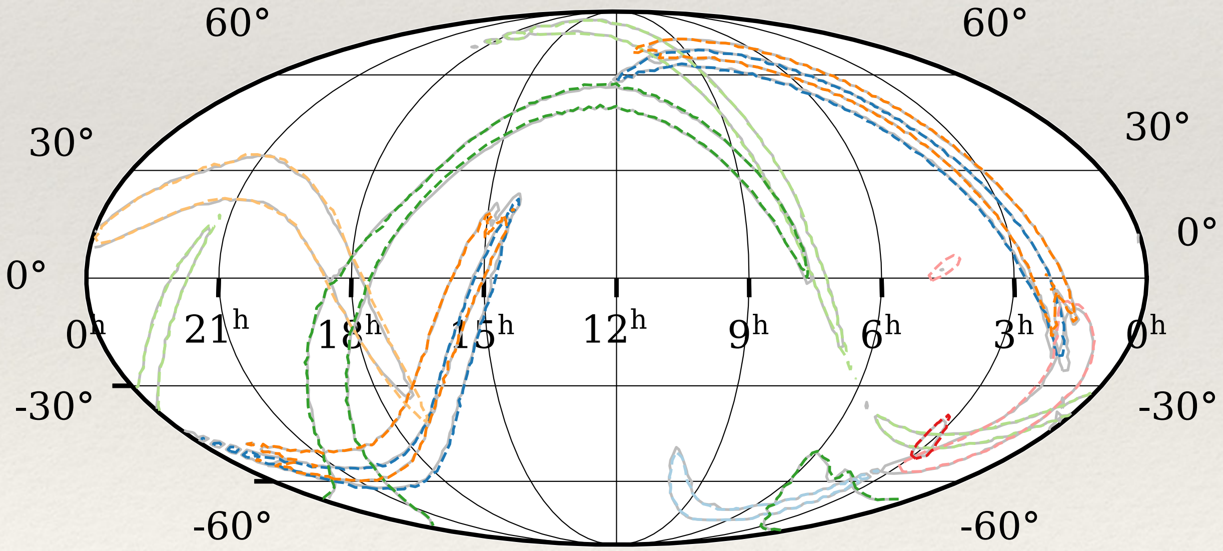
- Converges in O(10) iterations.
- GNPE exploits (near-) symmetries to simplify the learning task.

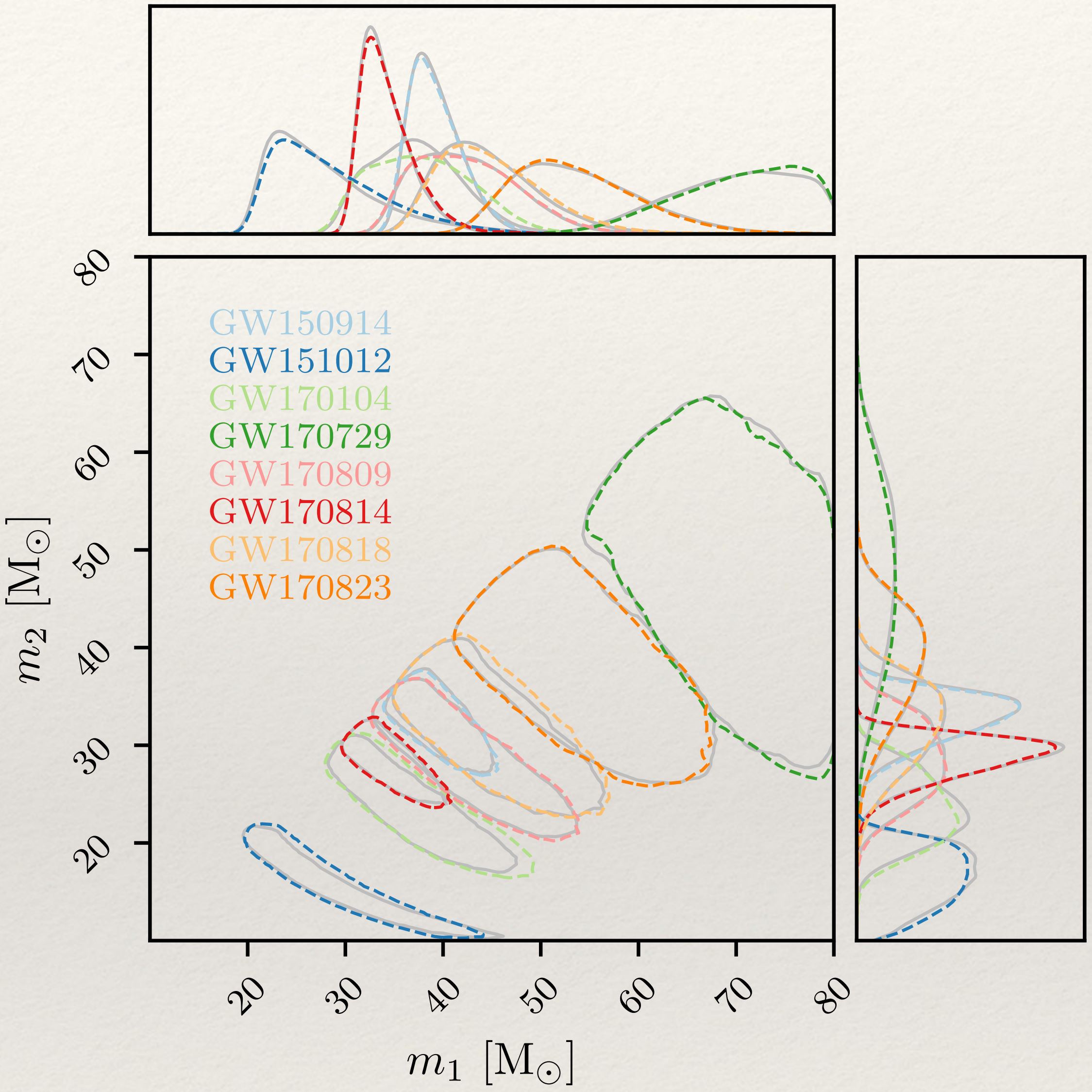


NPE refinements: noise variability

- Account for detector nonstationarity from event to event by conditioning on noise PSD

$$p(\theta \,|\, d, S_\mathrm{n})$$





GWTC-1

NPE refinements: noise variability

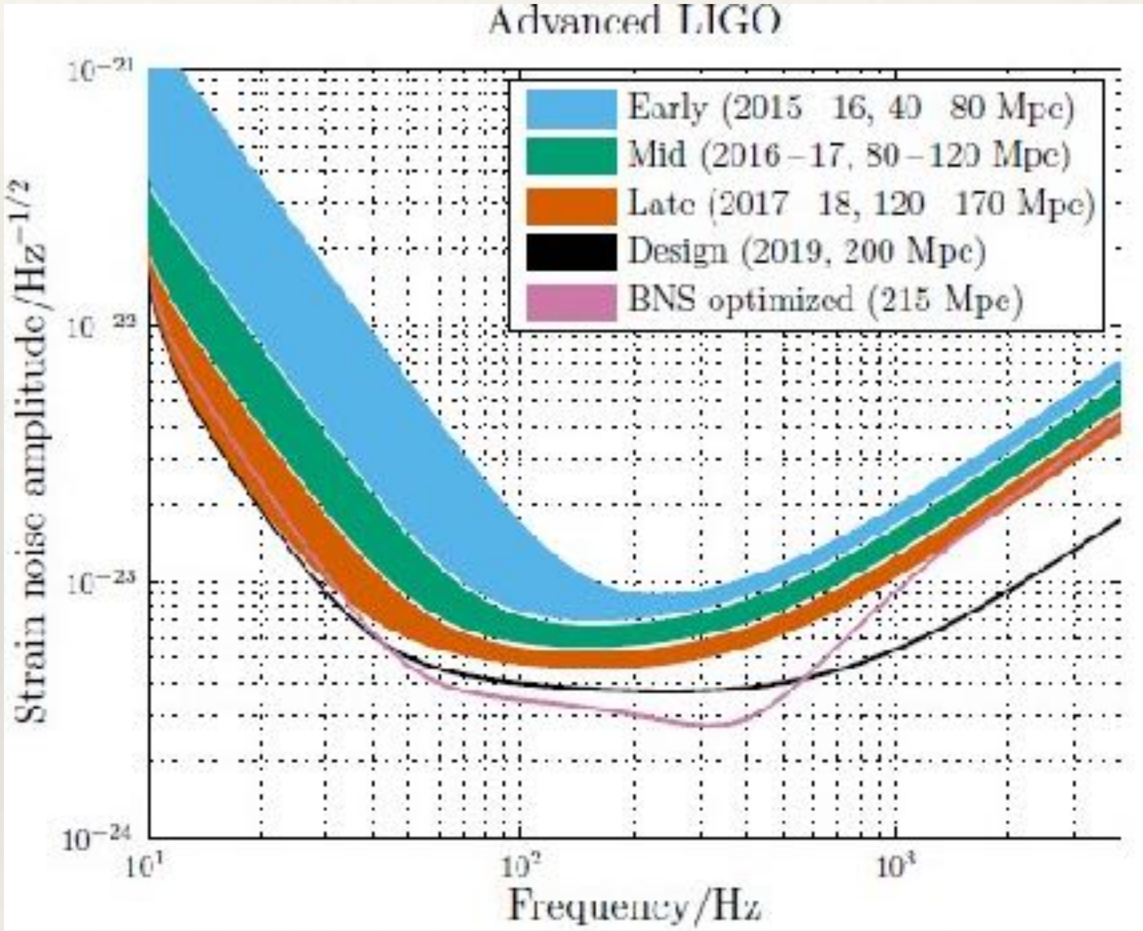
- LIGO noise varies from run to run. Want to be able to use NPE to analyse events right from the start of the run.

- Compare three approaches

  - *Early OR noise*: train network using PSD measured in the first ~week of the OR.

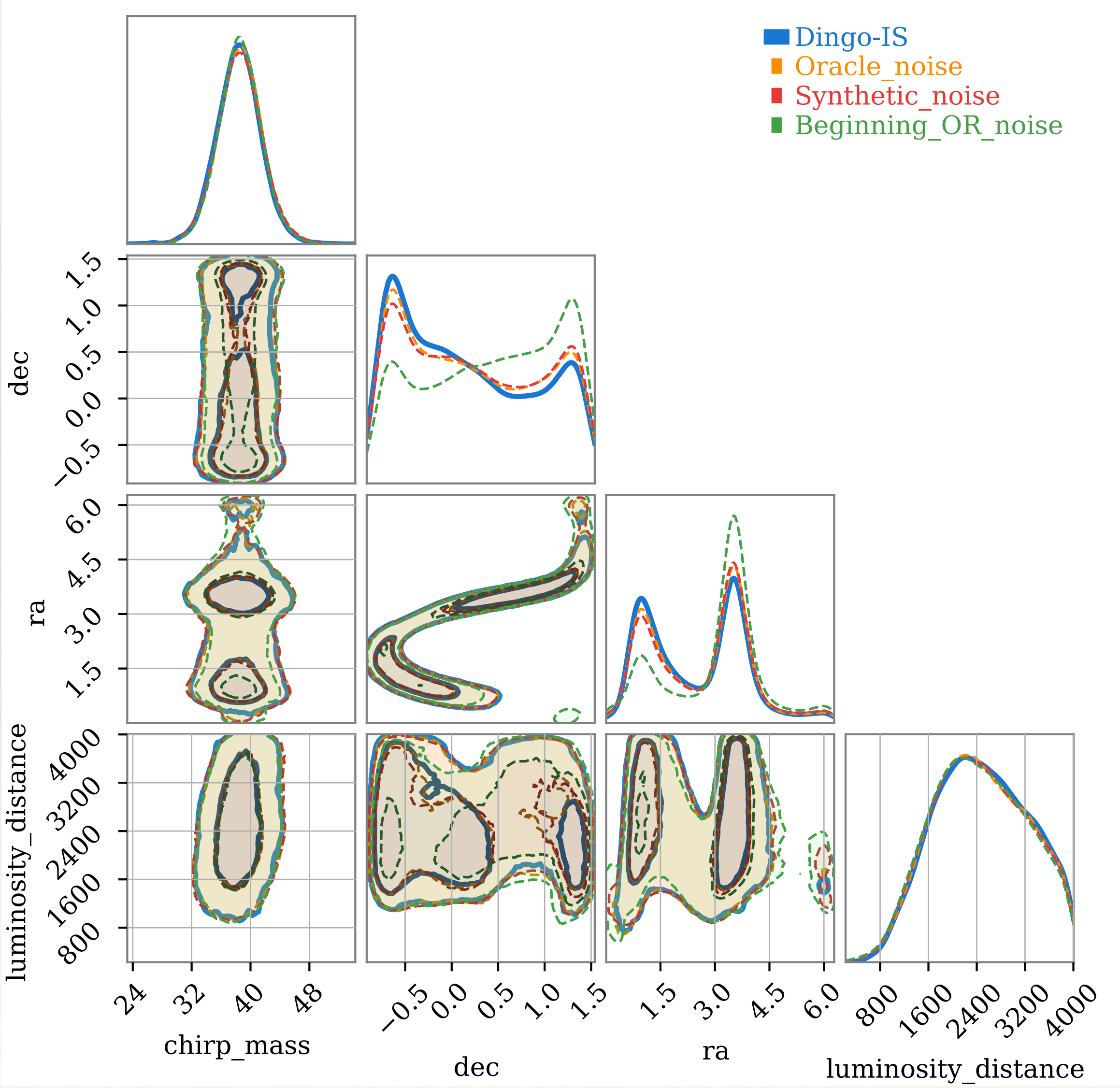
  - *Synthetic noise*: use earlier runs to infer distribution of noise over a run. Use this distribution and early OR noise to simulate the noise distribution for training.

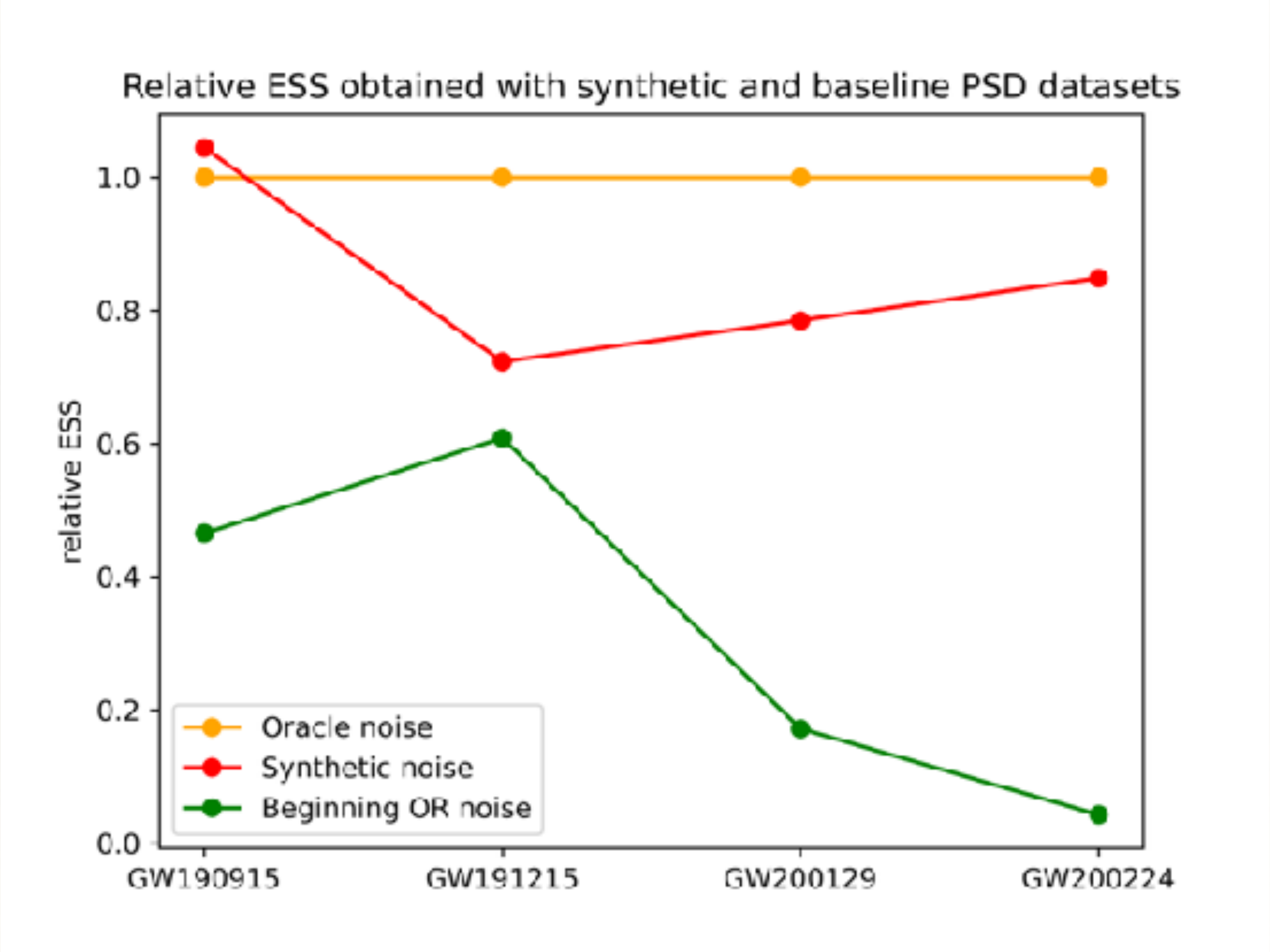
  - *Oracle*: use measured noise over the full run to train network.



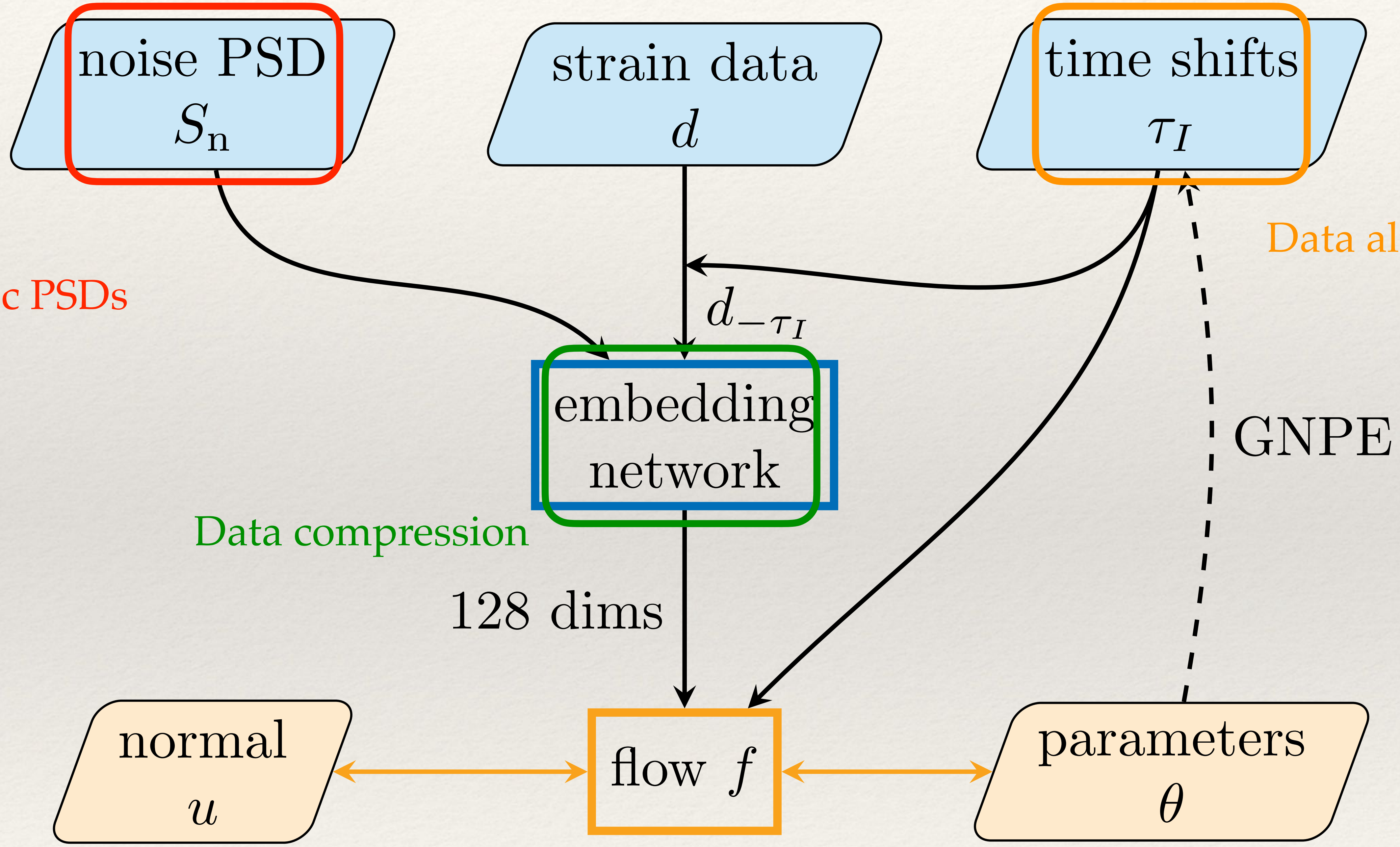
NPE refinements: noise variability

- Performance of synthetic noise much better than early OR, comparable to oracle.





Refinements



Synthetic PSDs

Data alignment

Data compression

Out-of-distribution extrapolation

- Standard methods directly sample the likelihood

$$p(s|\vec{\lambda}) = p(n(t) = s(t) - h(t;\vec{\lambda})) \propto \exp\left[-\frac{1}{2}(s - h(\vec{\lambda})|s - h(\vec{\lambda}))\right]$$

- Out-of-distribution data, e.g., waveform errors have a predictable effect on inference. The projection into the model space leads to a parameter bias

$$\Delta\theta^i_{\rm sys} = (\Gamma^{-1})^{ij}\big(\Delta\mathbf{h}|(\partial_j\mathbf{h}_{\rm m})_{\vec{\theta}_0}\big)$$

- Projections out of the model space have no effect.

Out-of-distribution extrapolation

- Neural networks extrapolate out-of-distribution (OOD) in ways that are difficult to predict.

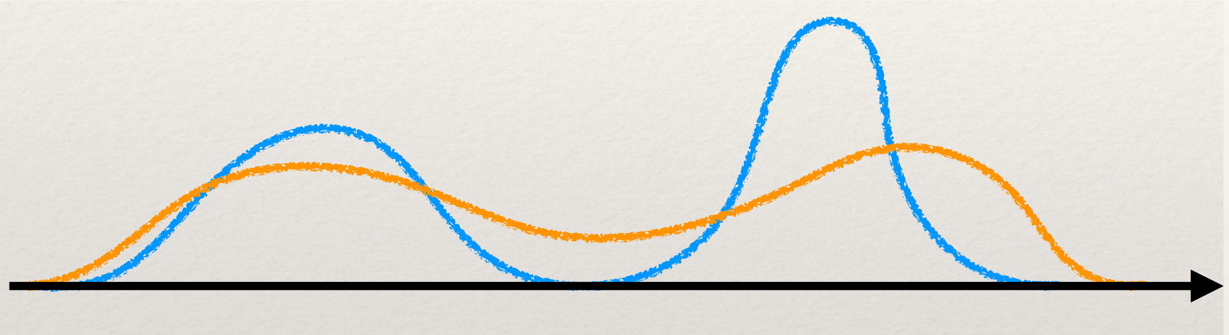
- This means that systematic errors are not directly comparable, but it provides sensitivity to OOD data.

- Reweight samples to target density using *importance sampling*.

- Effective sample size, n_{eff}, provides metric of samples quality.

- Evidence can also be directly computed from the weights.

- **But**: no longer likelihood-free! :-(



target (prior x likelihood)

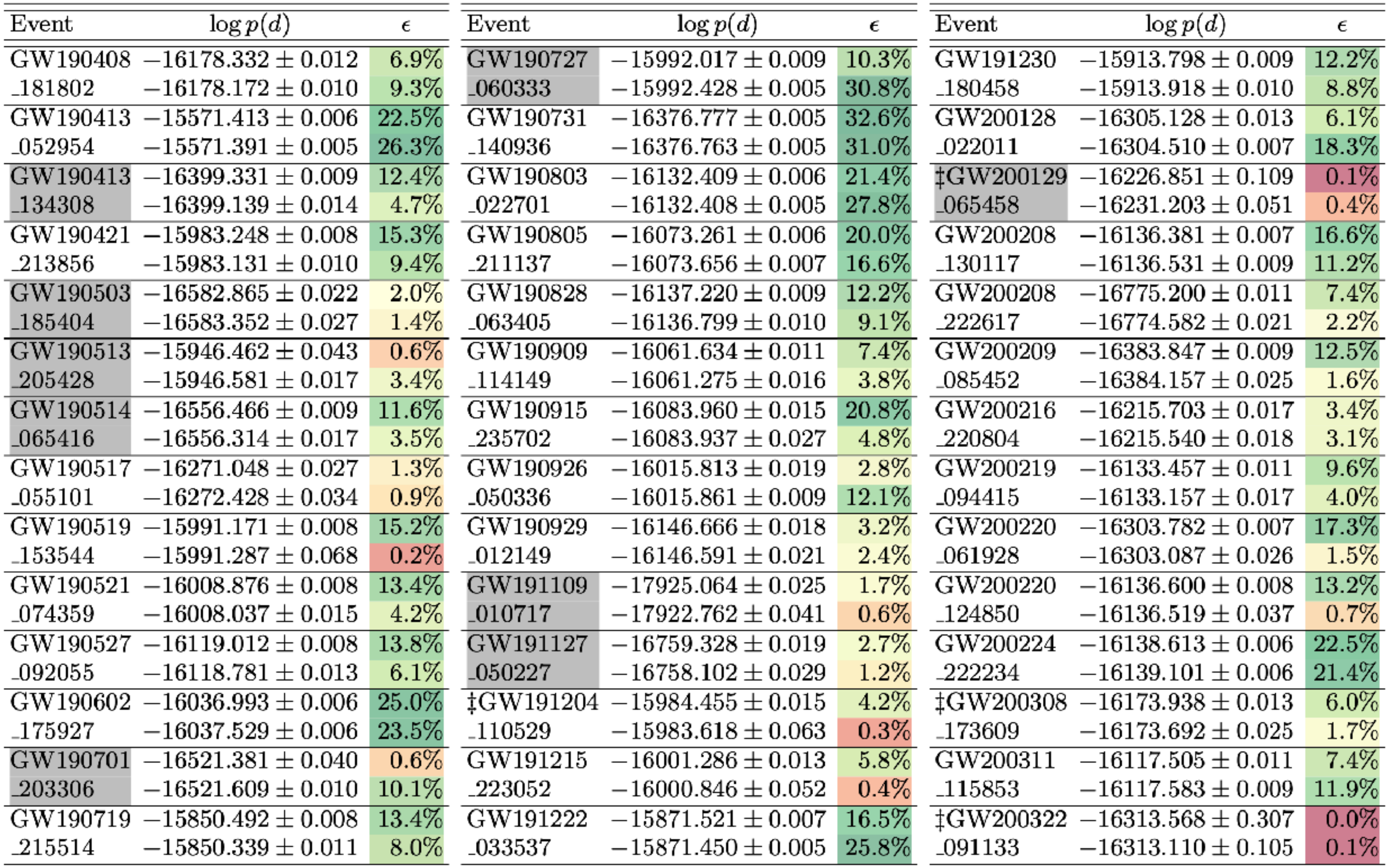
$$w_i \propto \frac{p(\theta_i)p(d\,|\,\theta_i)}{q(\theta_i\,|\,d)}$$

proposal (NPE)

$$n_{\text{eff}} = \frac{\left(\sum_i w_i\right)^2}{\sum_i w_i^2}$$

$$p(d) \approx \frac{1}{n}\sum_{i=1}^{n} w_i$$

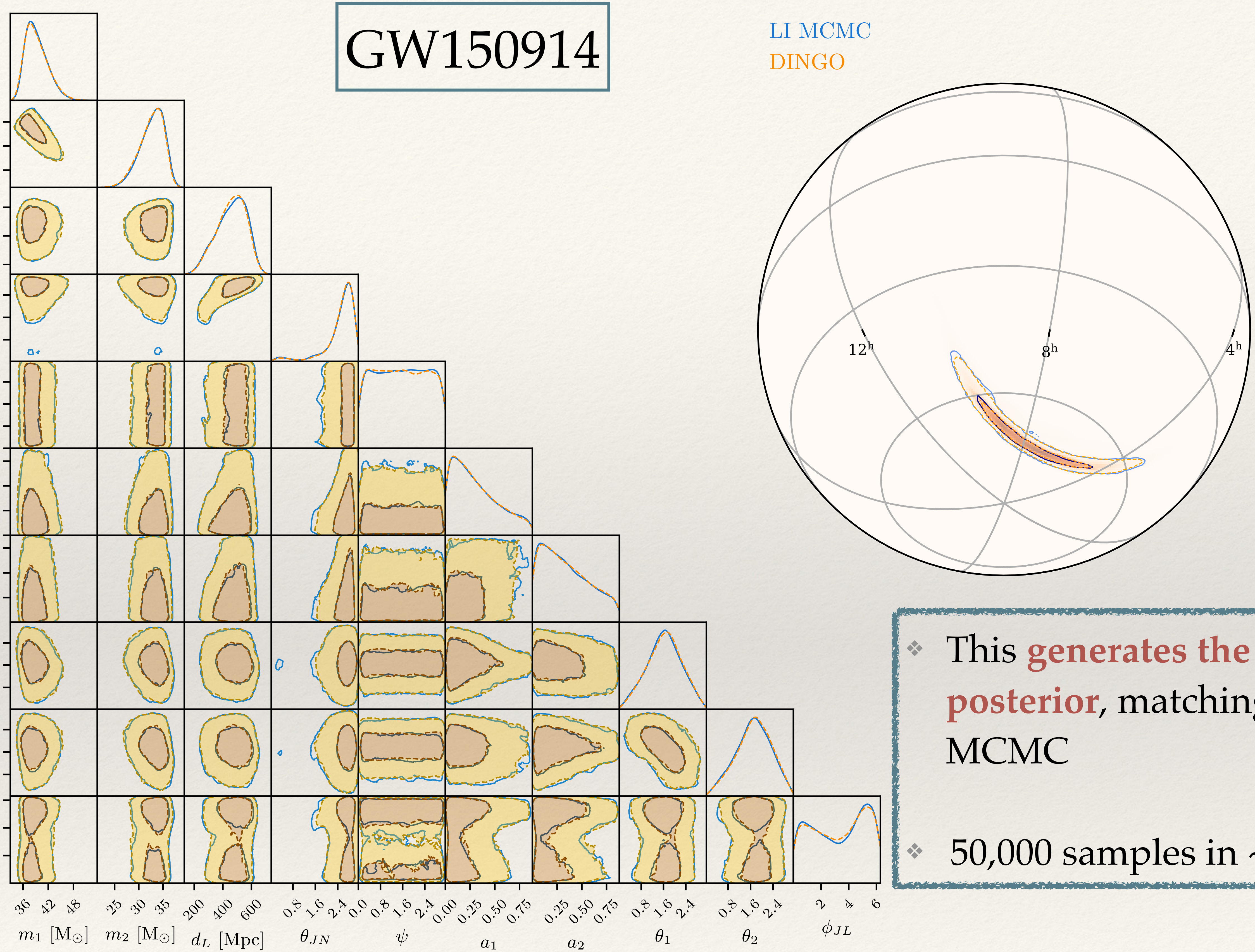
DINGO-IS



Event	$\log p(d)$	ϵ	Event	$\log p(d)$	ϵ	Event	$\log p(d)$	ϵ
GW190408	-16178.332 ± 0.012	6.9%	GW190727	-15992.017 ± 0.009	10.3%	GW191230	-15913.798 ± 0.009	12.2%
_181802	-16178.172 ± 0.010	9.3%	_060333	-15992.428 ± 0.005	30.8%	_180458	-15913.918 ± 0.010	8.8%
GW190413	-15571.413 ± 0.006	22.5%	GW190731	-16376.777 ± 0.005	32.6%	GW200128	-16305.128 ± 0.013	6.1%
_052954	-15571.391 ± 0.005	26.3%	_140936	-16376.763 ± 0.005	31.0%	_022011	-16304.510 ± 0.007	18.3%
GW190413	-16399.331 ± 0.009	12.4%	GW190803	-16132.409 ± 0.006	21.4%	‡GW200129	-16226.851 ± 0.109	0.1%
_134308	-16399.139 ± 0.014	4.7%	_022701	-16132.408 ± 0.005	27.8%	_065458	-16231.203 ± 0.051	0.4%
GW190421	-15983.248 ± 0.008	15.3%	GW190805	-16073.261 ± 0.006	20.0%	GW200208	-16136.381 ± 0.007	16.6%
_213856	-15983.131 ± 0.010	9.4%	_211137	-16073.656 ± 0.007	16.6%	_130117	-16136.531 ± 0.009	11.2%
GW190503	-16582.865 ± 0.022	2.0%	GW190828	-16137.220 ± 0.009	12.2%	GW200208	-16775.200 ± 0.011	7.4%
_185404	-16583.352 ± 0.027	1.4%	_063405	-16136.799 ± 0.010	9.1%	_222617	-16774.582 ± 0.021	2.2%
GW190513	-15946.462 ± 0.043	0.6%	GW190909	-16061.634 ± 0.011	7.4%	GW200209	-16383.847 ± 0.009	12.5%
_205428	-15946.581 ± 0.017	3.4%	_114149	-16061.275 ± 0.016	3.8%	_085452	-16384.157 ± 0.025	1.6%
GW190514	-16556.466 ± 0.009	11.6%	GW190915	-16083.960 ± 0.015	20.8%	GW200216	-16215.703 ± 0.017	3.4%
_065416	-16556.314 ± 0.017	3.5%	_235702	-16083.937 ± 0.027	4.8%	_220804	-16215.540 ± 0.018	3.1%
GW190517	-16271.048 ± 0.027	1.3%	GW190926	-16015.813 ± 0.019	2.8%	GW200219	-16133.457 ± 0.011	9.6%
_055101	-16272.428 ± 0.034	0.9%	_050336	-16015.861 ± 0.009	12.1%	_094415	-16133.157 ± 0.017	4.0%
GW190519	-15991.171 ± 0.008	15.2%	GW190929	-16146.666 ± 0.018	3.2%	GW200220	-16303.782 ± 0.007	17.3%
_153544	-15991.287 ± 0.068	0.2%	_012149	-16146.591 ± 0.021	2.4%	_061928	-16303.087 ± 0.026	1.5%
GW190521	-16008.876 ± 0.008	13.4%	GW191109	-17925.064 ± 0.025	1.7%	GW200220	-16136.600 ± 0.008	13.2%
_074359	-16008.037 ± 0.015	4.2%	_010717	-17922.762 ± 0.041	0.6%	_124850	-16136.519 ± 0.037	0.7%
GW190527	-16119.012 ± 0.008	13.8%	GW191127	-16759.328 ± 0.019	2.7%	GW200224	-16138.613 ± 0.006	22.5%
_092055	-16118.781 ± 0.013	6.1%	_050227	-16758.102 ± 0.029	1.2%	_222234	-16139.101 ± 0.006	21.4%
GW190602	-16036.993 ± 0.006	25.0%	‡GW191204	-15984.455 ± 0.015	4.2%	‡GW200308	-16173.938 ± 0.013	6.0%
_175927	-16037.529 ± 0.006	23.5%	_110529	-15983.618 ± 0.063	0.3%	_173609	-16173.692 ± 0.025	1.7%
GW190701	-16521.381 ± 0.040	0.6%	GW191215	-16001.286 ± 0.013	5.8%	GW200311	-16117.505 ± 0.011	7.4%
_203306	-16521.609 ± 0.010	10.1%	_223052	-16000.846 ± 0.052	0.4%	_115853	-16117.583 ± 0.009	11.9%
GW190719	-15850.492 ± 0.008	13.4%	GW191222	-15871.521 ± 0.007	16.5%	‡GW200322	-16313.568 ± 0.307	0.0%
_215514	-15850.339 ± 0.011	8.0%	_033537	-15871.450 ± 0.005	25.8%	_091133	-16313.110 ± 0.105	0.1%

Table II. 42 BBH events from GWTC-3 analyzed with DINGO-IS. We report the log evidence $\log p(d)$ and the sample efficiency ϵ for the two waveform models IMRPhenomXPHM (upper rows) and SEOBNRv4PHM (lower rows). Highlighting colors indicate the sample efficiency (green: high; yellow: medium; orange/red: low); DINGO-IS results can be trusted for medium and high ϵ (see Supplemental Material). Events in gray suffer from data quality issues [1, 21]. ‡See remarks on these events in text.

GW150914



- This **generates the posterior**, matching MCMC
- 50,000 samples in ~ 20 s

Simulation-based inference

- These machine-learning methods are all examples of **simulation-based inference,** which simply means that training uses simulated data (noisy waveforms).
  - Standard inference methods (e.g., MCMC) are **likelihood-based**.
  - Simulation-based inference is applicable in situations where likelihoods are unavailable or too expensive.
  - Because of this, machine learning can carry out analyses that are not possible using standard tools. E.g., non-Gaussian detector noise.

Summary

- Presented a **sample** of applications of machine learning for gravitational waves:
    - Search, parameter estimation, waveform modeling glitch classification
- Frequent new papers on arxiv
- **Next class:** practical session!