# Machine learning for rapid Bayesian parameter estimation

Computing gravitational wave posteriors in fractions of a second

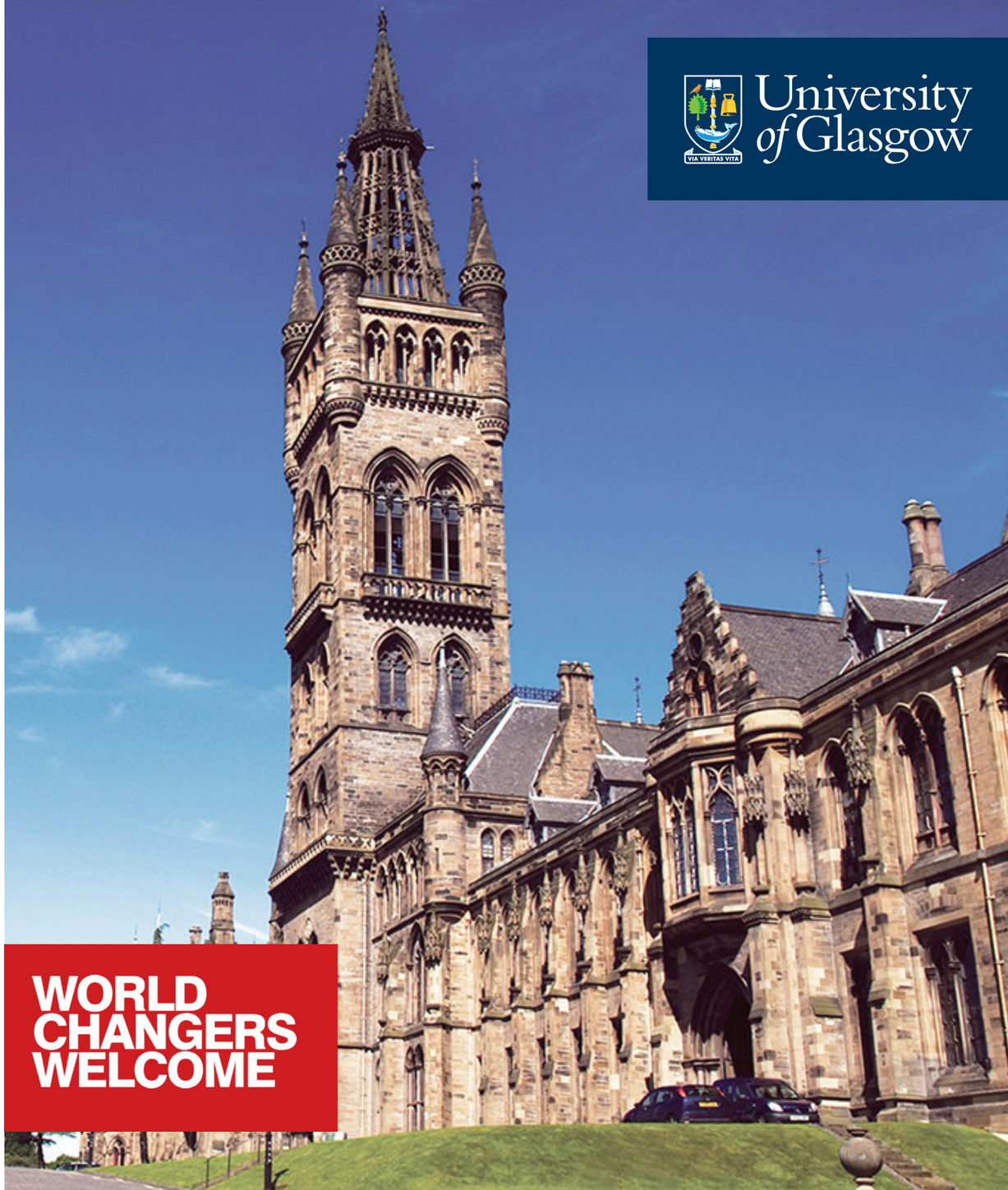
Chris Messenger 25th March 2021



University of Glasgow

# Overview

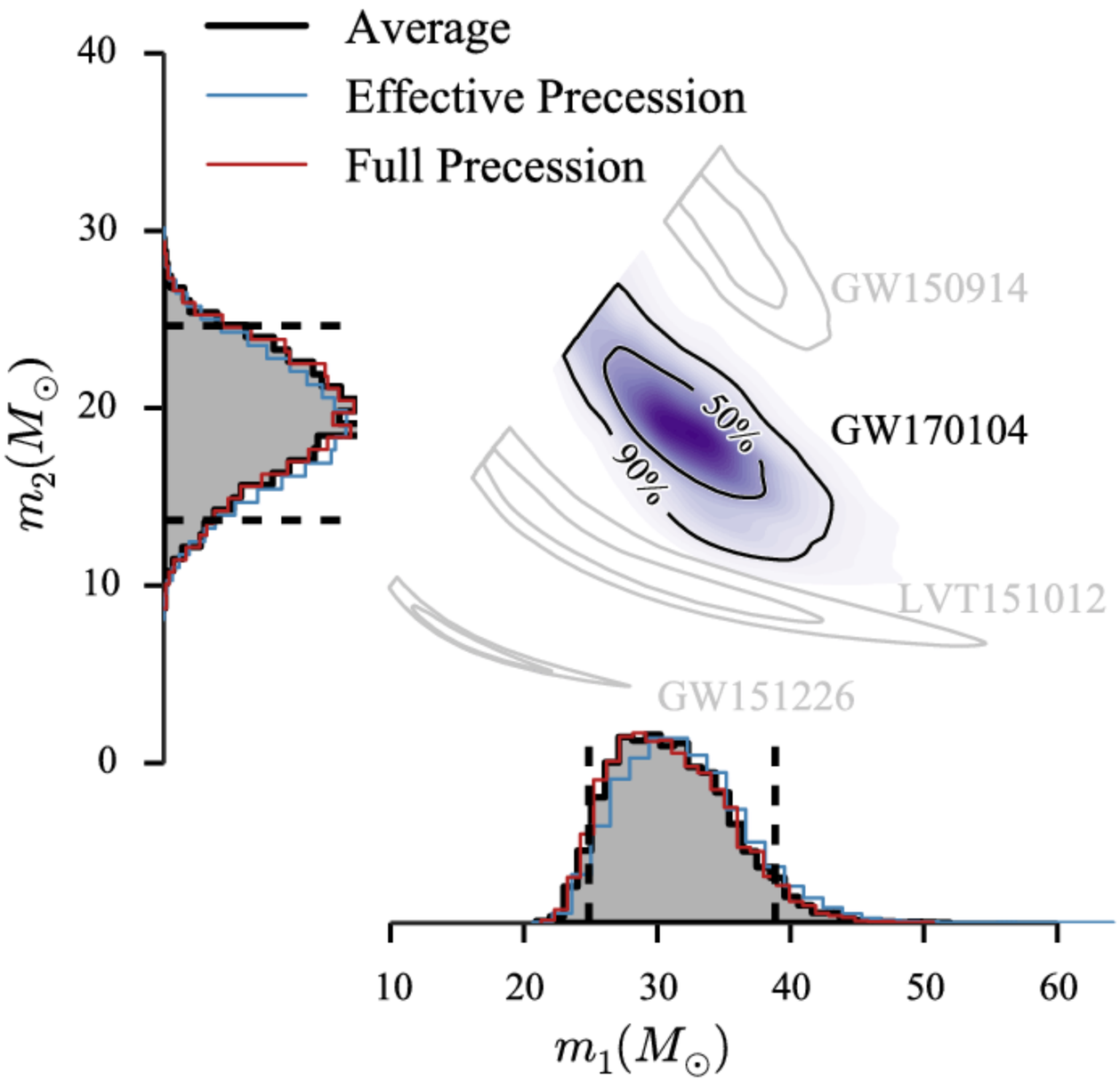
- The problem
- Autoencoders
- CVAE - Vitamin
  - The maths
  - Some results
- Summary



LIGO-Virgo Collaboration, PRL, 118, 22 (2017)

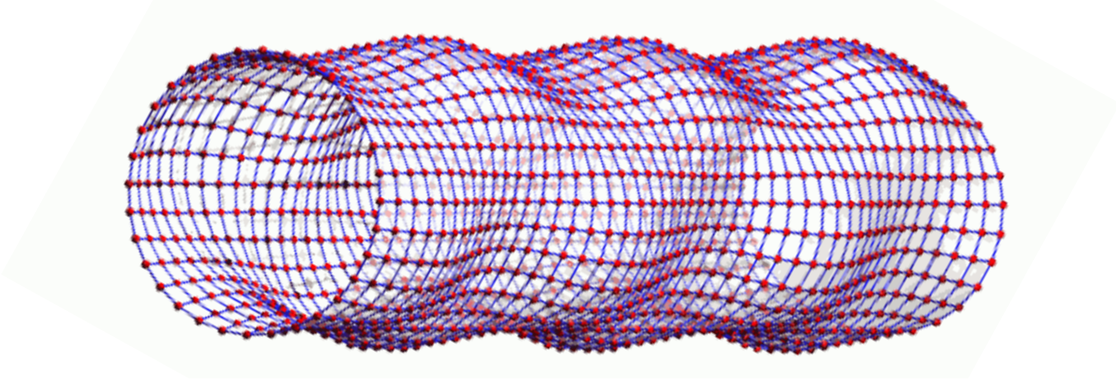
# The problem

GW parameter estimation is slow

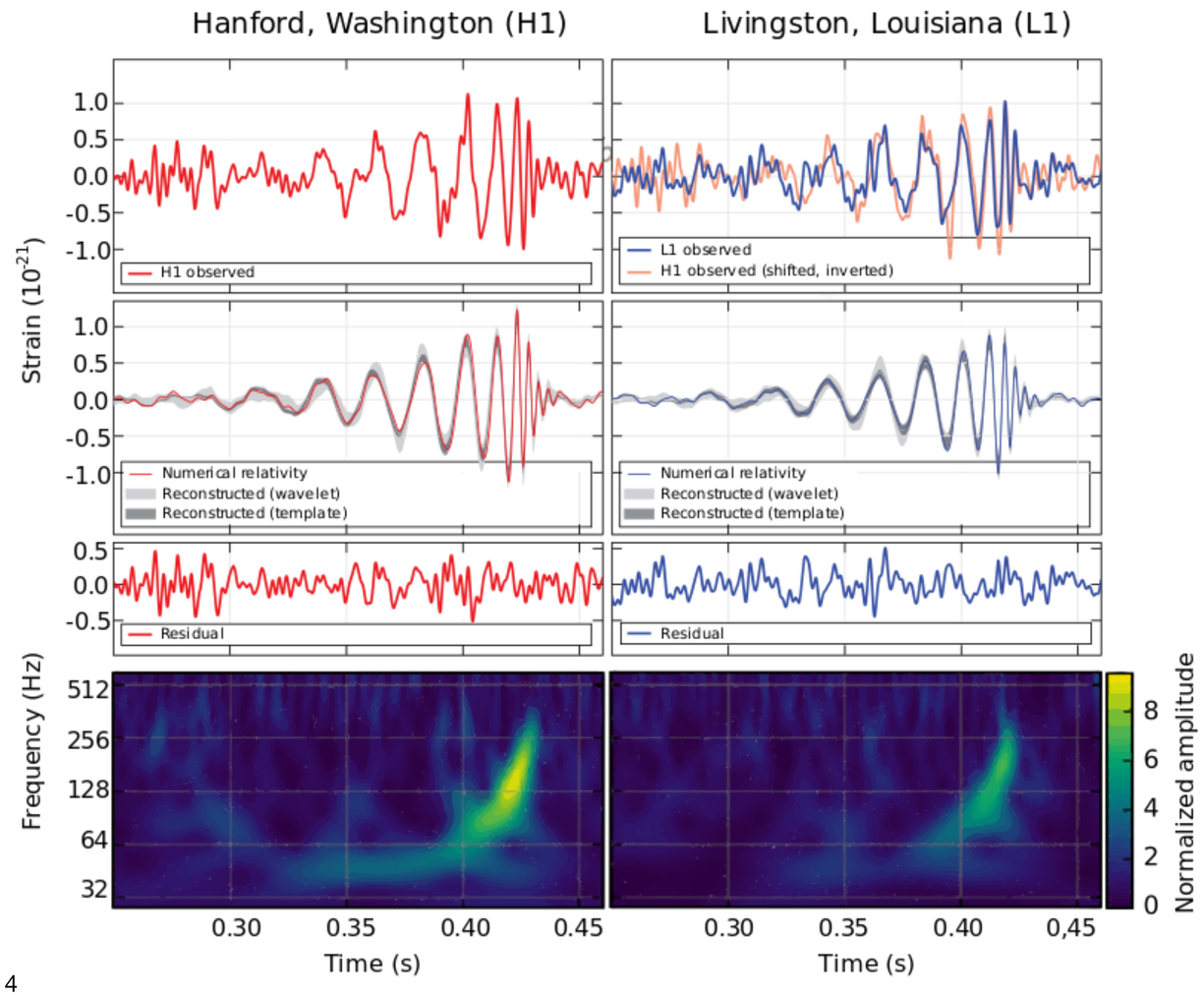


# Very brief GW intro

## What is it that we are interested in

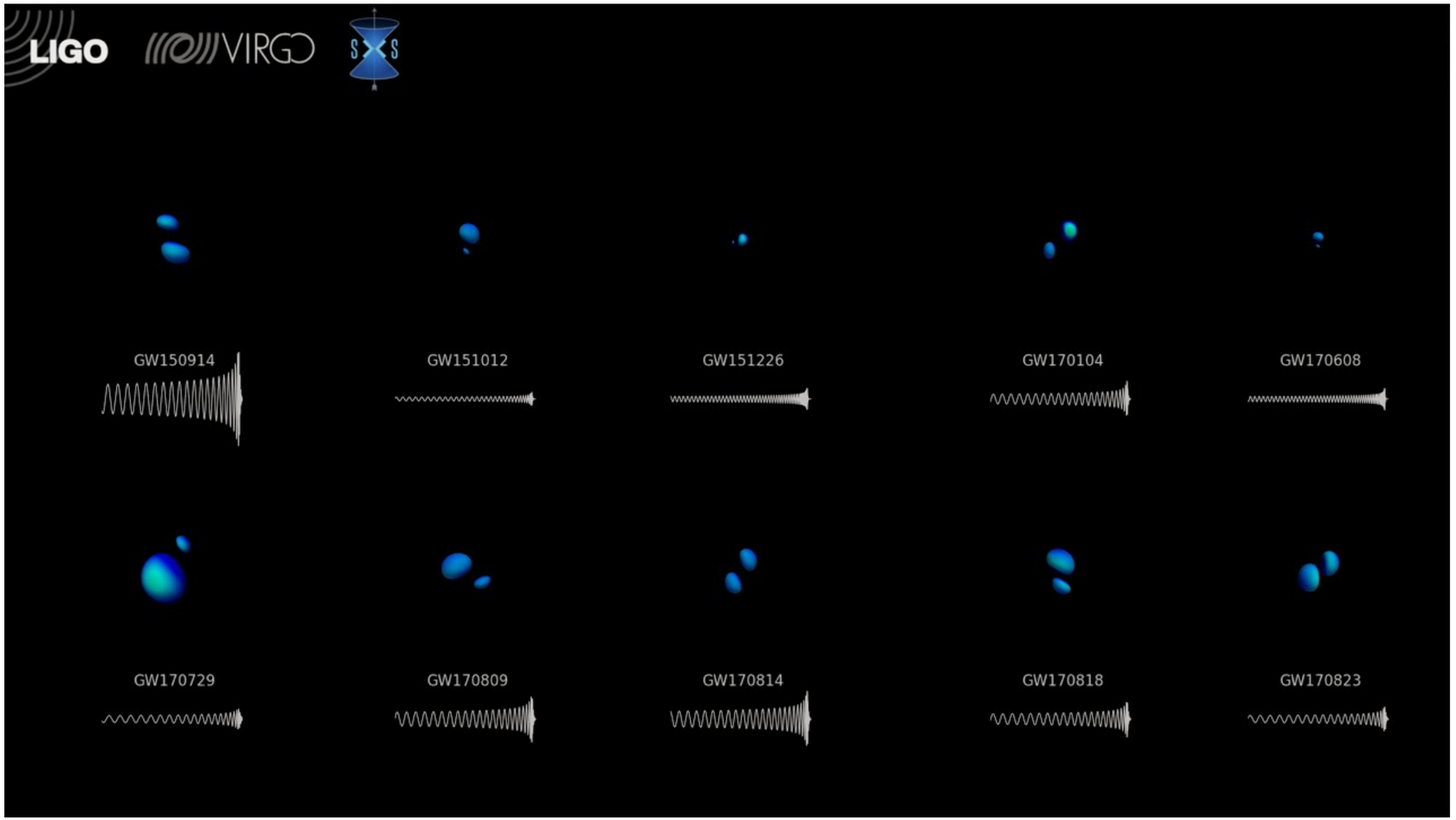


- GWs are ripples in space-time that travel at the speed of light
- They are generated by time varying mass distributions
- They have 2 polarisation states and affect the relative positions of test particles
- We will focus on signals generated from compact binary coalescences




[LIGO-Virgo Collaboration, PRL, 116, 6 (2016)]

# Example detections




https://www.youtube.com/watch?v=gmmD72cFOU4&t=28s

# Current latency

## Optimal but not fast

- Typical analyses (for O3) have taken between 6 hours and 5 days
- This is for full PE and not to be compared with the rapid sky only tools [Singer & Price PRD, 93, 2 (2016)]
- There are other overheads in getting analyses running
- Important for multi-messenger astrophysics and computationally

GraceDB **Public Alerts** Latest Search Documentation Login

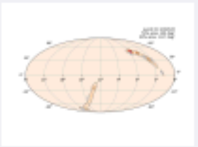
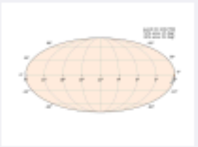
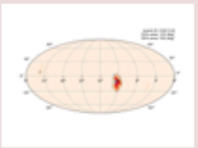
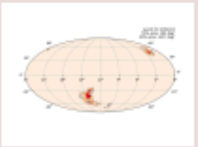
Please log in to view full database contents.

LIGO/Virgo O3 Public Alerts

Detection candidates: 56

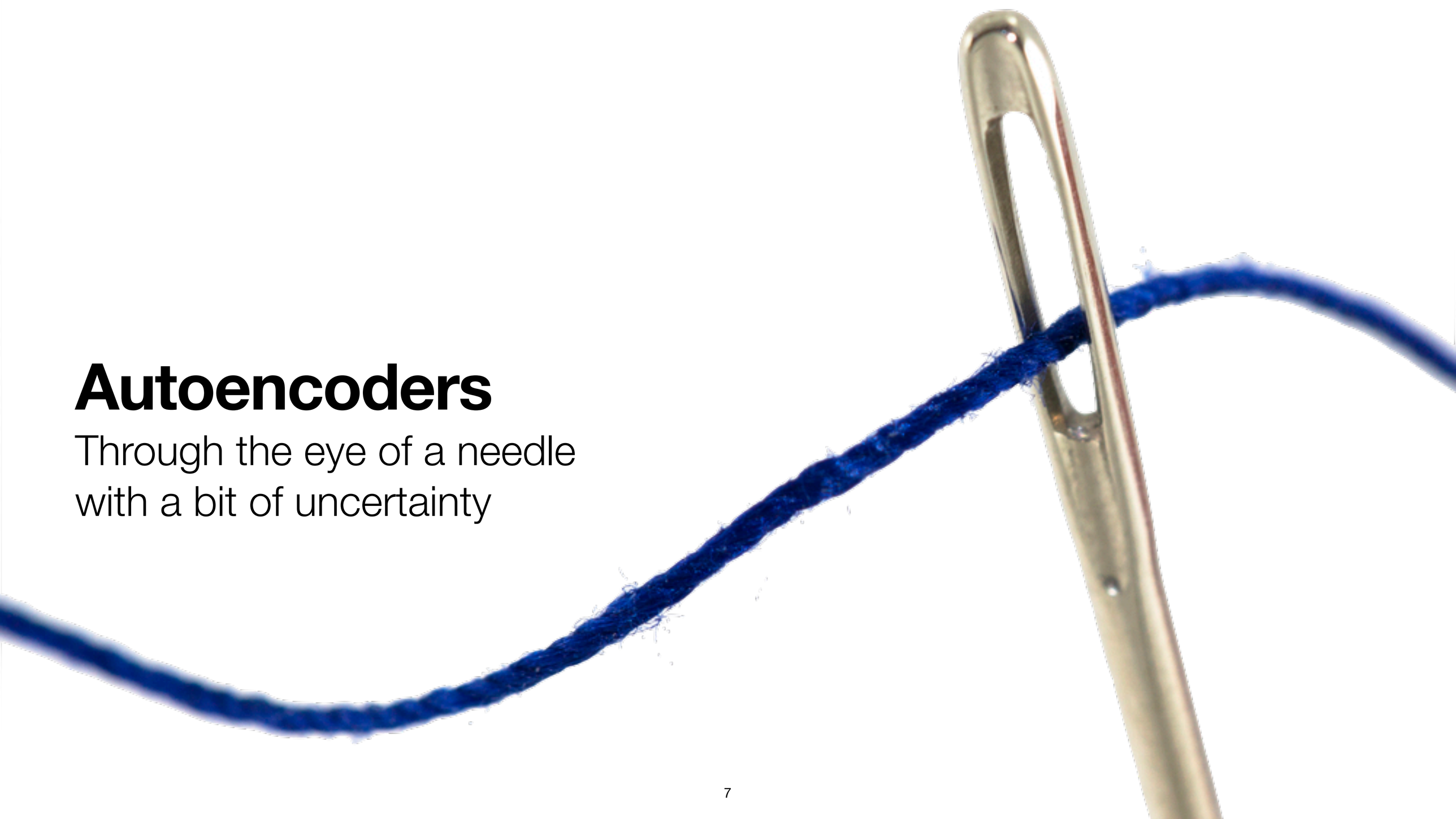
https://gracedb.ligo.org/superevents/public/O3/

SORT: EVENT ID (A-Z) ▼

| Event ID | Possible Source (Probability) | UTC | GCN | Location | FAR |
|---|---|---|---|---|---|
| S200316bj | MassGap (>99%) | March 16, 2020 21:57:56 UTC | GCN Circulars Notices \| VOE |  | 1 per 446.44 years |
| S200311bg | BBH (>99%) | March 11, 2020 11:58:53 UTC | GCN Circulars Notices \| VOE |  | 1 per 3.5448e+17 years |
| S200308e | NSBH (83%), Terrestrial (17%) | March 8, 2020 01:19:27 UTC | GCN Circulars Notices \| VOE |  | 1 per 8.757 years |
| S200303ba | BBH (86%), Terrestrial (14%) | March 3, 2020 12:15:48 UTC | GCN Circulars Notices \| VOE |  | 1 per 2.4086 years |

# Autoencoders

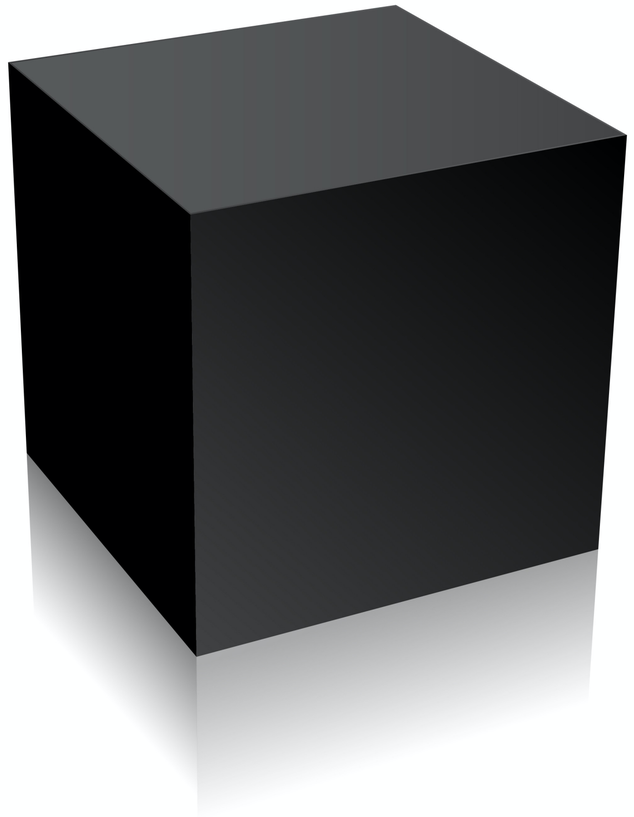
Through the eye of a needle
with a bit of uncertainty



# Machine learning background

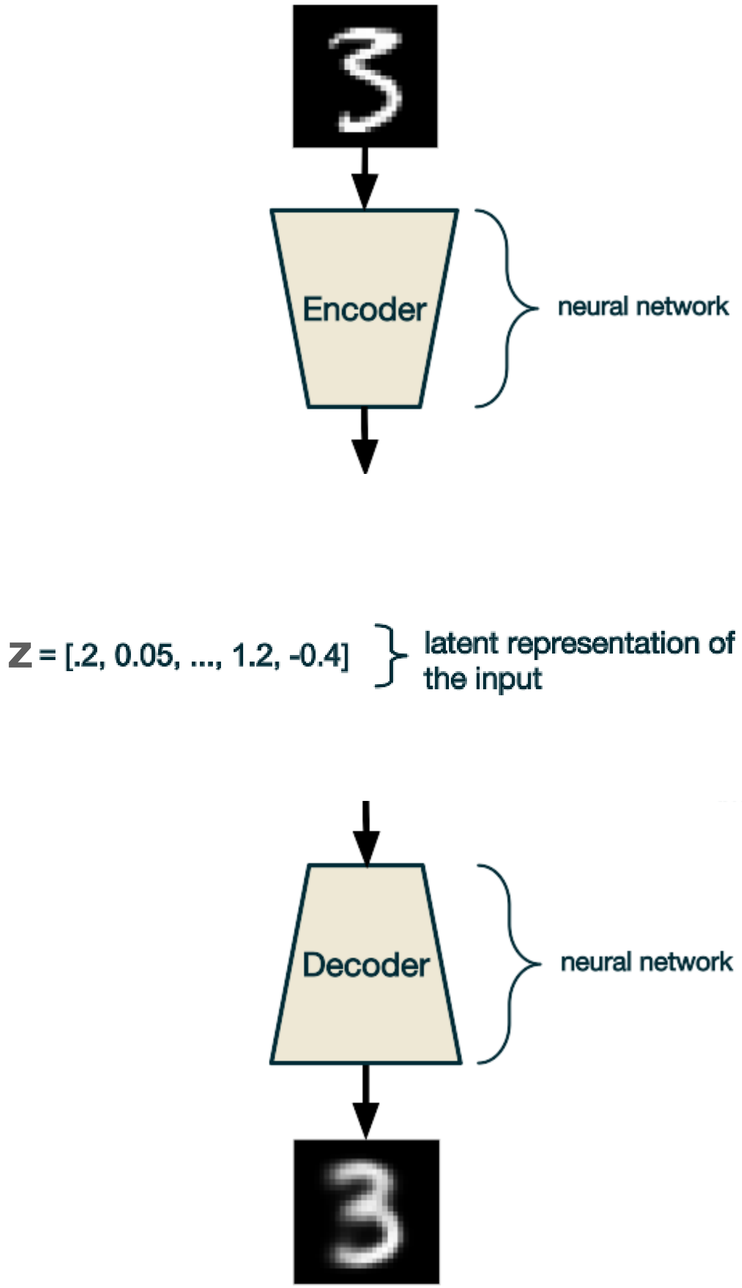
## Assumed knowledge

- I will assume that you know what the following are
  - neuron
  - layer
  - fully connected or convolutional layer
  - activation function
  - etc..
- If lost, just think of a network/layer as a black box with inputs and outputs



# Basic autoencoder

- 2 networks in sequence
- Encoder maps the input into a (reduced) abstract “latent” representation
- The Decoder network converts the latent representation into an output
- The loss function is minimised when the output best matches the input

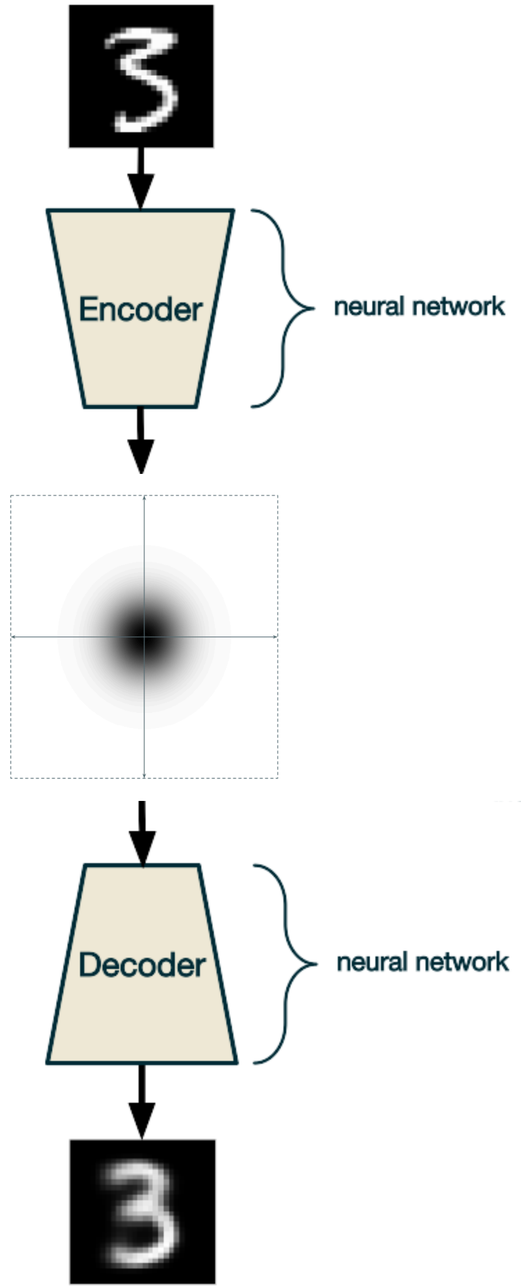



https://ijdykeman.github.io/ml/2016/12/21/cvae.html

# Variational autoencoder

Same as autoencoder, but…

- Encoder predicts the mean and covariance of a multi-dimensional Gaussian in the latent space
- We then randomly sample from that distribution
- The Decoder network converts the (random) latent representation into an output
- The loss function is minimised when the output best matches the input, and…
- there's an extra loss component that keeps the latent space Gaussian averaged over all inputs

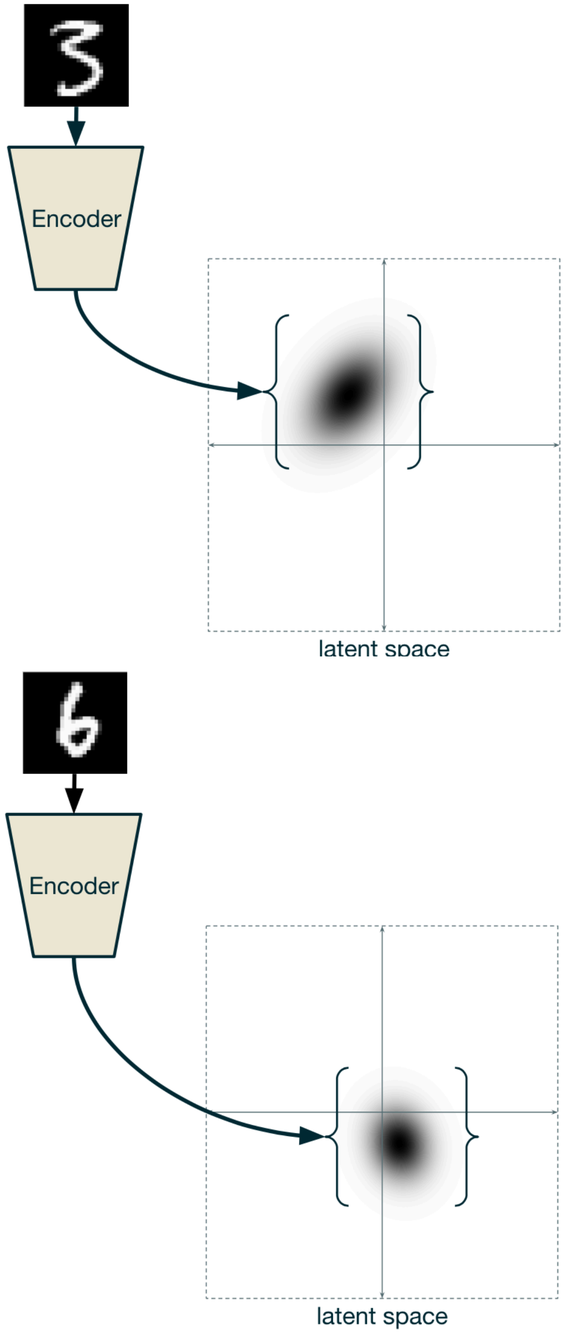



https://ijdykeman.github.io/ml/2016/12/21/cvae.html

# Variational autoencoder

Same as autoencoder, but…

- So, here an image of a “3” gets mapped to a particular part of the latent space.
- The inherent spread in latent space represents the acceptable variation in that ”3”.
- A “6” lives elsewhere in the latent space, probably close to the “8”s, and “5”s since they share similar characteristics.

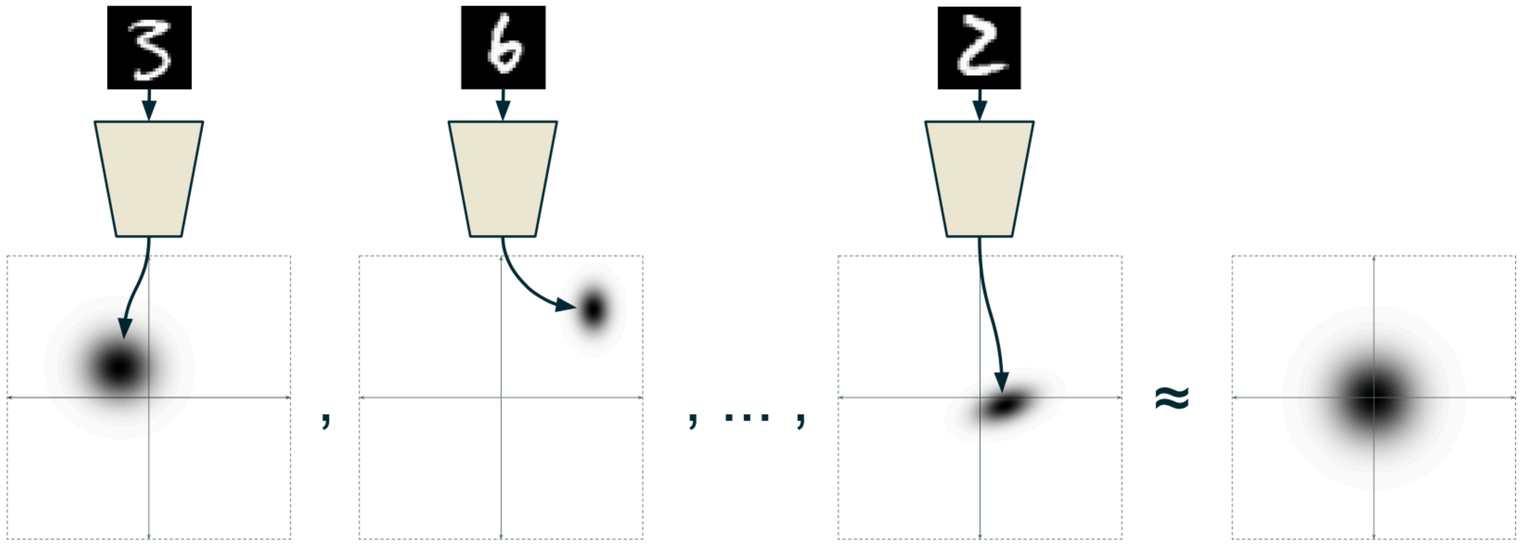


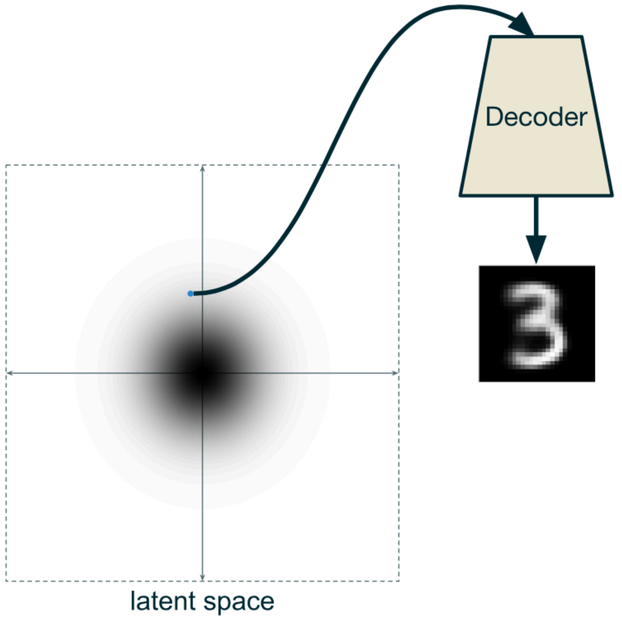

https://ijdykeman.github.io/ml/2016/12/21/cvae.html

# Variational autoencoder

Same as autoencoder, but…

- The KL loss keeps the ensemble of training data mapped to a zero-mean, unit-variance Gaussian.
- So you can then sample from it after training to generate new images



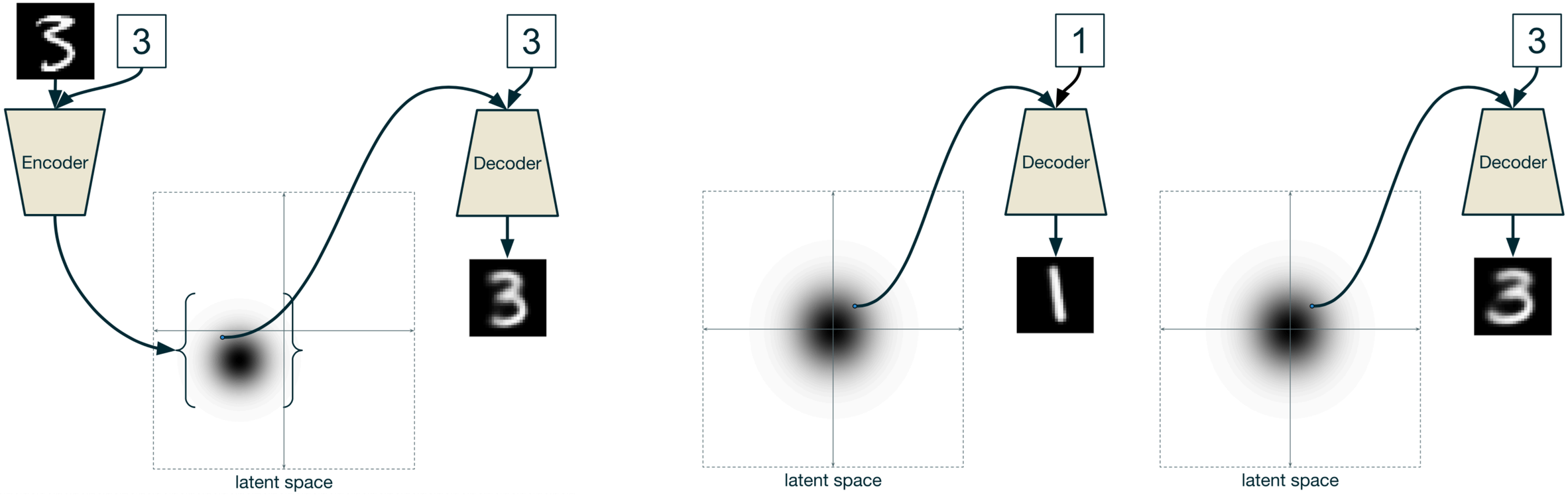




https://ijdykeman.github.io/ml/2016/12/21/cvae.html

# Conditional Variational Autoencoder (CVAE)

## Getting what you asked for

- Passing labels allows you specify properties of the output




https://ijdykeman.github.io/ml/2016/12/21/cvae.html

# Conditional Variational Autoencoder (CVAE)

Getting what you asked for

- You should think of the encoder network in terms of probability distributions.
- For this basic CVAE the encoder is modelling the distribution

$$p(z|x,y)$$

- The decoder is modelling the function

$$f(y,z)$$

x is the image
y is the label
z is the latent space location

- and the loss function is (something like)

$$L = \langle (f(z,y) - x)^2 \rangle_{p(x,y,z)} + \mathrm{KL}\left(p(z|x,y)|G(0,1)\right)$$

https://ijdykeman.github.io/ml/2016/12/21/cvae.html

# CVAE - Vitamin

Variational Inference …

… tamin?

https://hagabbar.github.io

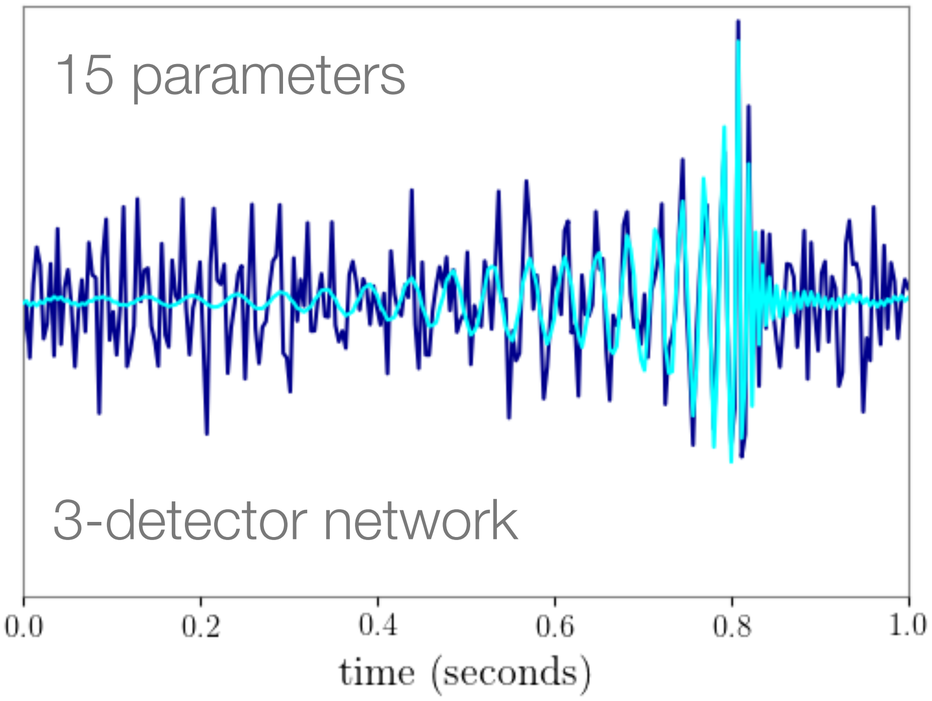
# Defining the data

## What are the quantities of interest

- The data we measure is a noisy timeseries (y) consisting of a deterministic signal plus noise
- The signal is defined by the parameters (x)
- We want to obtain the posterior on the signal parameters

$$p(x|y)$$

| Parameter name | symbol | min | max |
|---|---|---|---|
| mass 1 | $m_1$ | 35 | 80 |
| mass 2 | $m_2$[a] | 35 | 80 |
| luminosity distance | $d_{\rm L}$ | 1 | 3 |
| time of coalescence | $t_0$ | 0.65 | 0.85 |
| phase at coalescence | $\phi_0$ | 0 | $2\pi$ |
| right ascension | $\alpha$ | 0 | $2\pi$ |
| declination | $\delta$ | $-\pi/2$ | $\pi/2$ |
| inclination | $\iota$ | 0 | $\pi$ |
| polarisation | $\psi$ | 0 | $\pi$ |
| spins | - | | 0 |



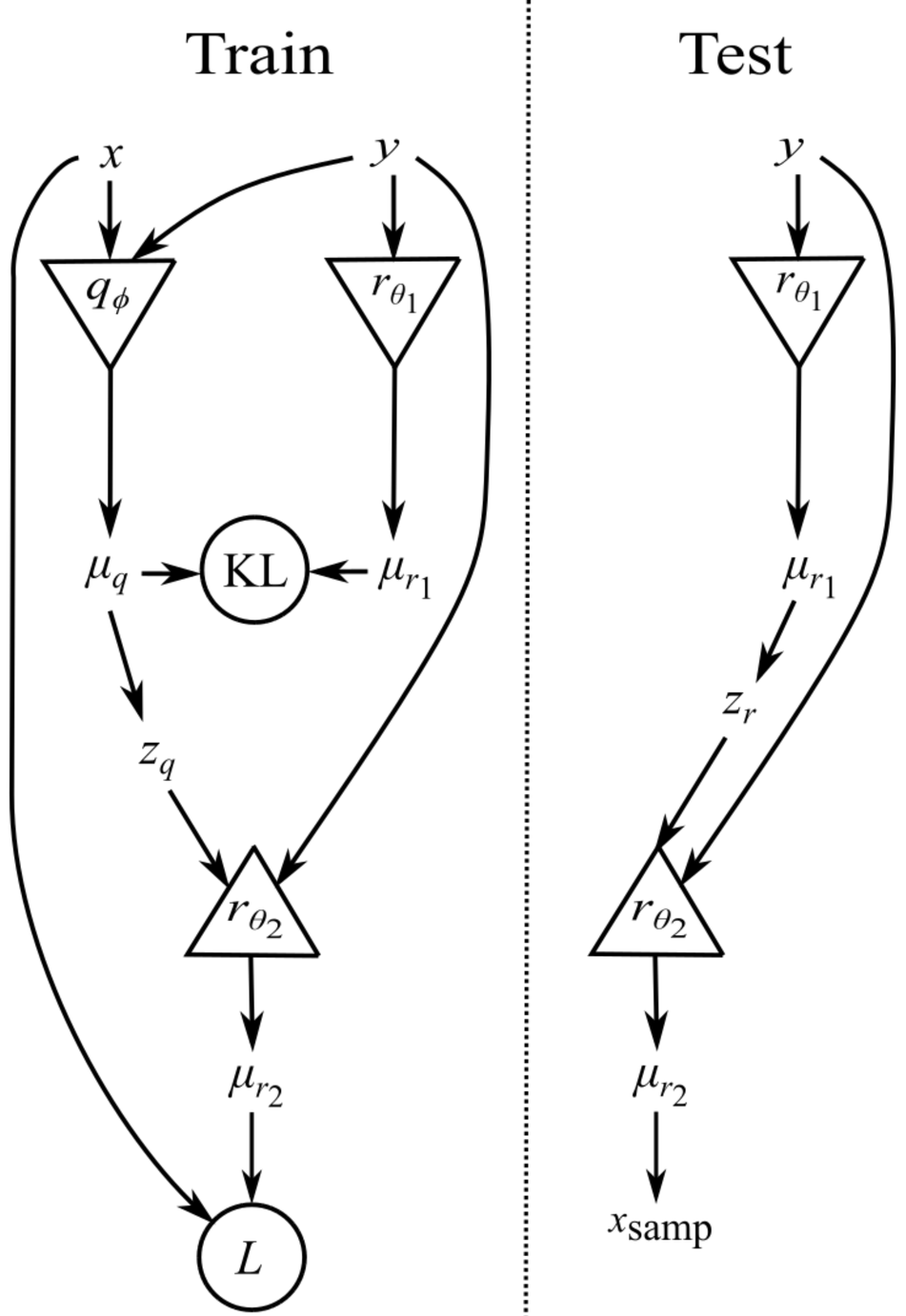

Gabbard et al, arXiv 1909.06296 (2019)

# CVAE mathematics

How the networks model probability distributions

- My aim in the next few slides is to show you how this diagram is completely motivated mathematically
- We start with the definition of the cross entropy between the true posterior and the approximation

$$H(y) = \int dx\, p(x|y) \log r(x|y).$$




[Gabbard et al, arXiv 1909.06296 (2019)]

# CVAE mathematics

How the networks model probability distributions

- Generalising the loss by taking the expectation over all possible values of y

$$
\begin{aligned}
H &= \left\langle \int dx\, p(x|y) \log r(x|y) \right\rangle_{y \sim p(y|x)}, \\
&= \int dx \int dy\, p(x|y) p(y) \log r(x|y), \\
&= \int dx \int dy\, p(x) p(y|x) \log r(x|y),
\end{aligned}
$$

- Introducing our latent variable and approximating the posterior allows us to have an expressive approximation

$$
r(x|y) = \int dz\, r(z|y) r(x|z,y).
$$

Bayes Theorem : $p(a|b)p(b) = p(b|a)p(a)$

# CVAE mathematics

## How the networks model probability distributions

- Introduce the recognition function, q, and derive the Evidence Lower Bound

$$\mathrm{KL}\left[q(z|x,y)||r(z|x,y)\right] = \int dz\, q(z|x,y) \log\left(\frac{q(z|x,y)}{r(z|x,y)}\right)$$

$$\begin{aligned}\mathrm{KL}\left[q(z|x,y)||r(z|x,y)\right] &= \int dz\, q(z|x,y) \log\left(\frac{q(z|x,y)r(x|y)}{r(x|z,y)r(z|y)}\right), \\ &= \int dz\, q(z|x,y) \log r(x|y) - \int dz\, q(z|x,y) \log\left(\frac{r(x|z,y)r(z|y)}{q(z|x,y)}\right), \\ &= \log r(x|y) - \int dz\, q(z|x,y) \log\left(\frac{r(x|z,y)r(z|y)}{q(z|x,y)}\right).\end{aligned}$$

$$\log r(x|y) = \mathrm{KL}\left[q(z|x,y)||r(z|x,y)\right] + \mathrm{ELBO},$$

$$\log r(x|y) \geq \mathrm{ELBO}$$

# CVAE mathematics

How the networks model probability distributions

- Back to the cross-entropy $H = \int dx \int dy\, p(x)p(y|x) \log r(x|y),$

$$
\begin{aligned}
H &\leq \int dx \int dy\, p(x)p(y|x)\mathrm{ELBO}, \\
&\leq \int dx \int dy\, p(x)p(y|x) \int dz\, q(z|x,y) \log \left( \frac{r(x|z,y)r(z|y)}{q(z|x,y)} \right), \\
&\leq \int dx \int dy\, p(x)p(y|x) \left( \int dz\, q(z|x,y) \log \left( \frac{r(z|y)}{q(z|x,y)} \right) + \int dz\, q(z|x,y) \log r(x|y,z) \right), \\
&\leq \int dx \int dy \int dz\, p(x)p(y|x)q(z|x,y) \left( \log \left( \frac{r(z|y)}{q(z|x,y)} \right) + \log r(x|y,z) \right)
\end{aligned}
$$

# CVAE mathematics

How the networks model probability distributions

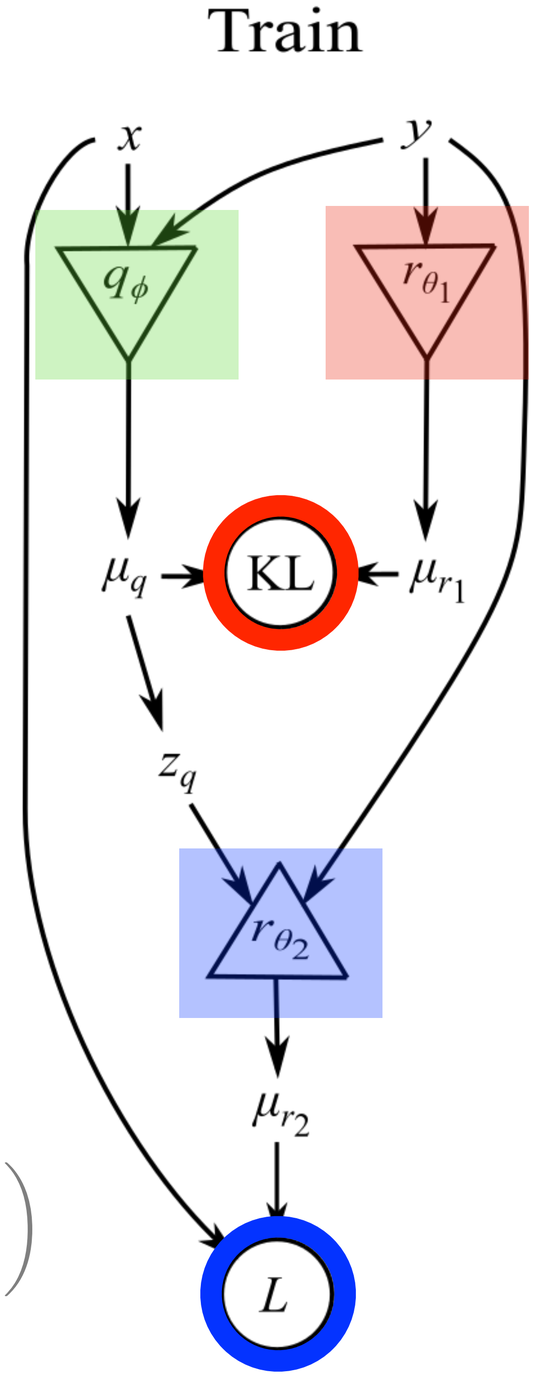
- Now we can see how the diagram is motivated

$$H \leq -\frac{1}{N}\sum_j \left(\log\left(\frac{r(z_j|y_j)}{q(z_j|x_j,y_j)}\right) + \log r(x_j|y_j,z_j)\right)\Bigg|_{\substack{x\sim p(x)\\ y\sim p(y|x)\\ z\sim q(z|x,y)}}$$

KL

$L$

$$\leq \int dx \int dy \int dz\, p(x)p(y|x)q(z|x,y)\left(\log\left(\frac{r(z|y)}{q(z|x,y)}\right) + \log r(x|y,z)\right)$$



# CVAE mathematics

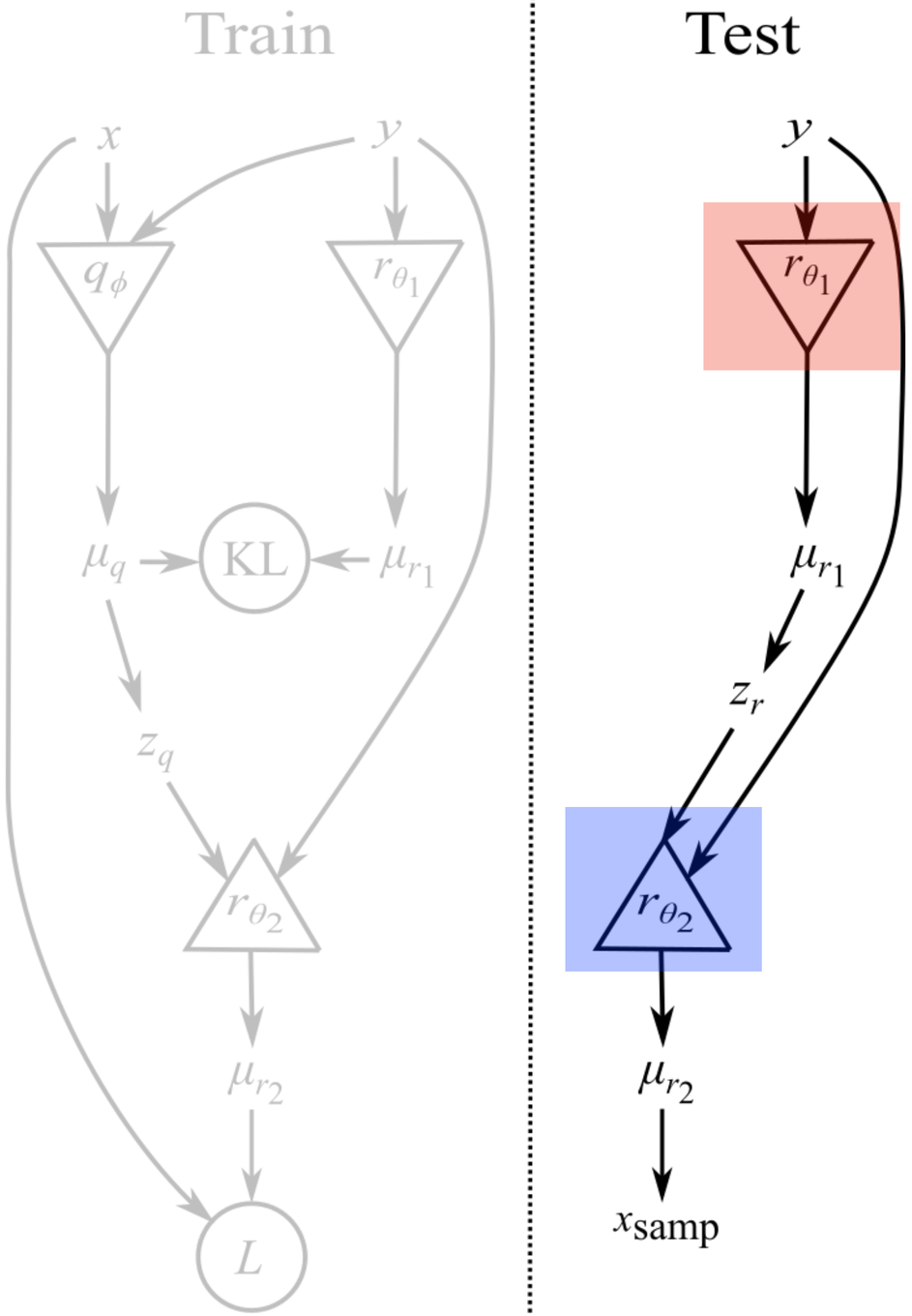
How the networks model probability distributions

- Using the network to generate samples from the posterior

$$r(x|y) = \int dz\, r(z|y) r(x|z,y).$$

- So x is drawn from

$$x \sim r(x|y,z)|_{z\sim r(z|y)}$$



# Vitamin design

How we construct the individual networks

- Each probability distribution is modelled by a network that takes inputs and outputs the parameters governing a distribution
- Each is a deep convolutional network
- In our case we use 15-dim uncorrelated Gaussians for q(z|x,y) and r(x|y,z)
- We choose to use a 15-dim Gaussian mixture model for r(z|y)

[Gabbard et al, arXiv 1909.06296 (2019)]

TABLE III. The `VItamin` network hyper-parameters

| Layer \ Network | $r_{\theta_1}(z\|y)$ | $r_{\theta_2}(x\|y,z)$ | $q_\phi(z\|x,y)$ |
|---|---|---|---|
| Input $y$ | [256,3][a] | [256,3] | [256,3] |
| Layer 1 | conv(5,3,33)[b]<br>act[c]=ReLU | conv(5,3,33)<br>act=ReLU | conv(5,3,33)<br>act=ReLU |
| Layer 2 | conv(8,33,33)<br>maxpool(2,2)[d]<br>act=ReLU | conv(8,33,33)<br>maxpool(2,2)<br>act=ReLU | conv(8,33,33)<br>maxpool(2,2)<br>act=ReLU |
| Layer 3 | conv(11,33,33)<br>act=ReLU | conv(11,33,33)<br>act=ReLU | conv(11,33,33)<br>act=ReLU |
| Input $z, x$ | - | flatten[e]$\rightarrow$[4224]<br>append[f]$(z)\rightarrow$[4234] | flatten$\rightarrow$[4224]<br>append$(x)\rightarrow$[4231] |
| Layer 4 | conv(10,33,33)<br>maxpool(2,2)<br>act=ReLU | FC(4234,2048)[g]<br>dropout(0.2)[h]<br>act=ReLU | FC(4231,2048)<br>dropout(0.2)<br>act=ReLU |
| Layer 5 | conv(10,33,33)<br>act=ReLU<br>flatten$\rightarrow$[2112] | FC(2048,2048)<br>dropout(0.2)<br>act=ReLU | FC(2048,2048)<br>dropout(0.2)<br>act=ReLU |
| Layer 6 | FC(2112,2048)<br>dropout=0.2<br>act=ReLU | FC(2048,14)<br>act=(Sigmoid,-ReLU)[i]<br>output=$\mu_{r_2}$<br>$\rightarrow$[7,2][j] | FC(2048,20)<br>act=None<br>output=$\mu_q$<br>$\rightarrow$[10,2][k] |
| Layer 7 | FC(2048,2048)<br>dropout(0.2)<br>act=ReLU | | |
| Layer 8 | FC(2048,320)<br>act=None<br>output=$\mu_{r_1}$<br>$\rightarrow$[10,16,2][l] | | |

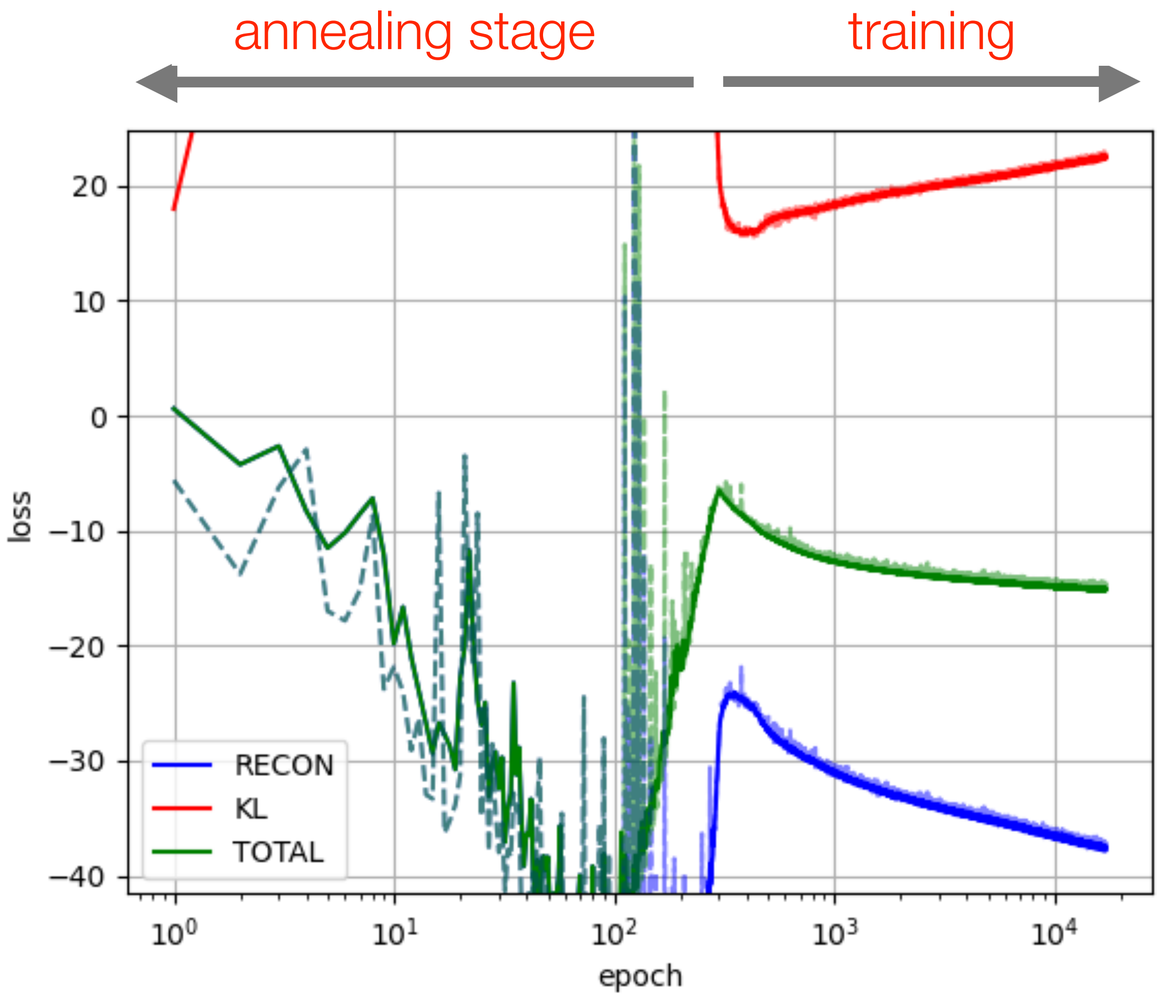
Representative but now outdated

Updated version of Gabbard et al, arXiv 1909.06296 (2019)

# Vitamin results

## The training process

- Needs lots of training data in the form of examples of noisy signals (y) plus the true signal parameters (x)
- Need a GPU and still takes ~days
- We do not need any costly pre-computed posteriors
- The total cost/loss is minimised at the expense of increased KL

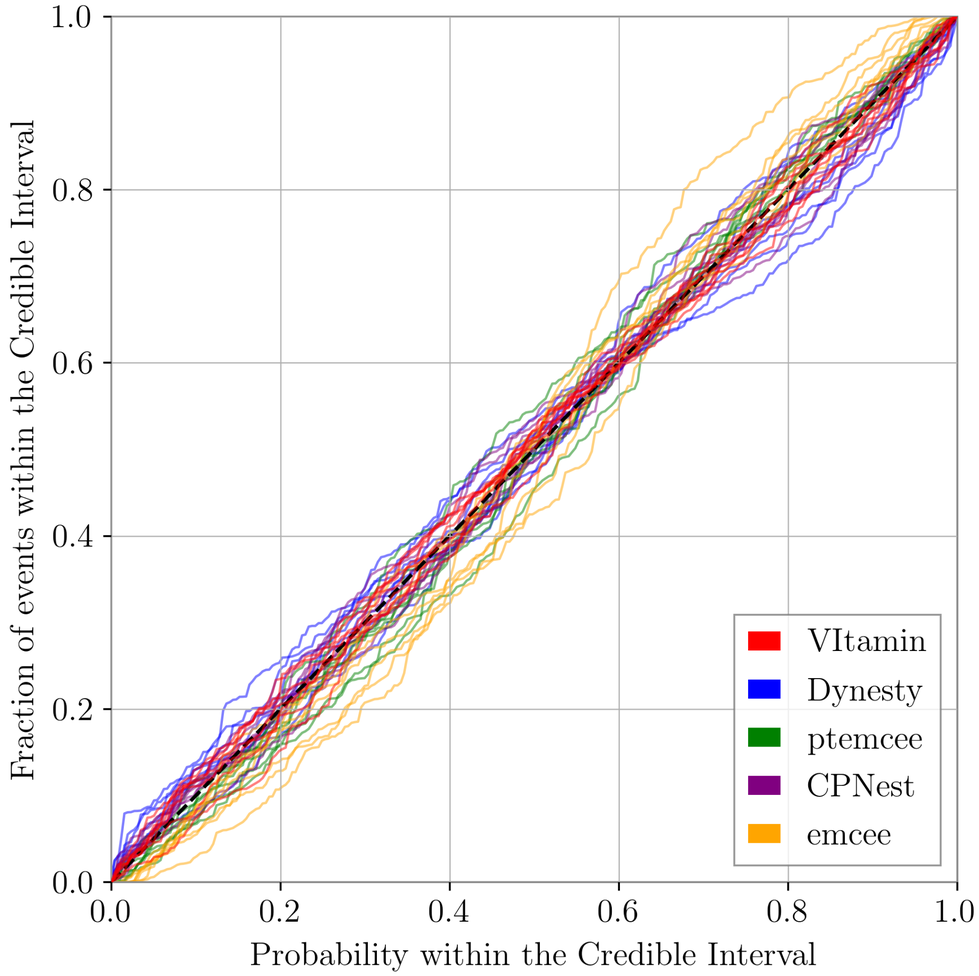


# Vitamin results

## The validation process

Gabbard et al, arXiv 1909.06296 (2019)

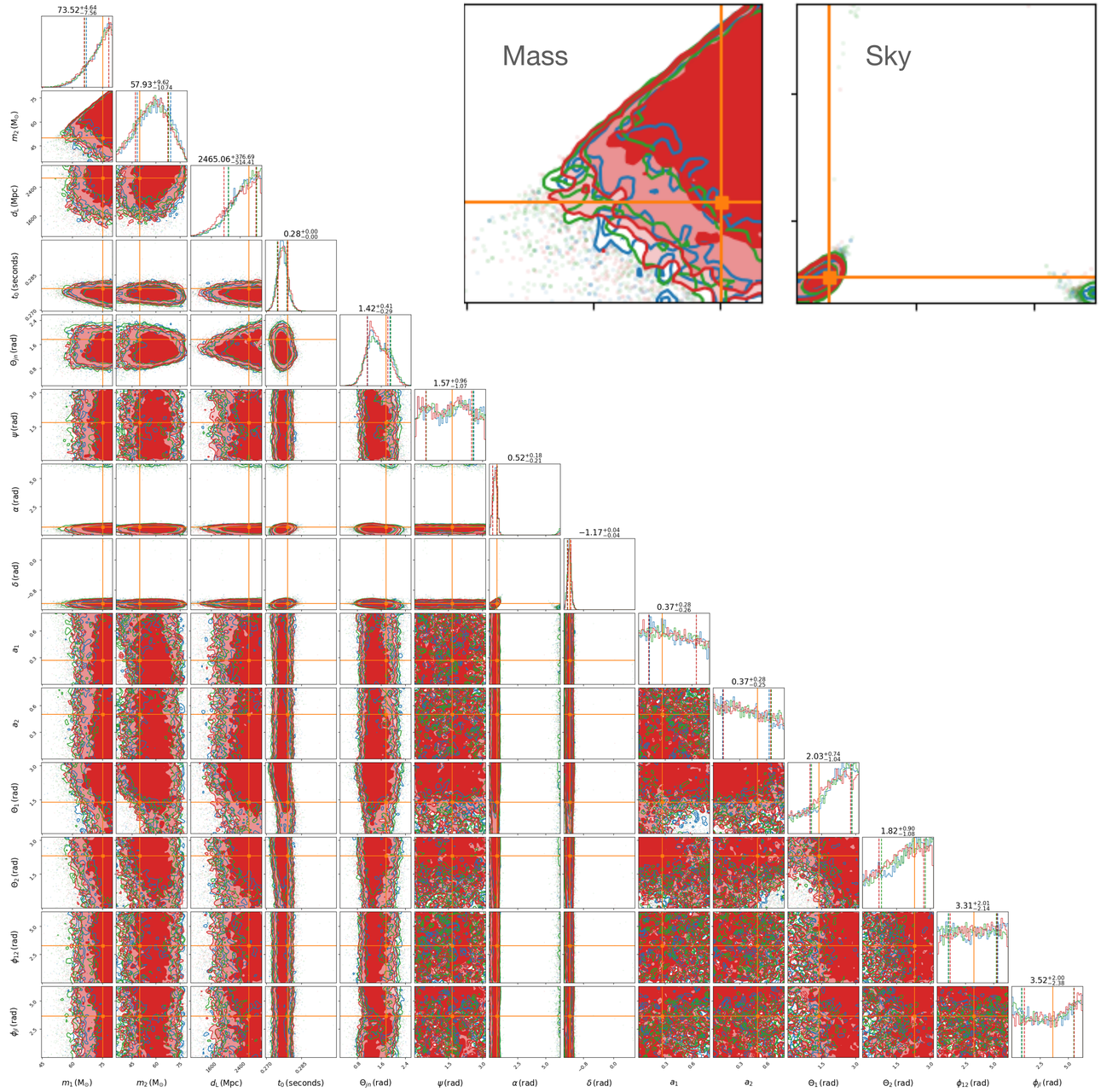
- We are able to test the statistical consistency of the outputs.
- A p-p plot basically compares the Bayesian confidence with the frequentist interpretation
- Doesn’t prove that the output posteriors are correct - just that they are probabilistically consistent



# Vitamin posteriors

## Compared with existing approaches

- We run a number of (very) costly analyses with existing sampling approaches for comparison
- Results do not agree perfectly and there is still work to do to fine tune the networks
- Note that existing samplers also disagree in some circumstances




Updated version of Gabbard et al, arXiv 1909.06296 (2019)

# Vitamin speed

There is no contest (unless you count training)

- The primary difference is that the CVAE is pre-trained so that all cost is up-front
- We get a ~6 order of magnitude speed up in our test cases
- Can now generate $10^4$ posterior samples in <1 sec
- Training still takes O(days) but needs to only be done once*

Gabbard et al, arXiv 1909.06296 (2019)

TABLE I. Durations required to produce samples from each of the different posterior sampling approaches.

| sampler | run time (seconds) min | max | median | ratio $\frac{\tau_{\mathrm{VItamin}}}{\tau_X}$ |
|---|---|---|---|---|
| `Dynesty`[a] [15] | 11795 | 29838 | 19400 [b] | $5.2 \times 10^{-6}$ |
| `emcee` [16] | 18838 | 69272 | 32070 | $3.1 \times 10^{-6}$ |
| `ptemcee` [17] | 17124 | 37446 | 24372 | $4.1 \times 10^{-6}$ |
| `CPNest` [14] | 9943 | 53315 | 26202 | $3.8 \times 10^{-6}$ |
| `VItamin`[c] | $\mathbf{1 \times 10^{-1}}$ | | | 1 |

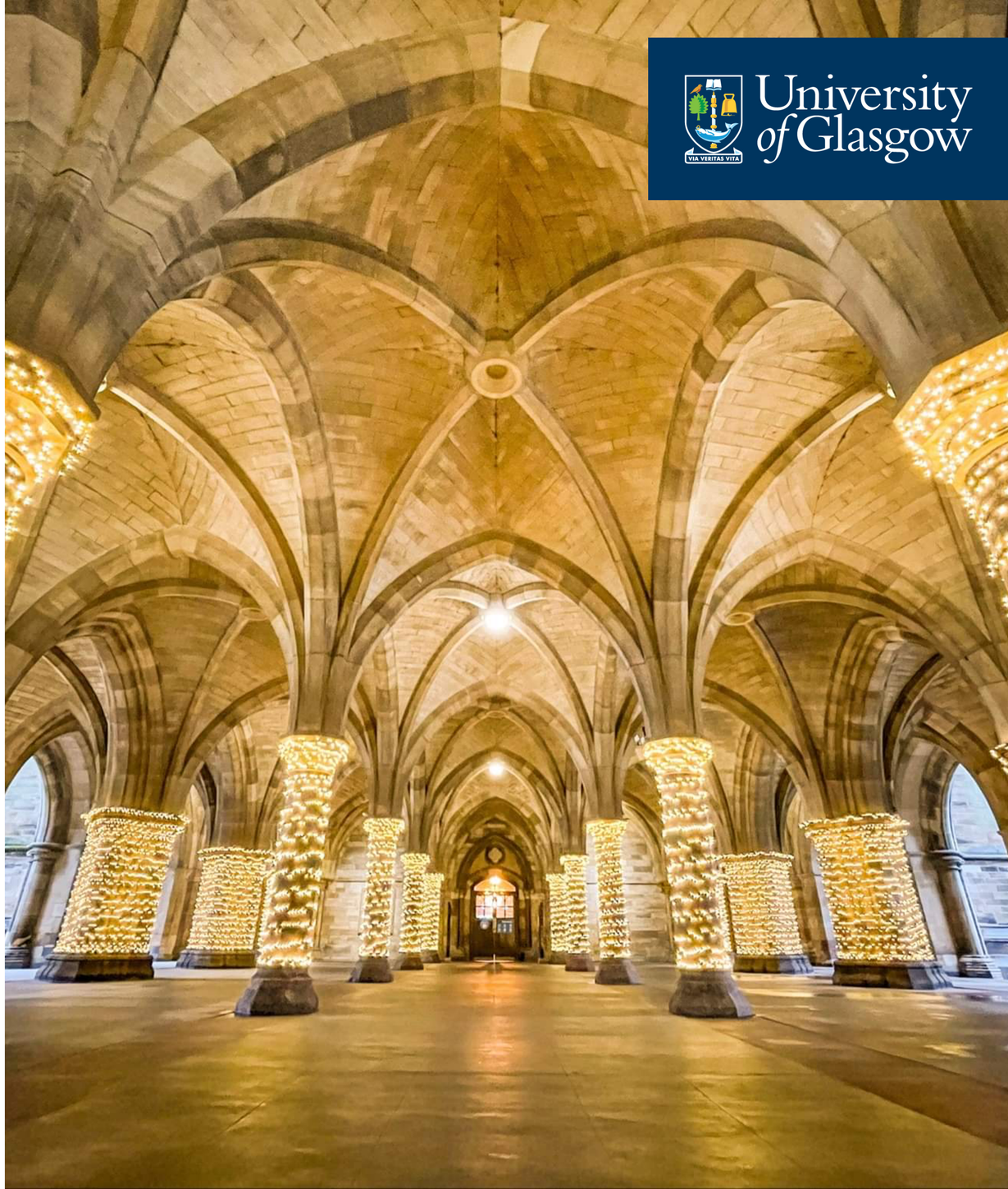
[a] The benchmark samplers all produced $\mathcal{O}(10000)$ samples dependent on the default sampling parameters used.

[b] We note that there are a growing number of specialised techniques [31–33] designed to speed up traditional sampling algorithms that could be used to reduce the runtimes quoted here by $\mathcal{O}(1-2)$ orders of magnitude.

[c] For the `VItamin` sampler 10000 samples are produced as representative of a typical posterior. The run time is independent of the signal content in the data and is therefore constant for all test cases.

# Summary

Its all good but there’s still a lot more to be done



# Summary

- ML can provide a direct replacement for existing Bayesian parameter estimation
- This will enable realtime multi messenger astronomy
- There is also the scope for pre-merger detections
- There are still many challenges, e.g., real detector noise, longer duration signals, etc…
- This type of analysis is applicable to general Bayesian inference problems
- Other solutions are available, e.g., Normalising Flows [Kobyzev et al, arXiv 1908.09257 (2019)]

# Thank you for your attention

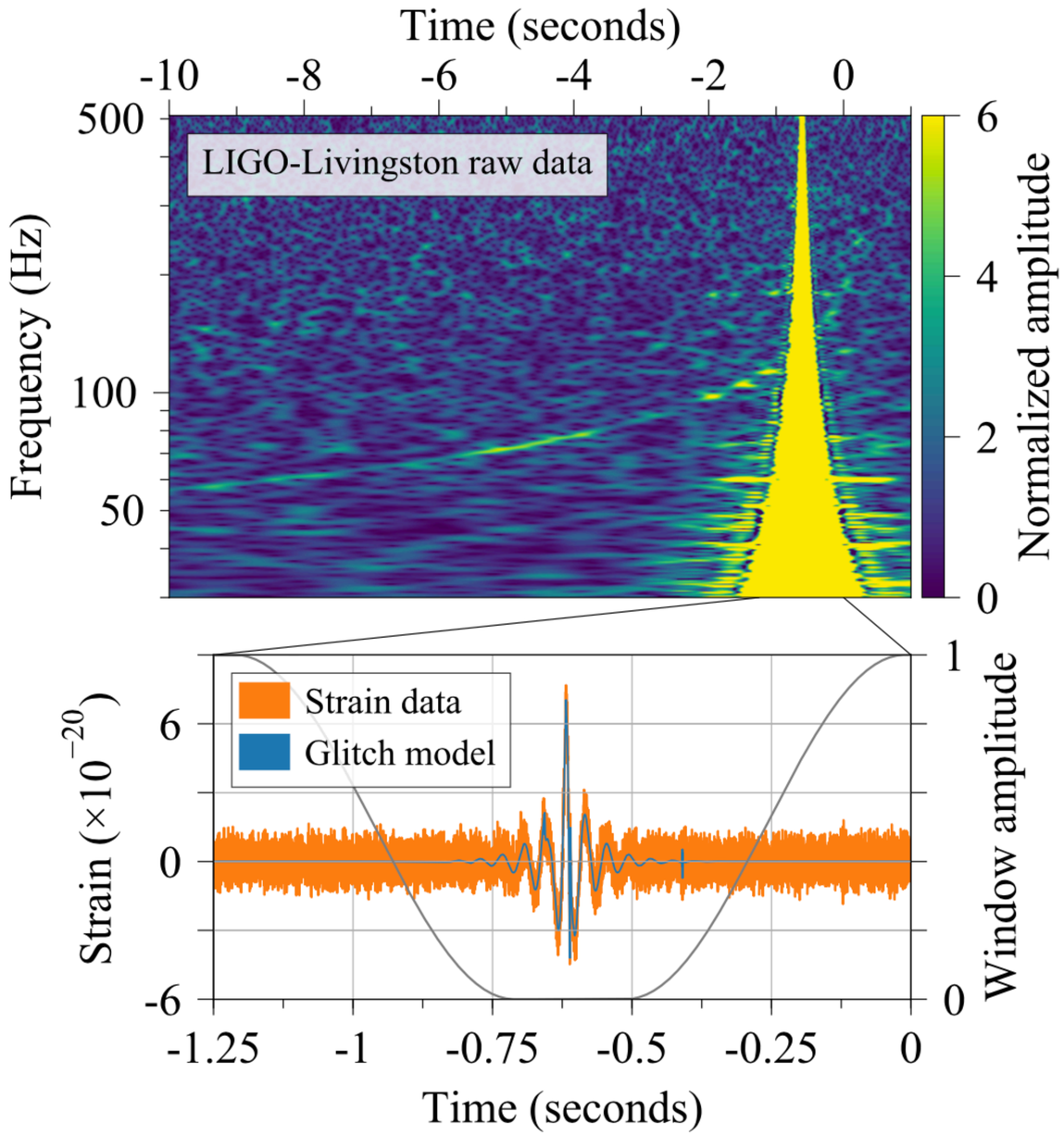
# Extra slides

LVC Collaboration, PRL, 119, 161101 (2017)

# Non-Gaussian Noise

## Train on real GW (noise) data

- Real GW data is not exactly Gaussian
- We are in the process of training our models using real historical GW detector noise
- This will allow us to also make the network conditional on the noise properties (PSD)
- Training on rare transient detector noise events will be challenging



# Refinement

## Resampling VItamin samples

- An idea borrowed from colleagues at Monash [Payne et al, PRD 100, 12, 2019]
- We take the approximate Vitamin points and then compute

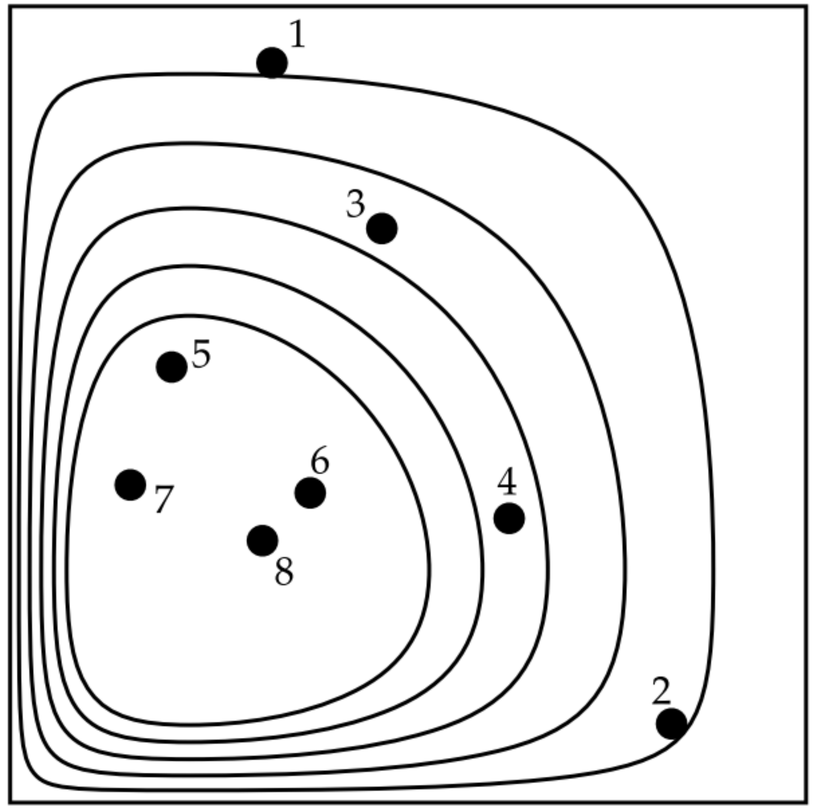
$$r(x|y) = \langle r(x|z,y)\rangle|_{z\sim r(z|y)}$$

- We then use importance/rejection sampling to resample to achieve refined results

# Nessai https://github.com/mj-will/nessai

Using ML to enhance existing methods

- One of the bottlenecks in classical nested sampling is generating points that
  - are sampled from the prior, and
  - also within a given likelihood contour
- Normalising flows translate simple distributions to more complicated ones and have been shown to accelerate Nested sampling [Williams et al arXiv 2102.11056 (2021)].

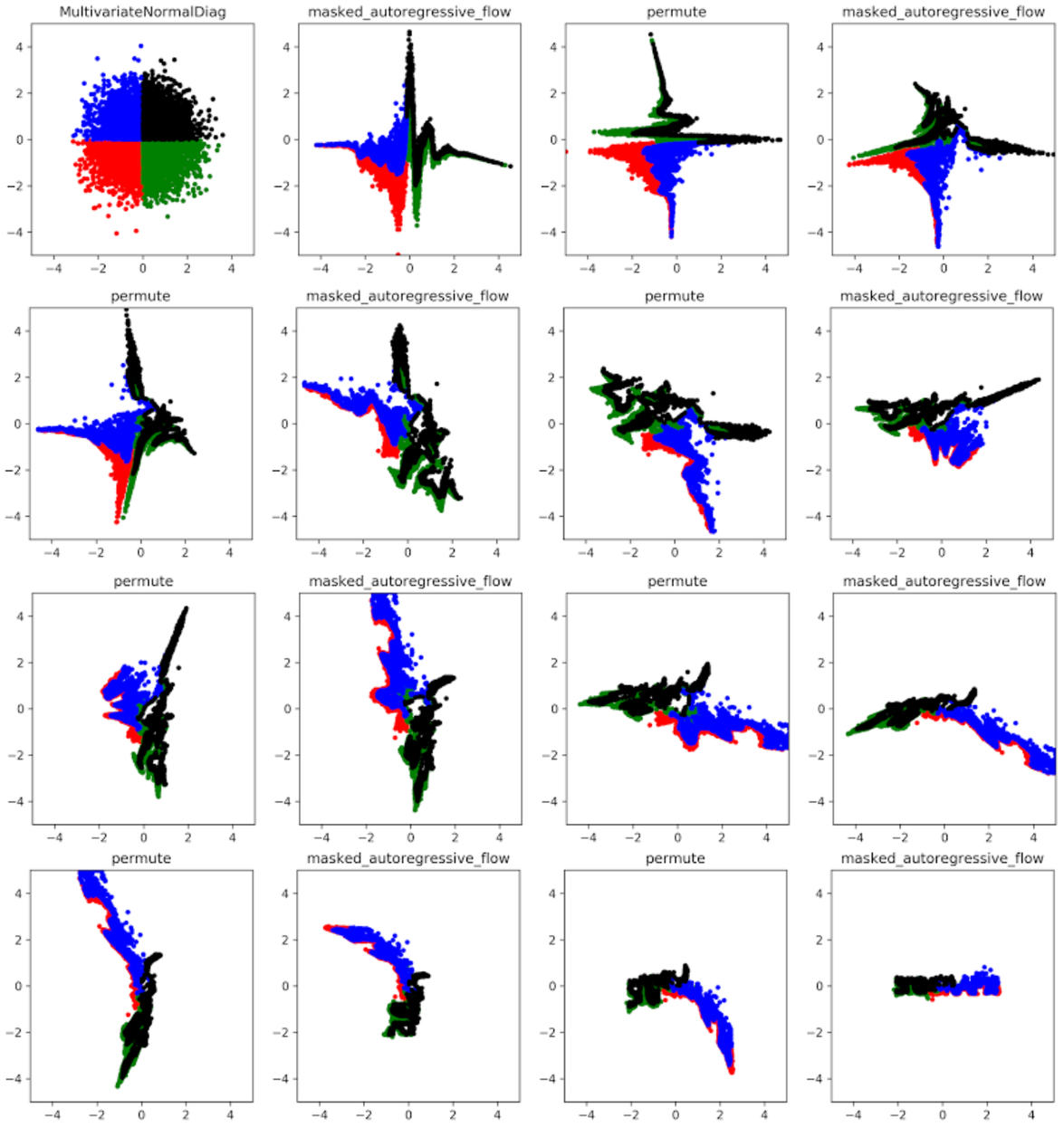


# Normalising Flows

Another way of doing this

- Another Likelihood-free approach that can also obtain Bayesian posteriors is Normalising Flows [Green et al, PRD 102, 10 (2020)]
- These are generative models which produce tractable distributions where both sampling and density evaluation can be efficient and exact

$$p_X(x) = p_Z(f(x)) \left| \det \left( \frac{\partial f(x)}{\partial x^T} \right) \right|$$




https://blog.evjang.com/2018/01/nf2.html