Reshaping gravitational wave data analysis with machine learning

Chris Messenger
University of Glasgow

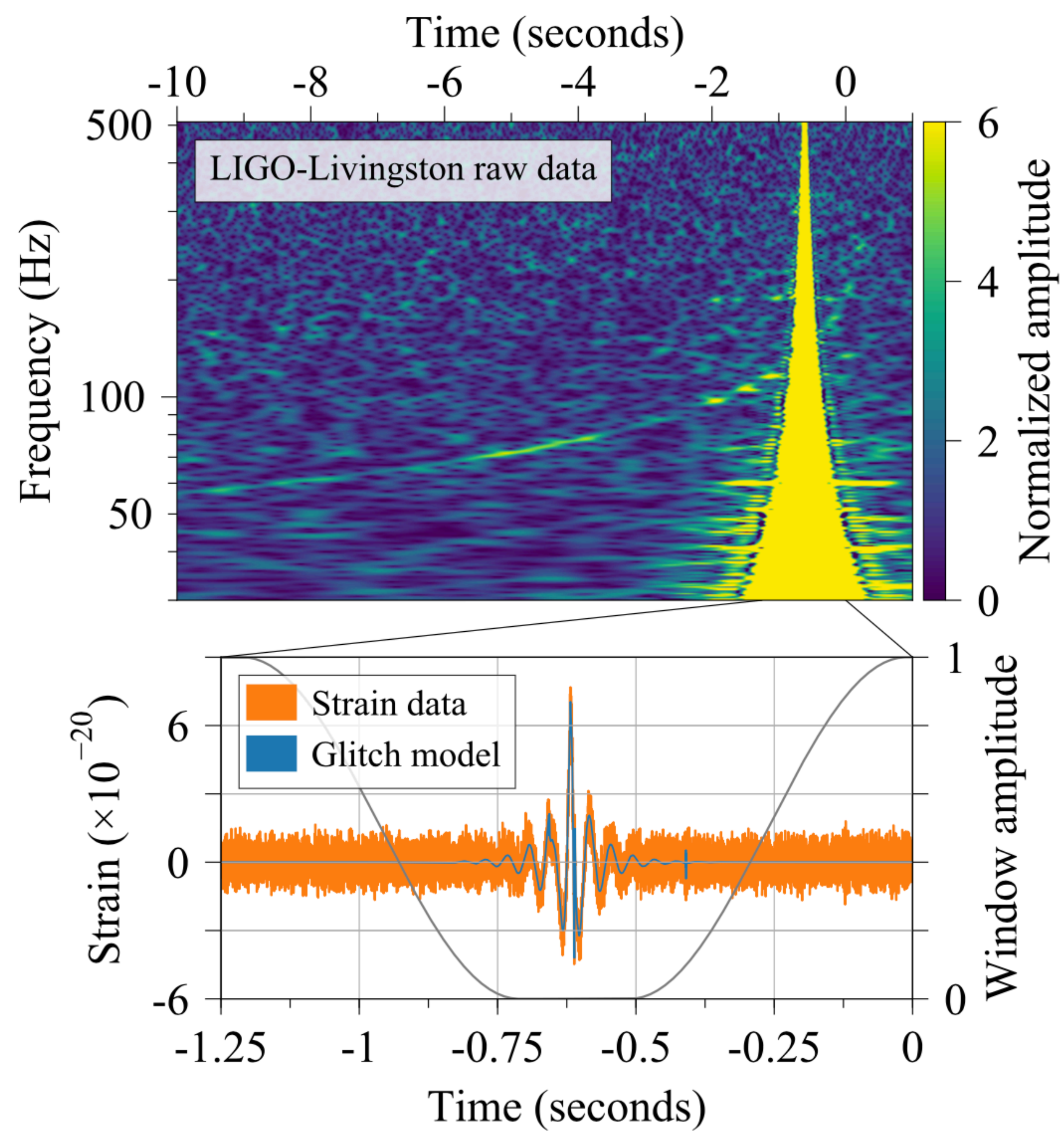
Outline

- GW data analysis
- Neural networks
- Training/Learning
- Finding GW signals
- Getting started
- GANs
- Conclusions

https://xkcd.com/1838/

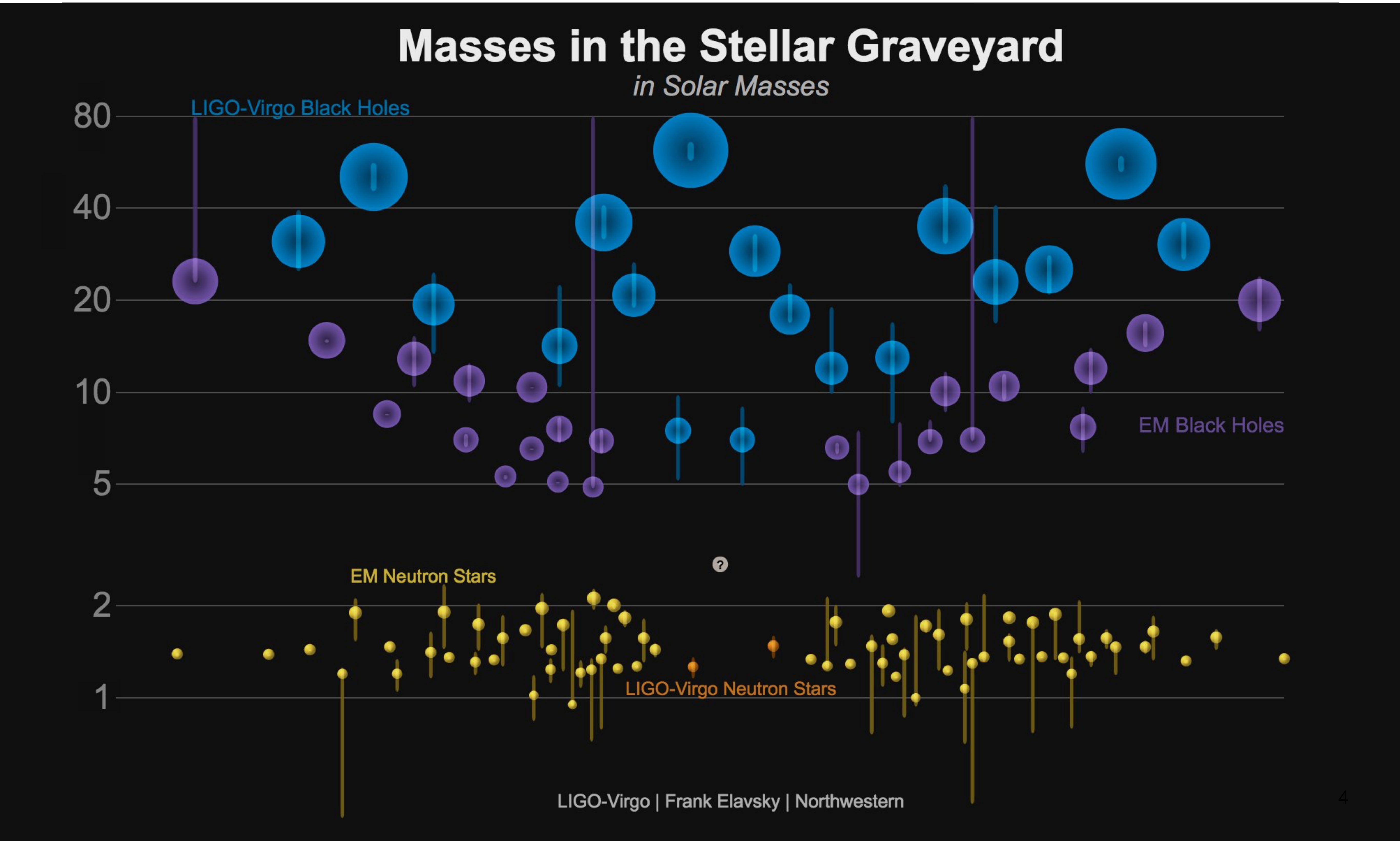
GW data analysis

What we currently do and why it needs to be improved

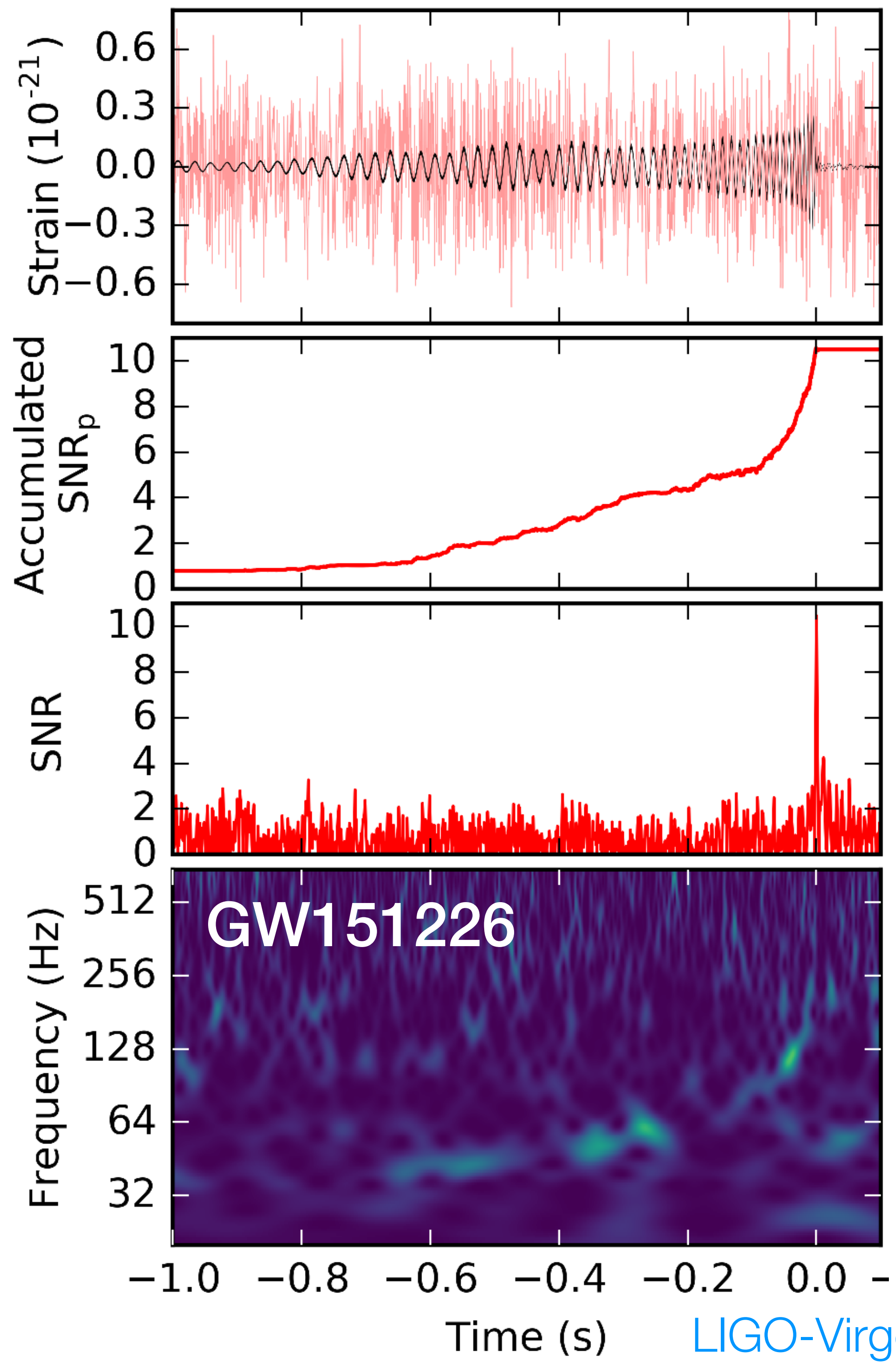


LIGO-Virgo Collaboration, PRL 119, 161101 (2017)

The detection era

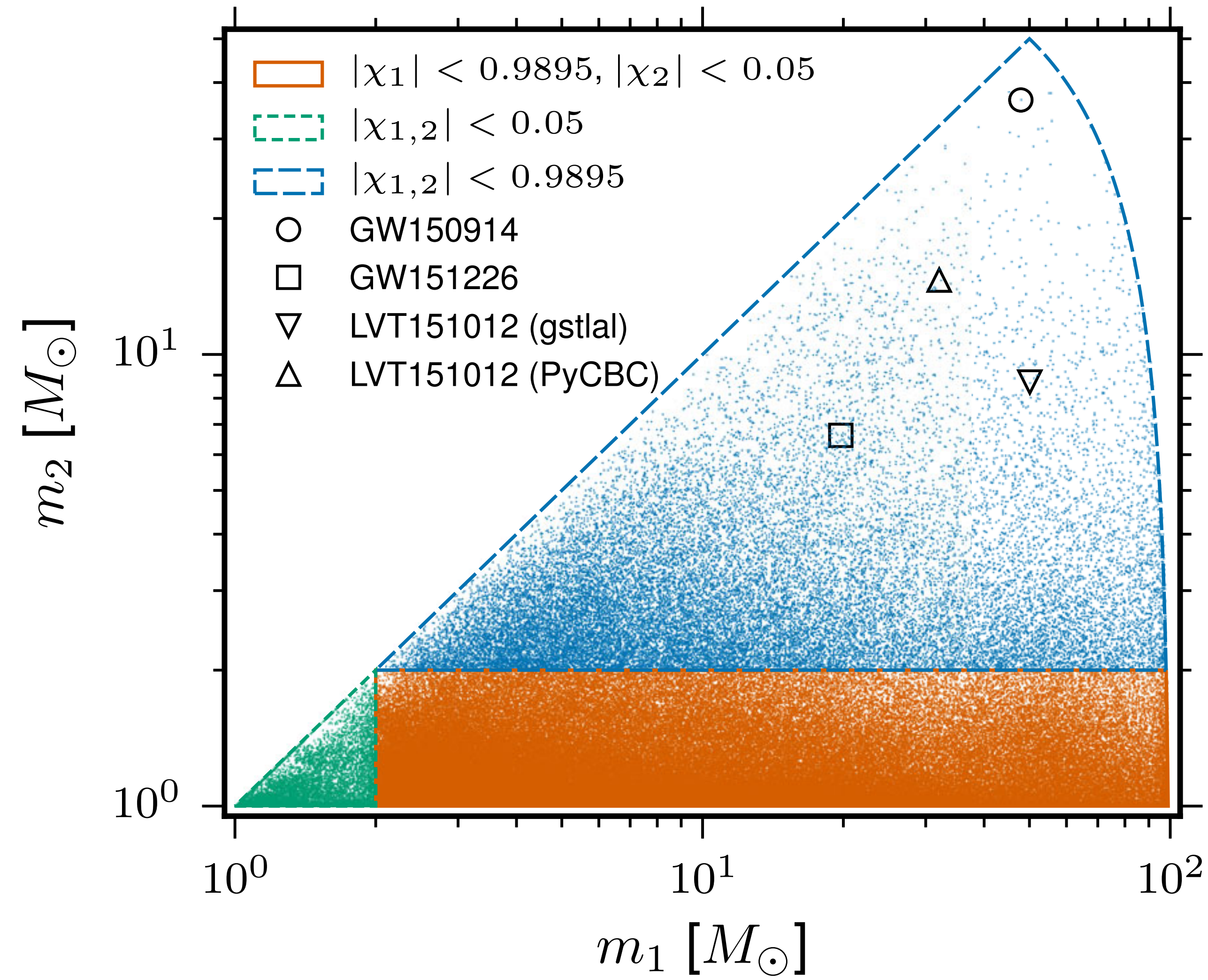


Current techniques - Searches



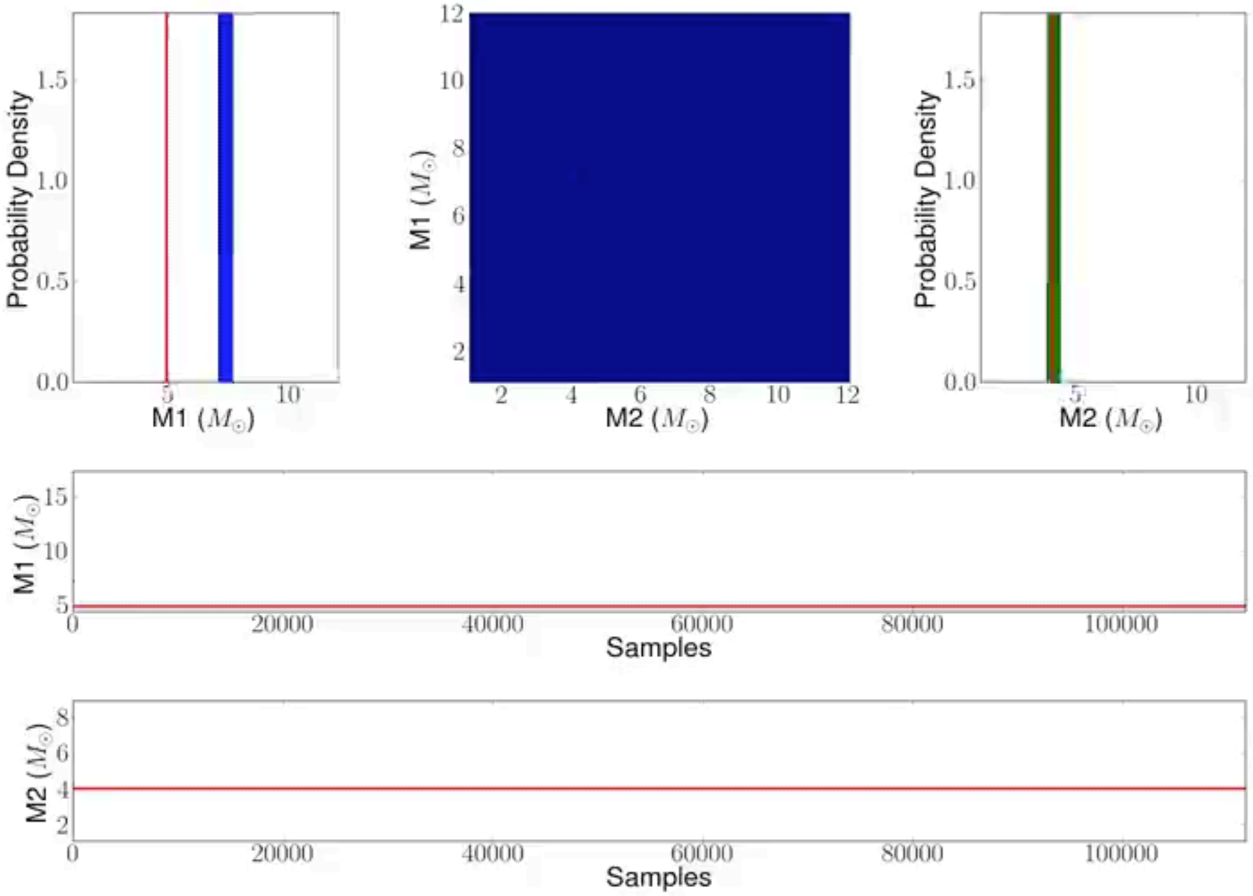
LIGO-Virgo Collaboration PRL 116, 241103 (2016)

matched-filtering

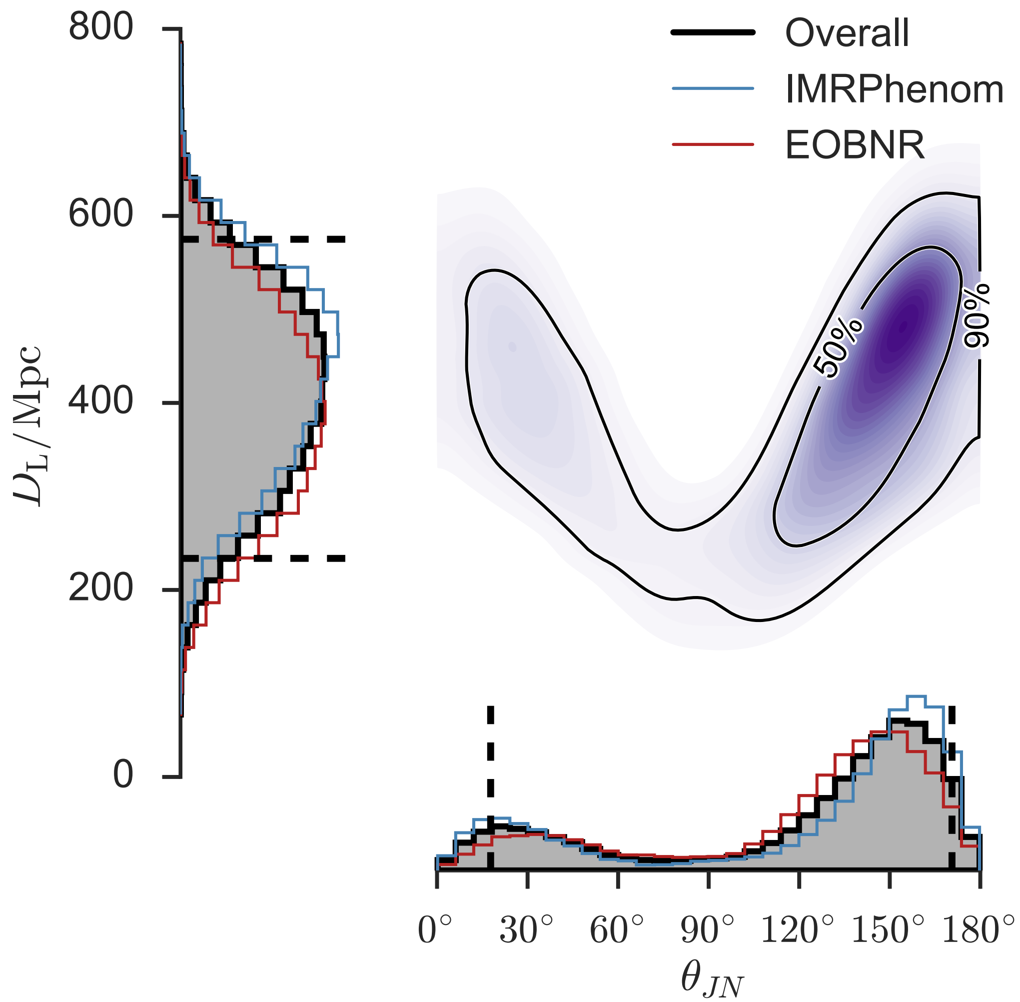


LIGO-Virgo Collaboration PRX 6, 041015 (2016)

Current techniques - Parameter Estimation



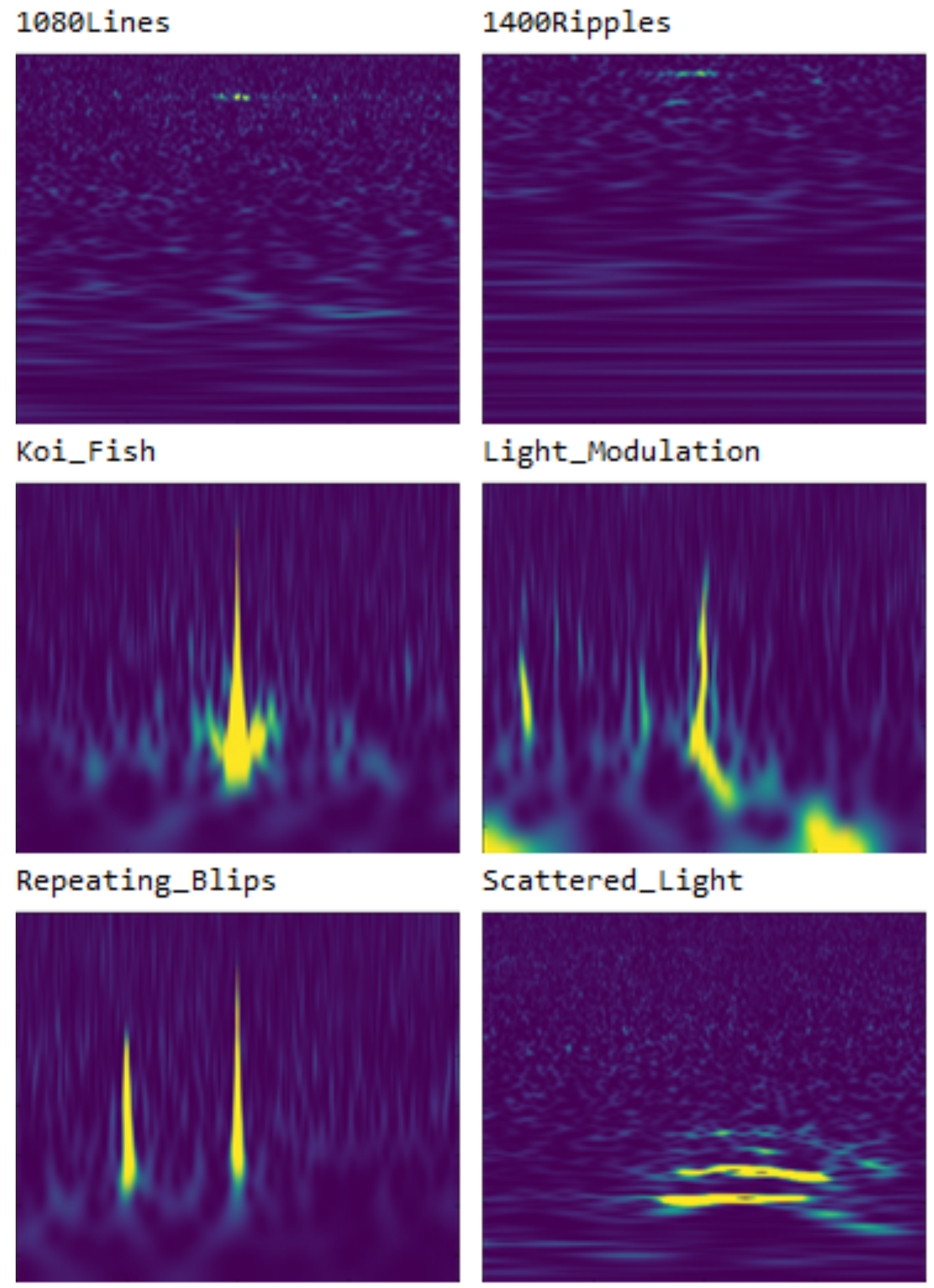
credit: Marc van der Sluys



LIGO-Virgo Collaboration PRL 116, 241102 (2016)

Current techniques - Challenges

- Non-Gaussian noise
  - currently limits the optimality of searches
- Speed of PE
  - for rapid-PE
  - and for the expected detection rate in the future
- Un-modelled signals
  - the most exciting prospect?



Zevin *et al* CQG, 34, 6, 064003 (2017)

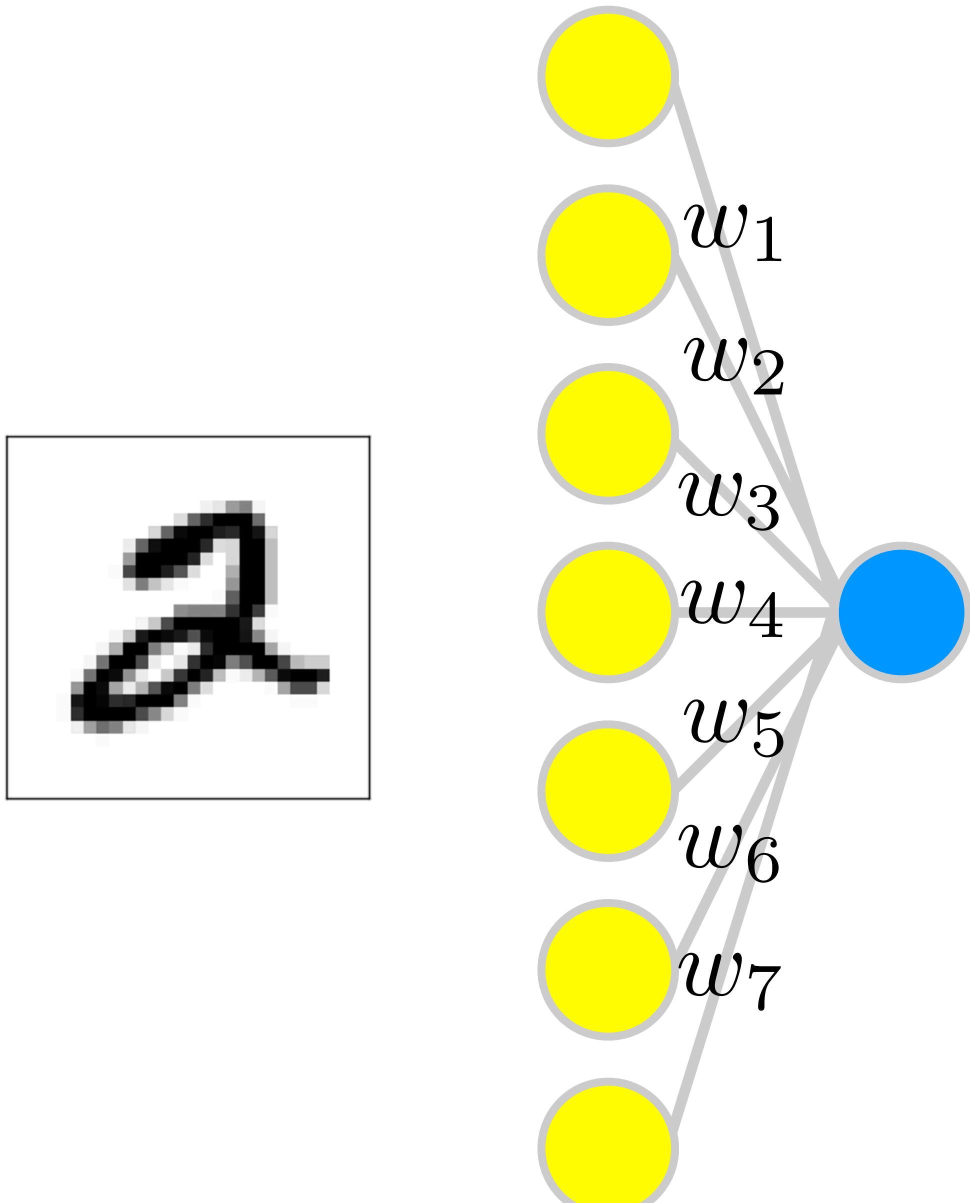
Neural networks

They're not that hard to understand
(honestly)

The MNIST dataset

LeCun et al. (1999)

A single neuron

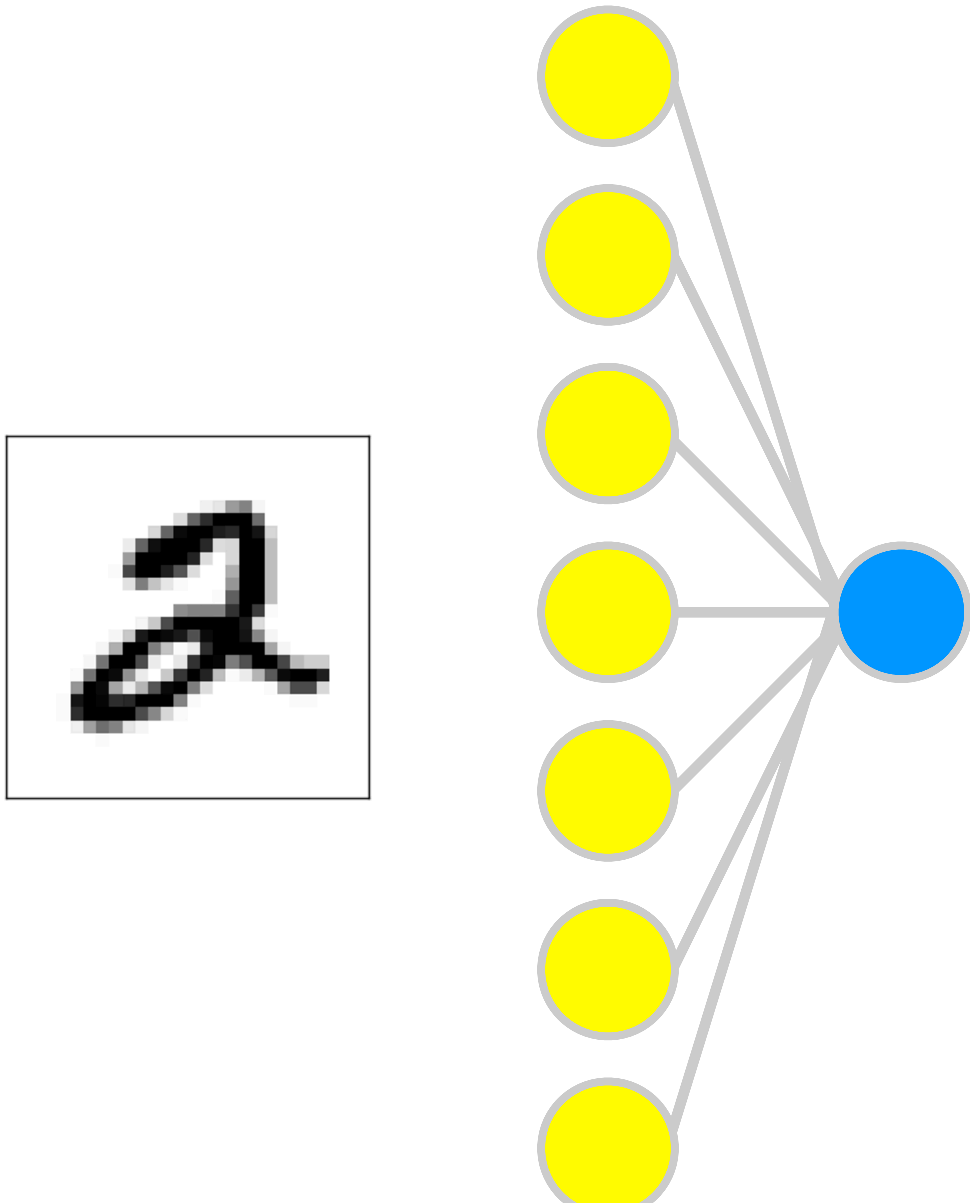


input layer

$$w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

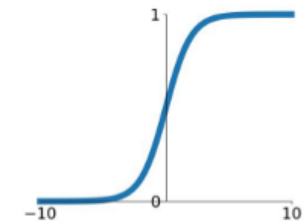
the specific values of the weights and bias are not determined until training is performed

Activation functions



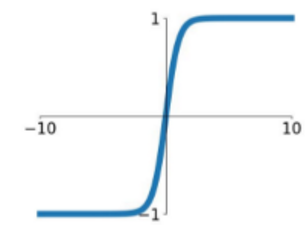
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



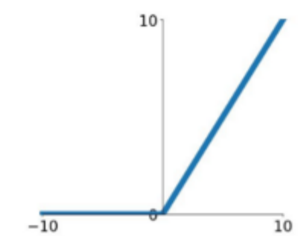
tanh

$\tanh(x)$



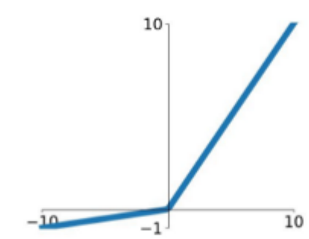
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

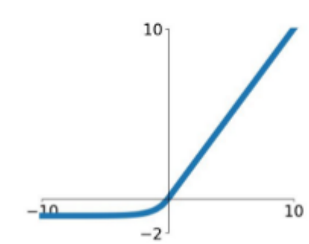


Maxout

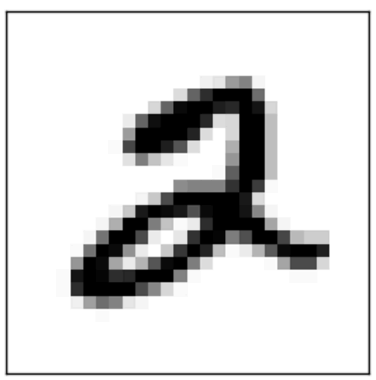
$\max(w_1^T x + b_1, w_2^T x + b_2)$

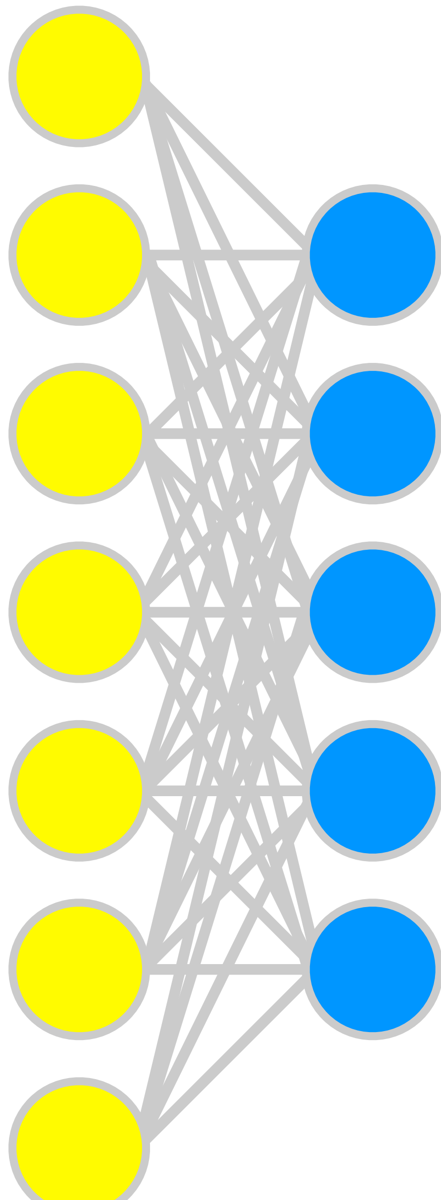
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



A simple network - A layer





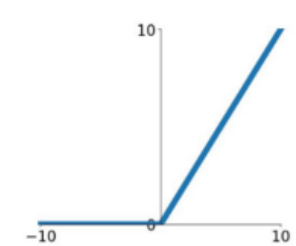
- Fully connected layer- all inputs to all neurons
- Overall result is a big matrix operation

$$\begin{bmatrix} x_1 & x_2 & \dots & x_d \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{d1} & w_{d2} & w_{d3} & \dots & w_{dn} \end{bmatrix}$$

$$+ \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix}$$

- still processed through a non-linear activation function, e.g.,

ReLU
$\max(0, x)$



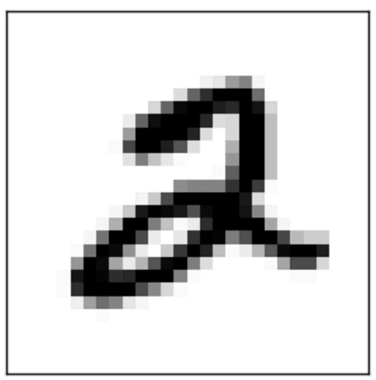
A simple network - Multiple layers

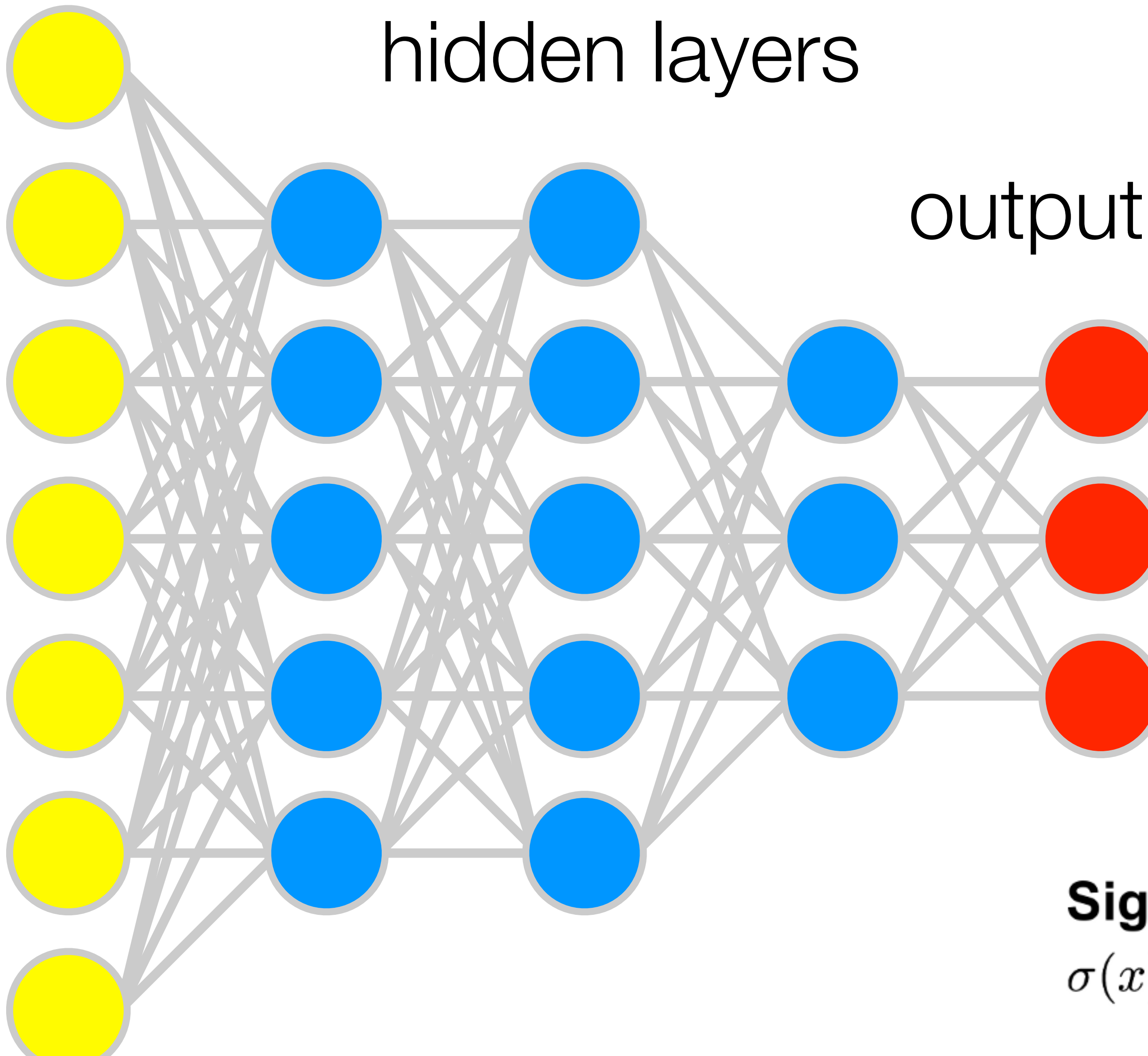
multi-layer
perceptron

input layer

hidden layers

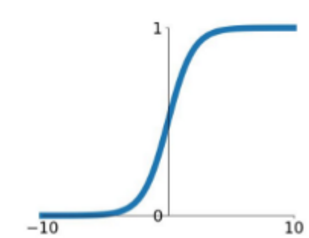
output layer





Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

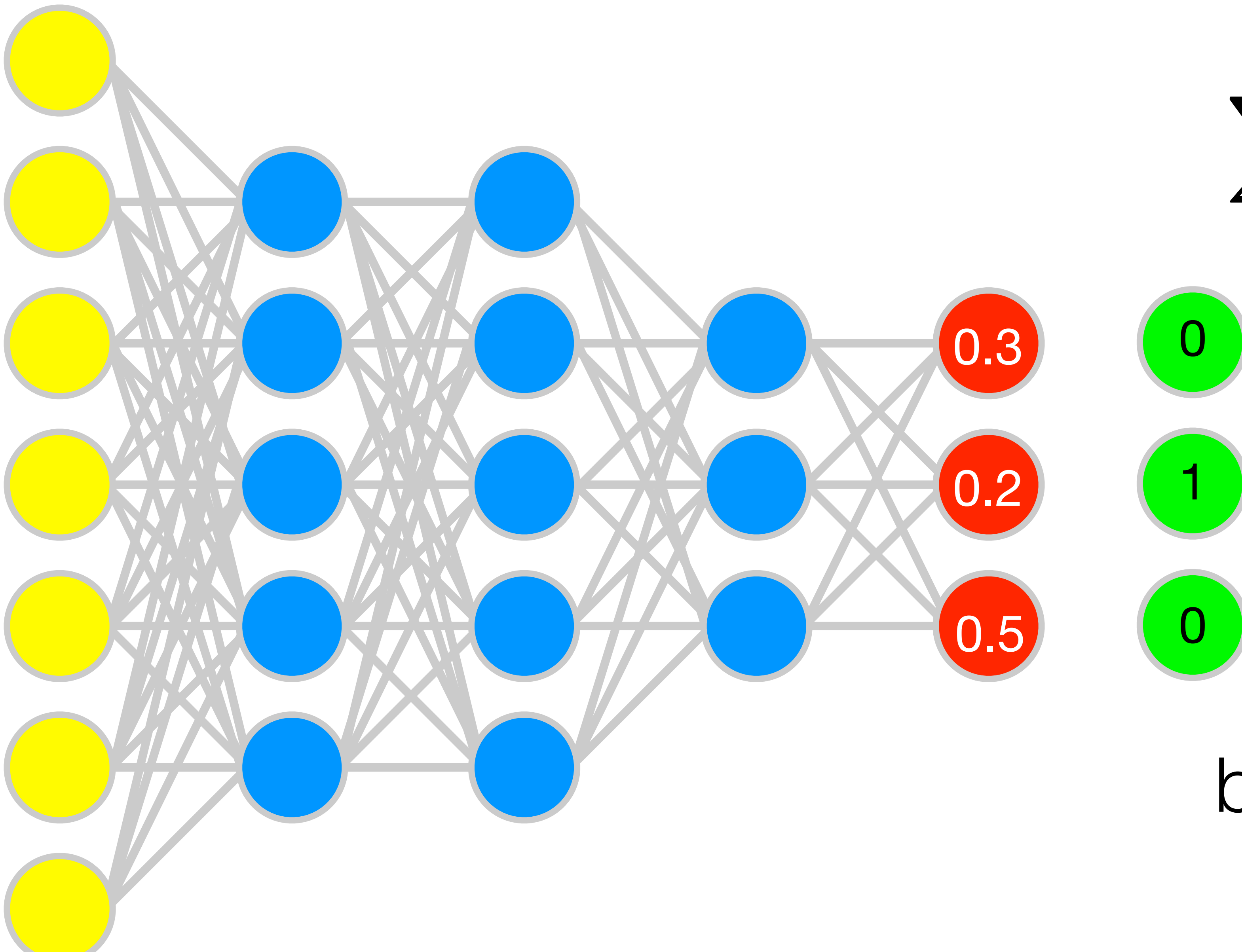


classification

Training/Learning

A brain is useless without input

Loss functions



least squares

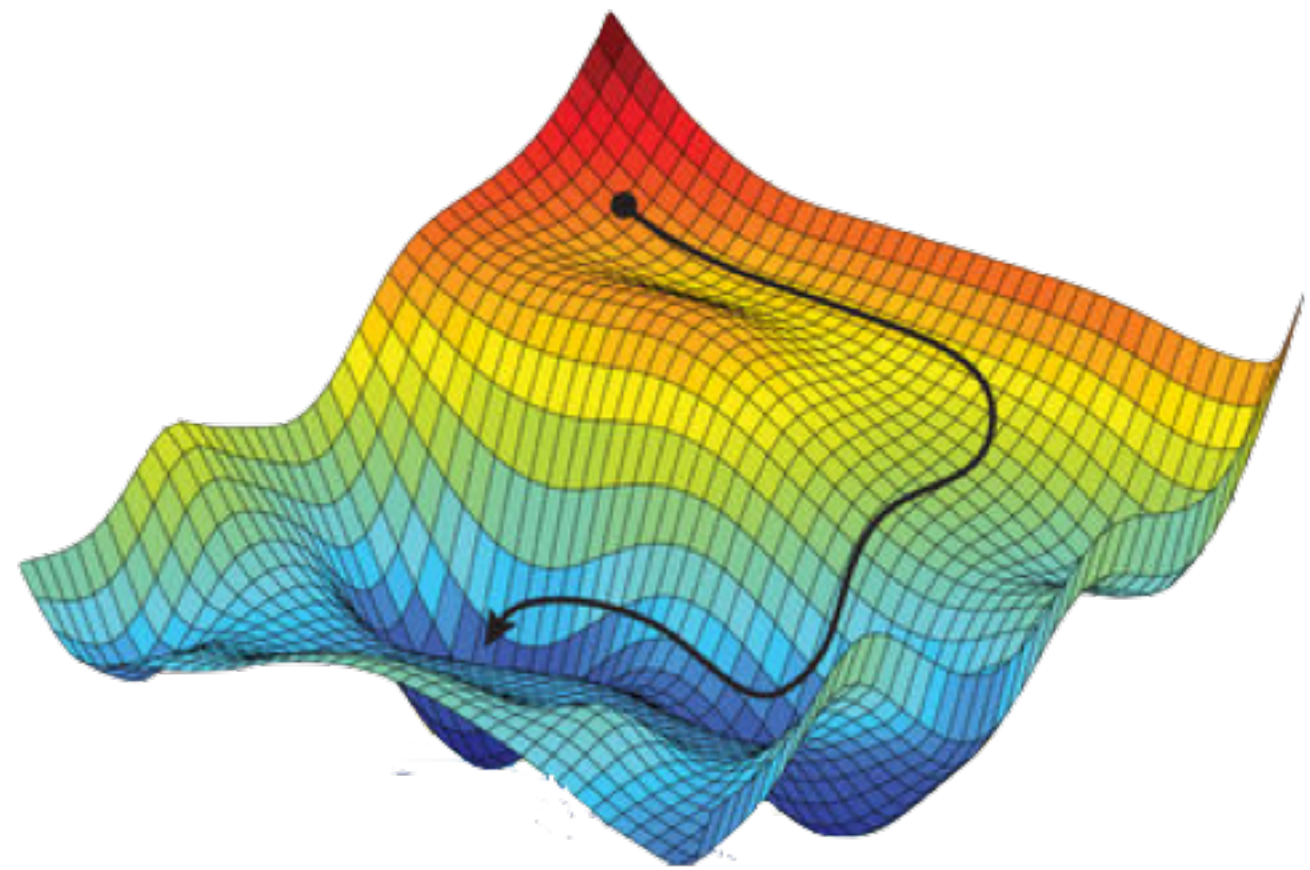
$$\sum_i \left(y_i^{\text{true}} - y_i^{\text{pred}} \right)^2$$

binary cross-entropy

$$-\sum_j^{n \text{ class}} p_j^{\text{true}} \log p_j$$

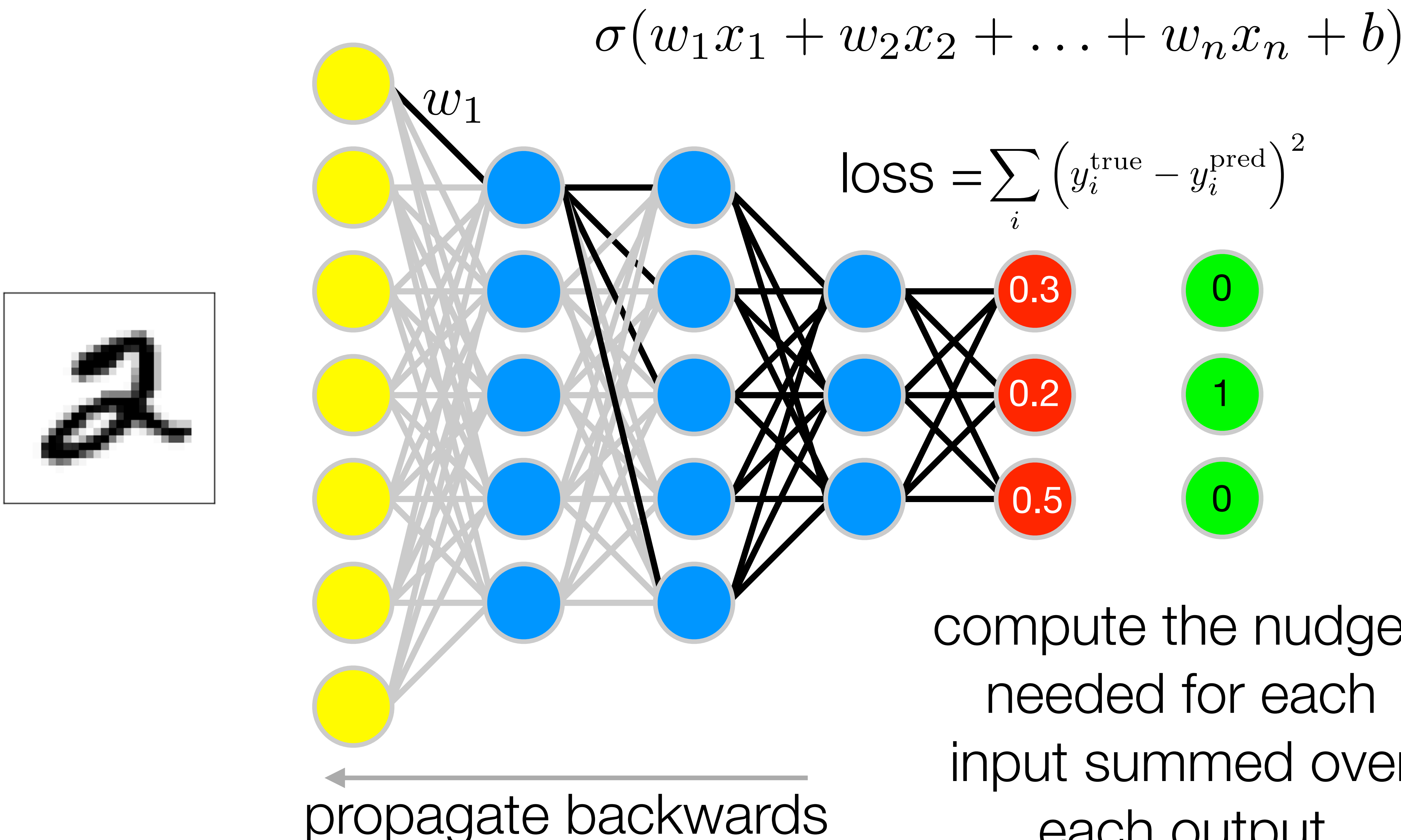
Gradient descent

- Very large number of parameters
- Compute local gradient and move “downhill” in proportion
- Tricks to avoid local minima
    - Work in “batches” - stochastic gradient descent
    - Learning rate
    - Use momentum
- in reality you can have millions of weights and biases



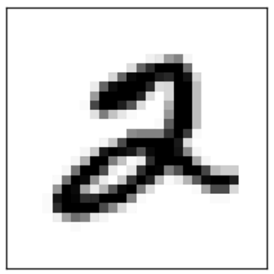
Machine “learning” here is simply minimising a loss function

Back propagation

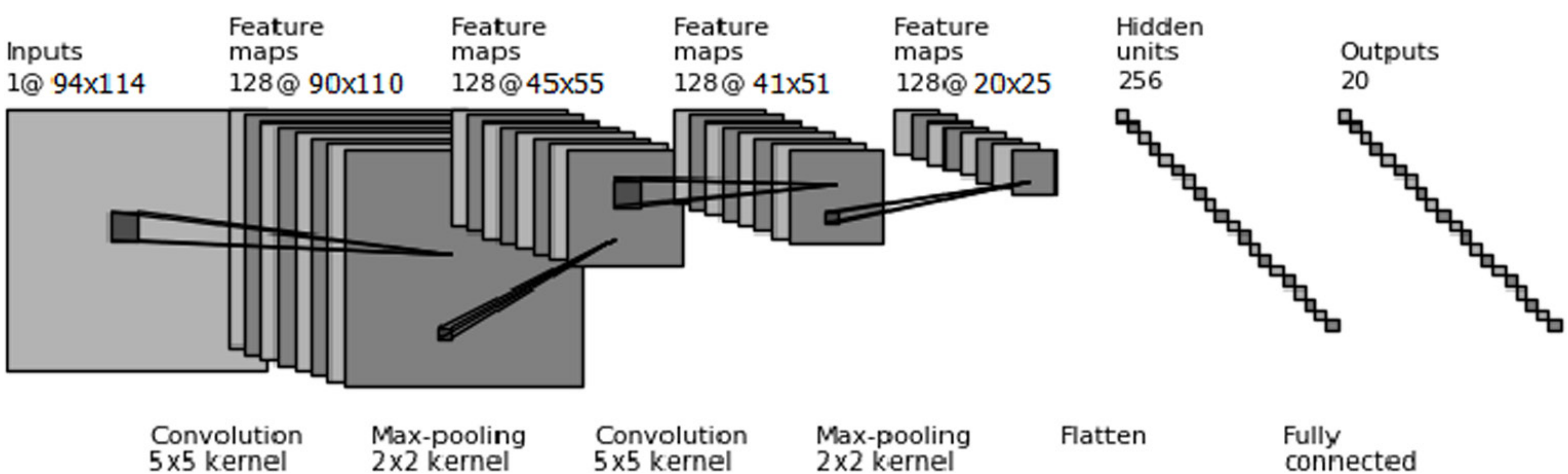


compute the nudges needed for each input summed over each output

Convolutional neurons



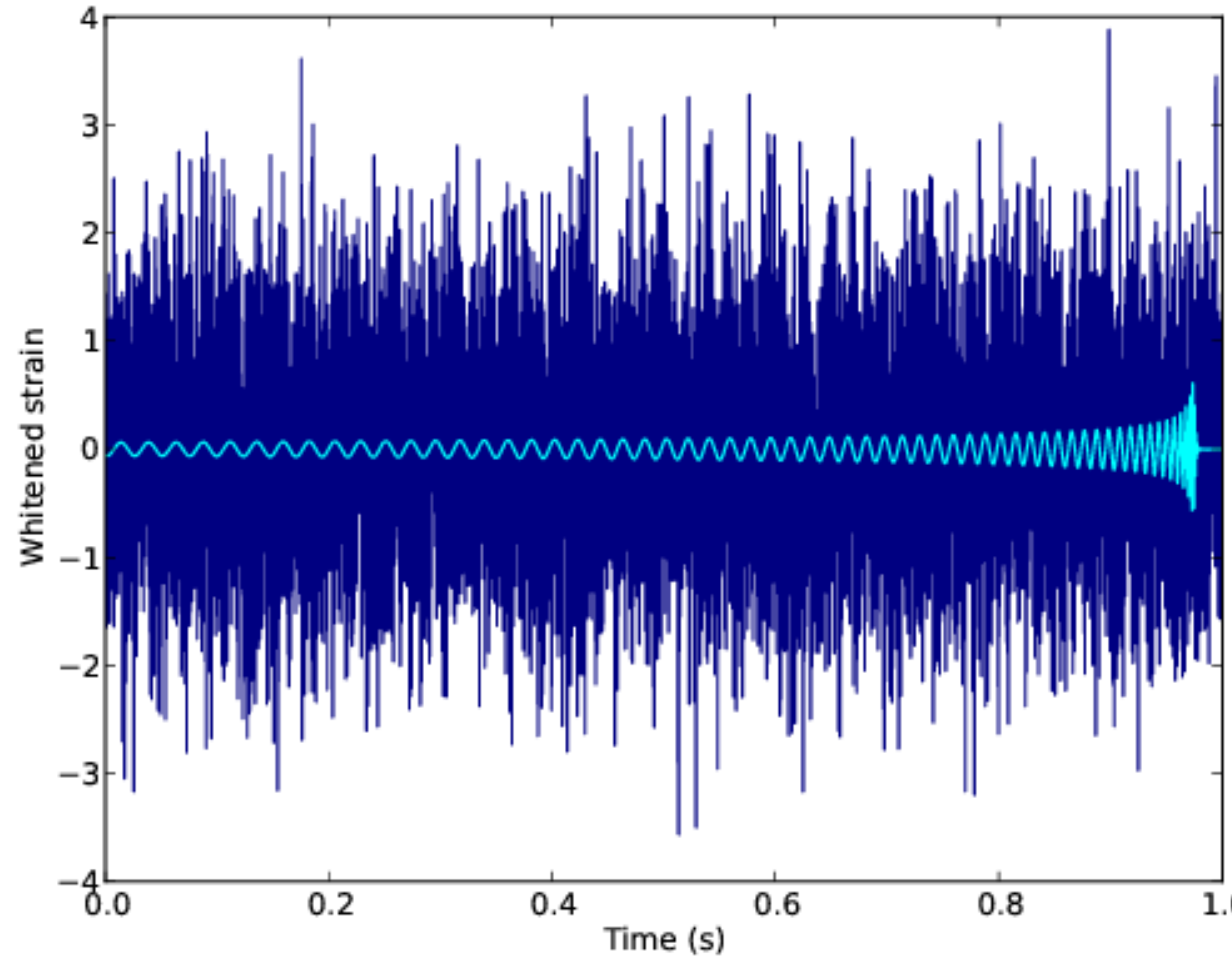
$$\begin{bmatrix} 5 & 2 & 4 & 6 \\ 3 & 9 & 7 & 1 \\ 7 & 5 & 4 & 4 \\ 8 & 7 & 1 & 8 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 14 & 9 & 5 \\ 8 & 13 & 11 \\ 14 & 6 & 12 \end{bmatrix}$$



Zevin *et al* CQG, 34, 6, 064003 (2017)

Finding GW signals

Using CNNs to find BBH signals in GW detector noise

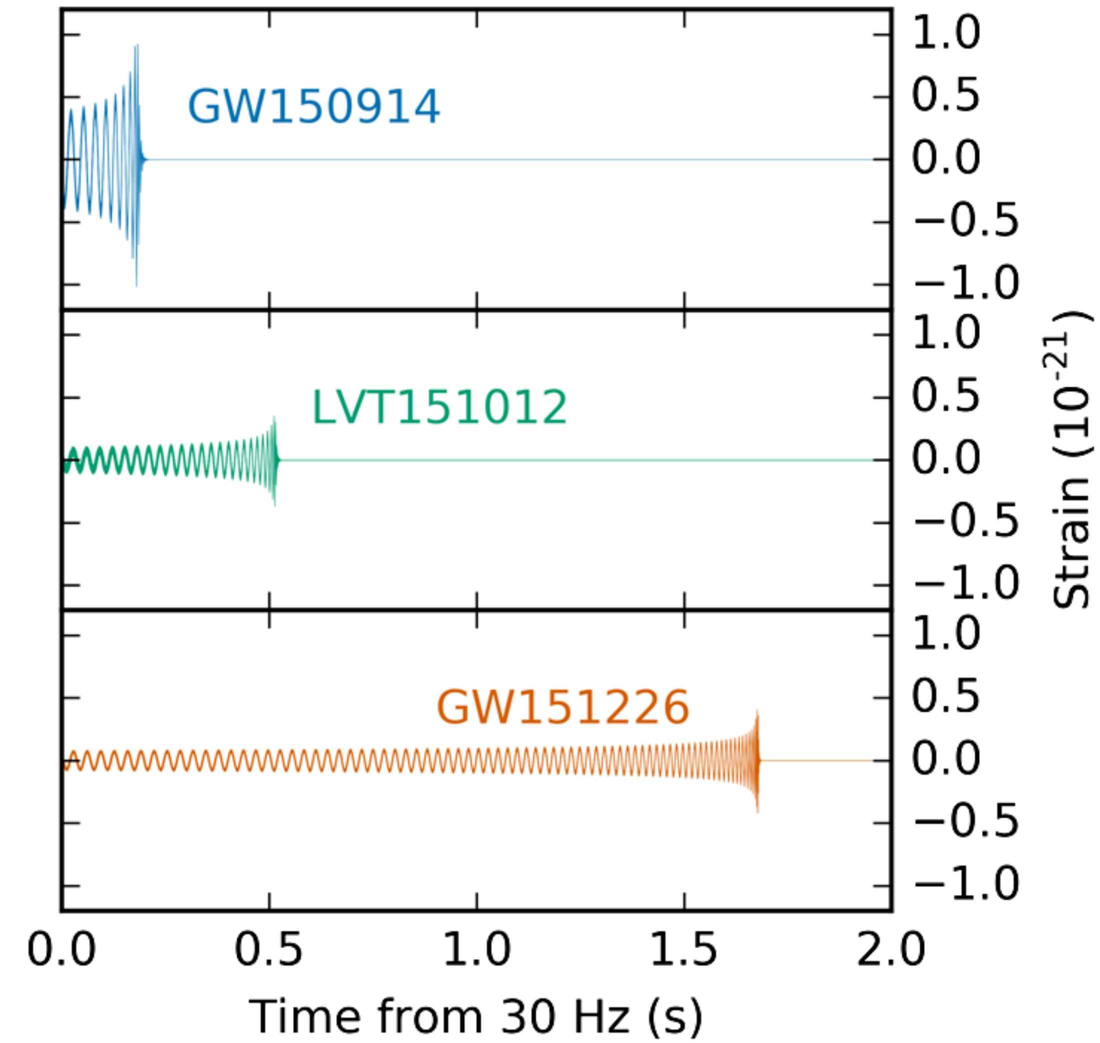


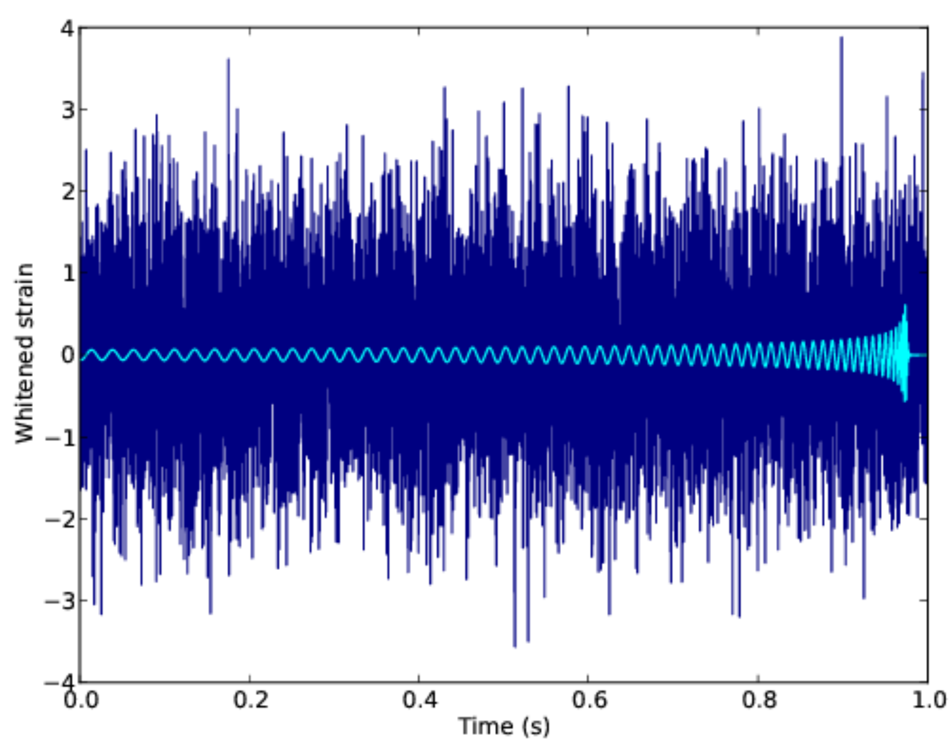
Gabbard *et al* PRL 120, 141103 (2018)

LIGO-Virgo Collaboration PRX 6, 041015 (2016)

The problem

- Design a network that takes in noisy GW data and classifies it as noise or signal+noise
- Can the network achieve the same sensitivity as matched filtering?
- We do not consider PE
- We do not consider non-Gaussianities





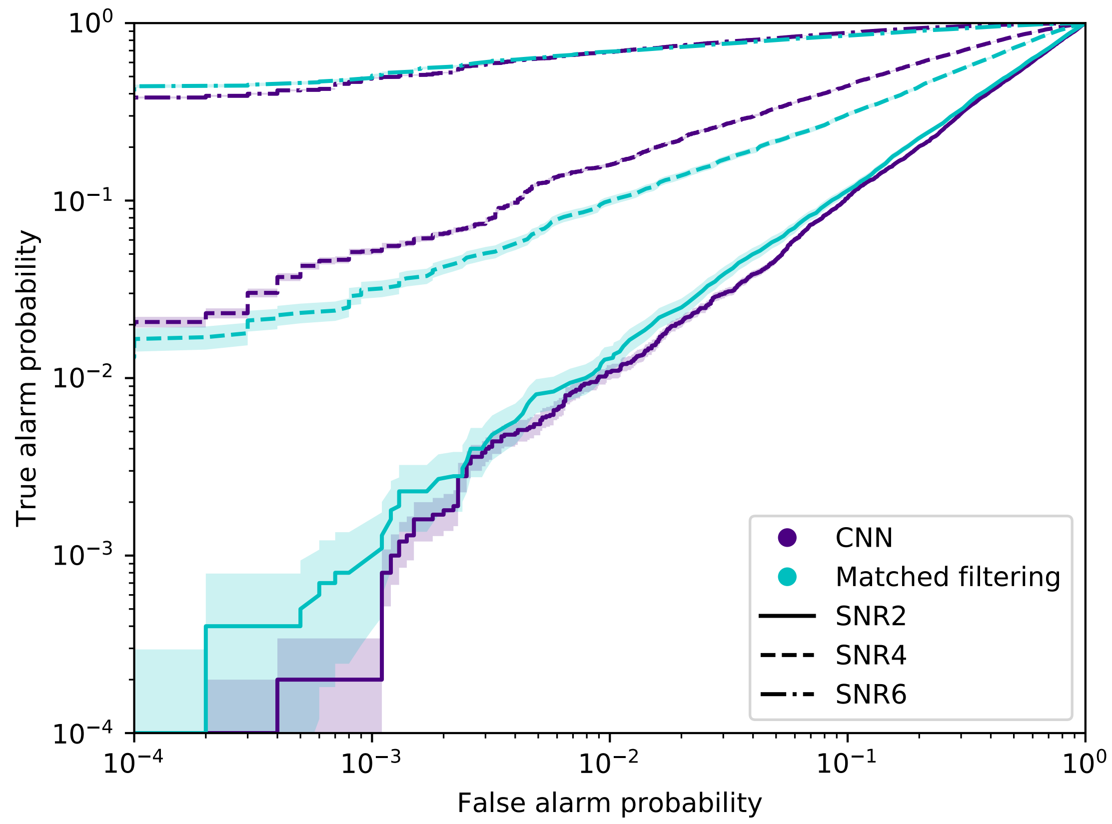
Gabbard *et al* PRL 120, 141103 (2018)

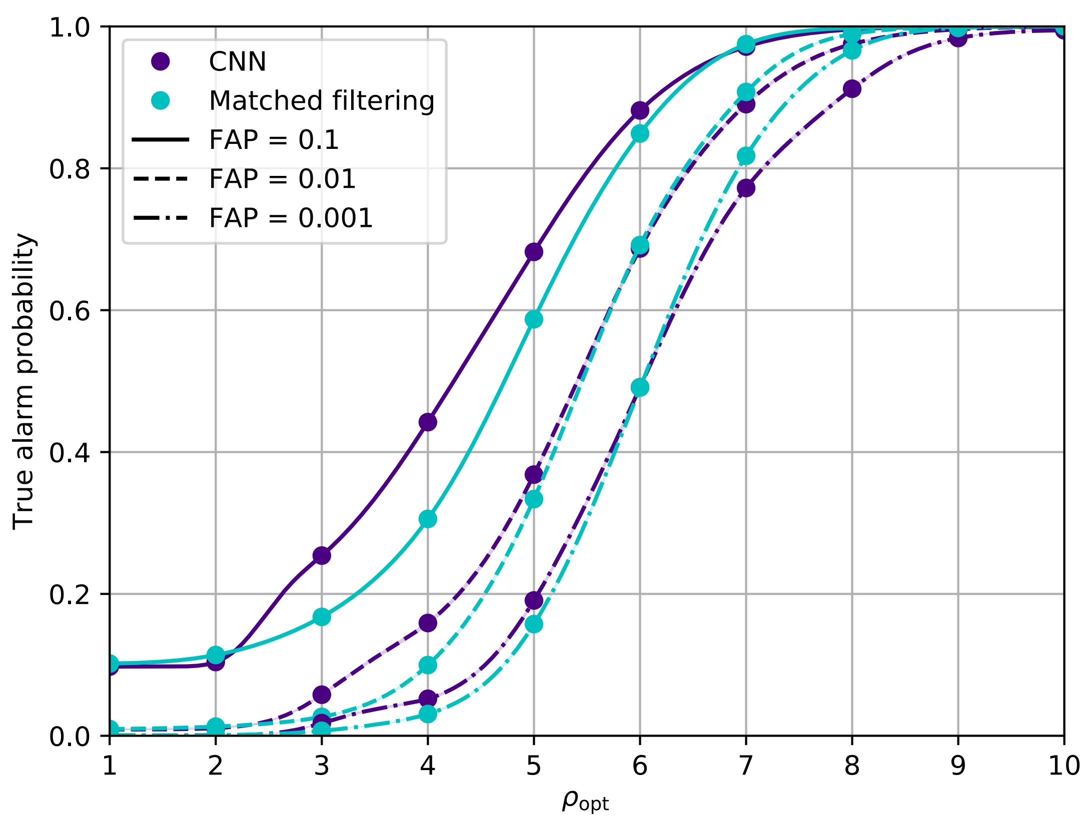
The process

- time-series (not frequency-series, spectrogram or complex spectrogram)
- single detector
- randomised parameters
- same signal - different noise
- transfer learning?
- more data - more sensitivity?

Parameter	Layer								
(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	n/a	n/a	n/a
MaxPool Size	n/a	8	n/a	6	n/a	4	n/a	n/a	n/a
Drop out	0	0	0	0	0	0	0.5	0.5	0
Act. Func.	Elu	Elu	Elu	Elu	Elu	Elu	Elu	Elu	SMax

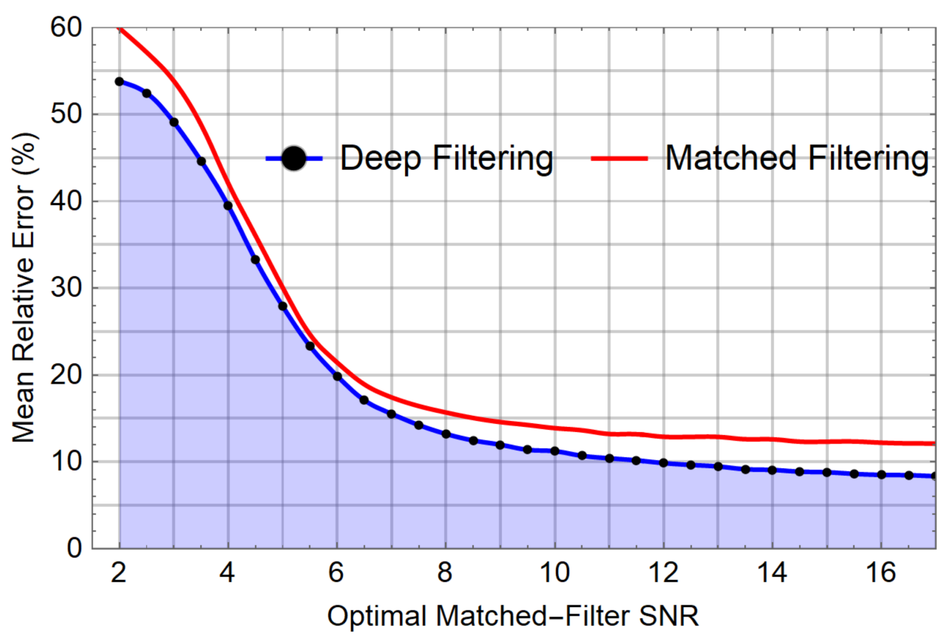
Results



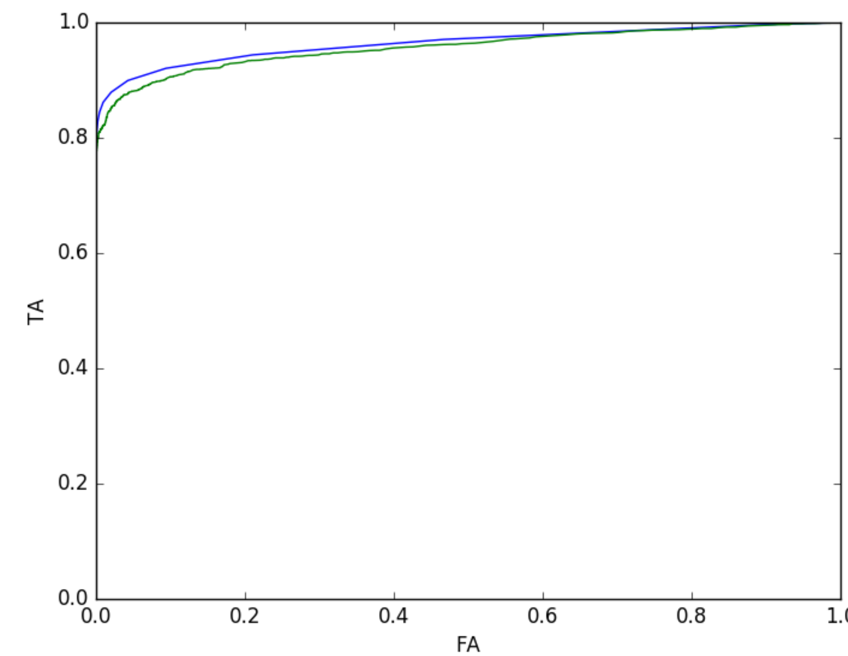


Gabbard *et al* PRL 120, 141103 (2018)

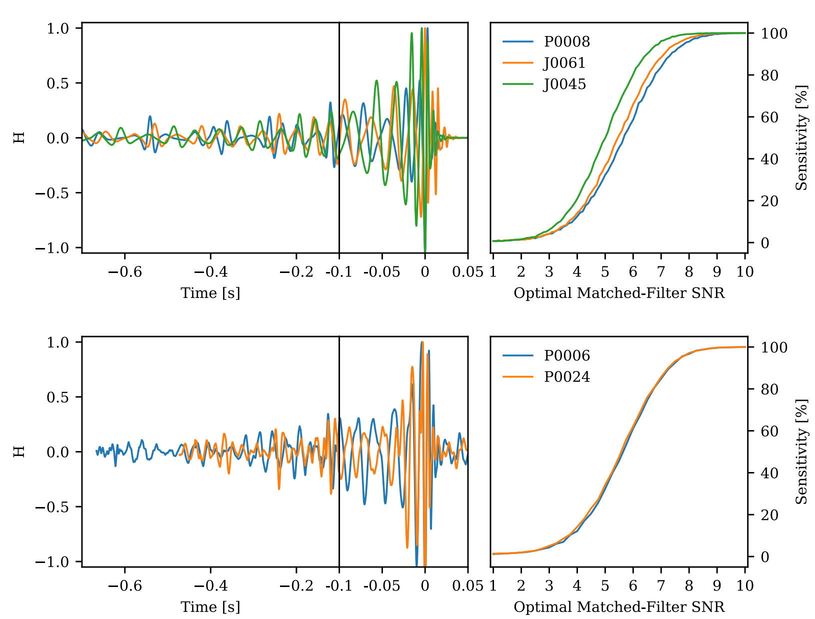
Other work



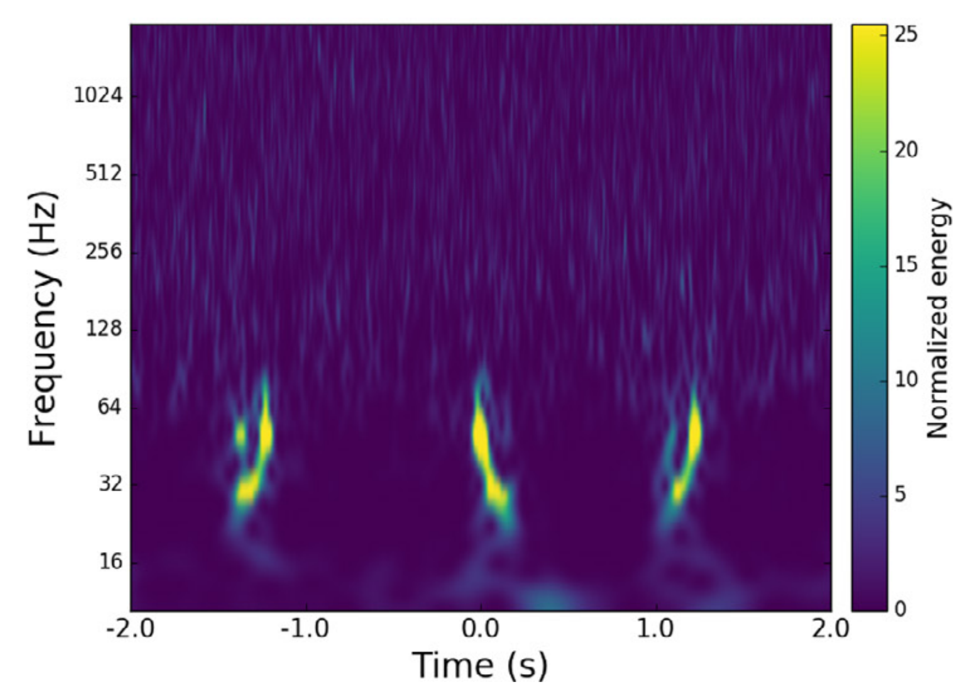
George & Huerta arXiv:1701.00008 (2017)



Williams & Messenger in prep (2018)



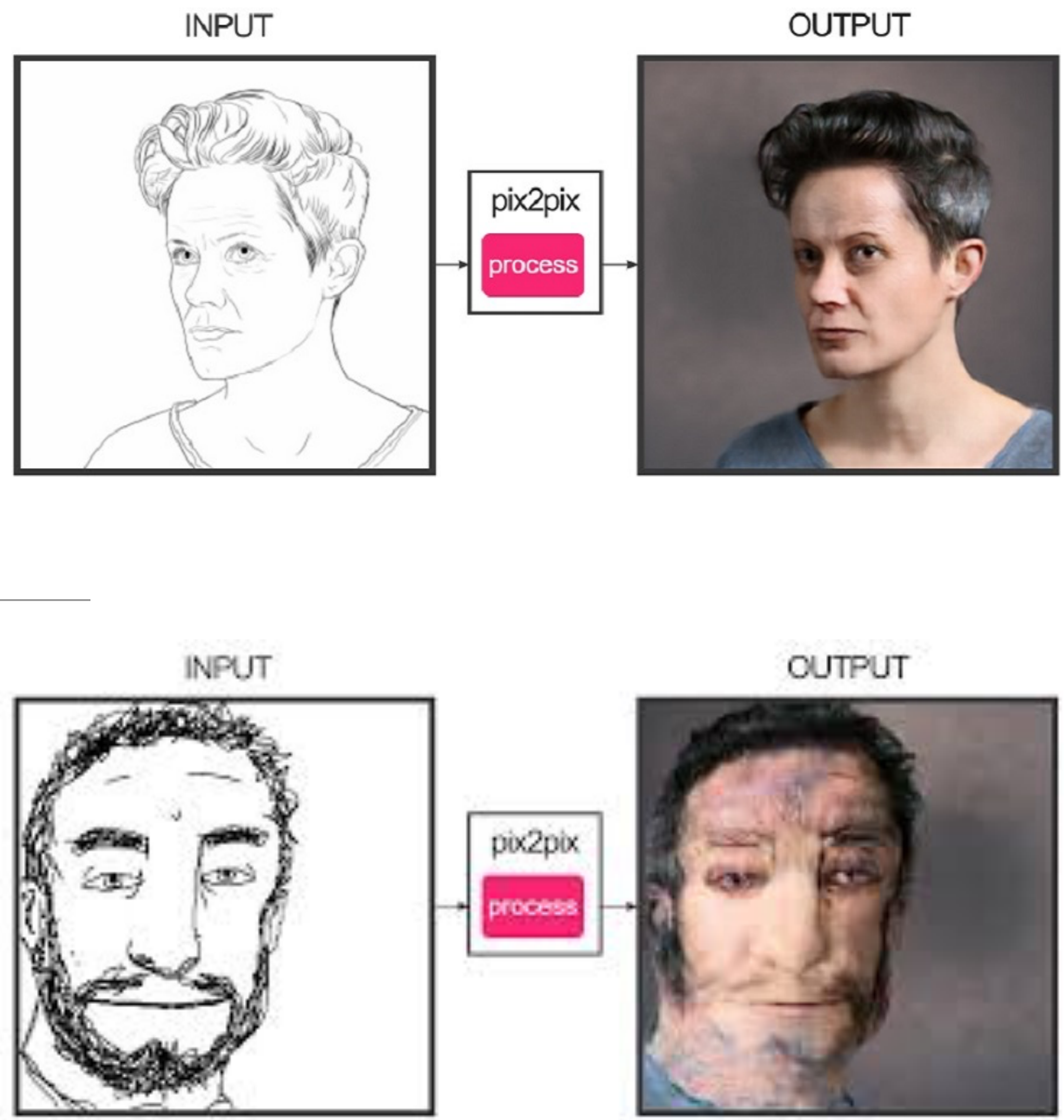
Rebei *et al* arXiv:1807.09787 (2018)



Zevin *et al* CQG, 34, 6, 064003 (2017)

Getting started

You can do this too



Practicalities - Actually getting started

- Find a problem to solve (classification, parameter estimation, generation, …)

- Find a simple network architecture that does better than chance

- Build from there adding complexity in small increments and testing the performance

- Simultaneously build up your training data size

Practicalities - General tips

- Lots of software available (Keras, Tensorflow, Theano, PyTorch, …)
- You are (kind of) wasting your time if you don't have an Nvidia GPU
- Be careful in generating your datasets
- More training data usually means better performance

Practicalities - Training, validation and testing

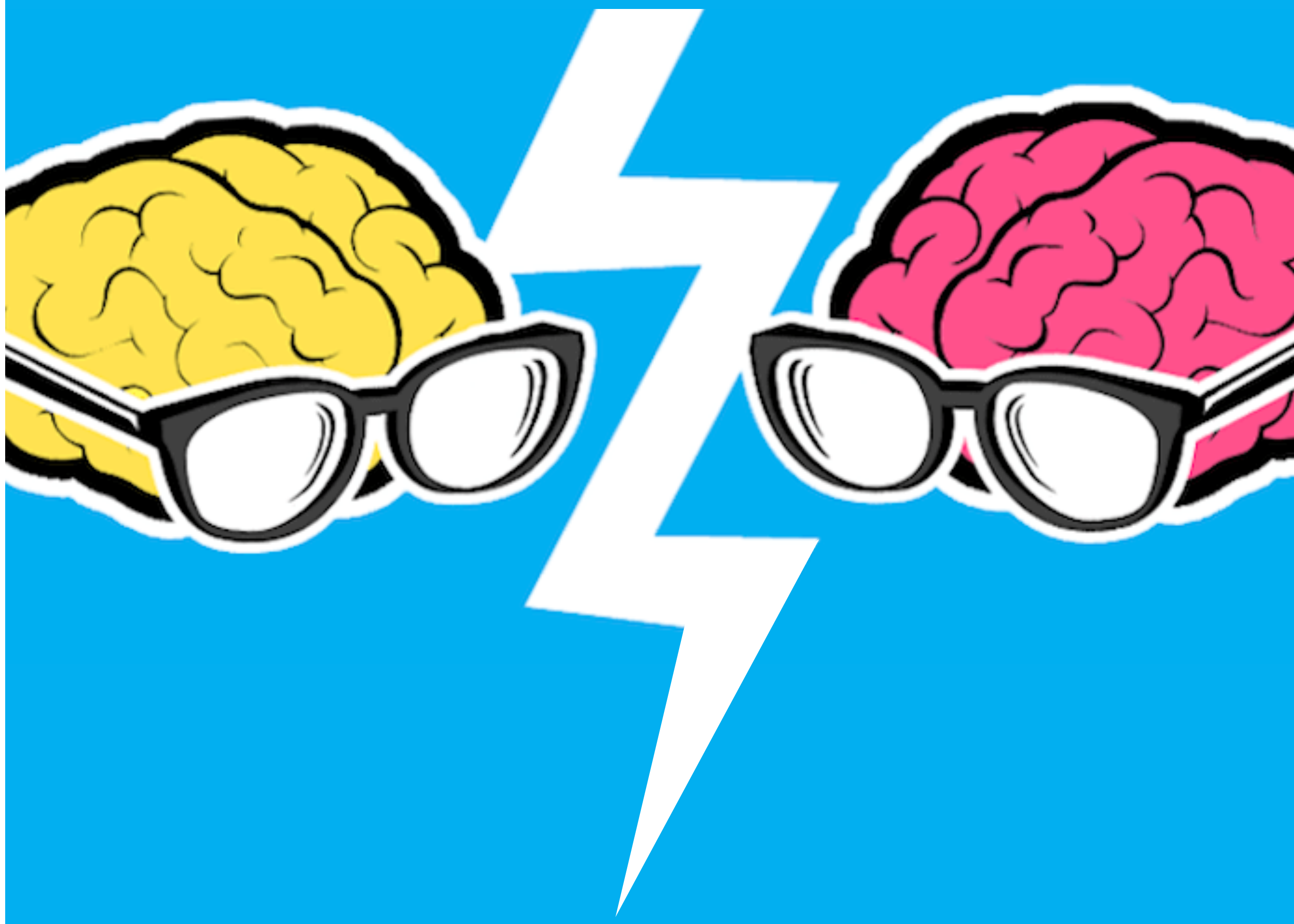
- Your entire dataset is usually divided into 3 groups
  - Training
    - Data used to train the network
  - Validation
    - Data used to check that the network isn't over-fitting
  - Test
    - Data used to quantify the performance of the network

Practicalities - Bells and whistles

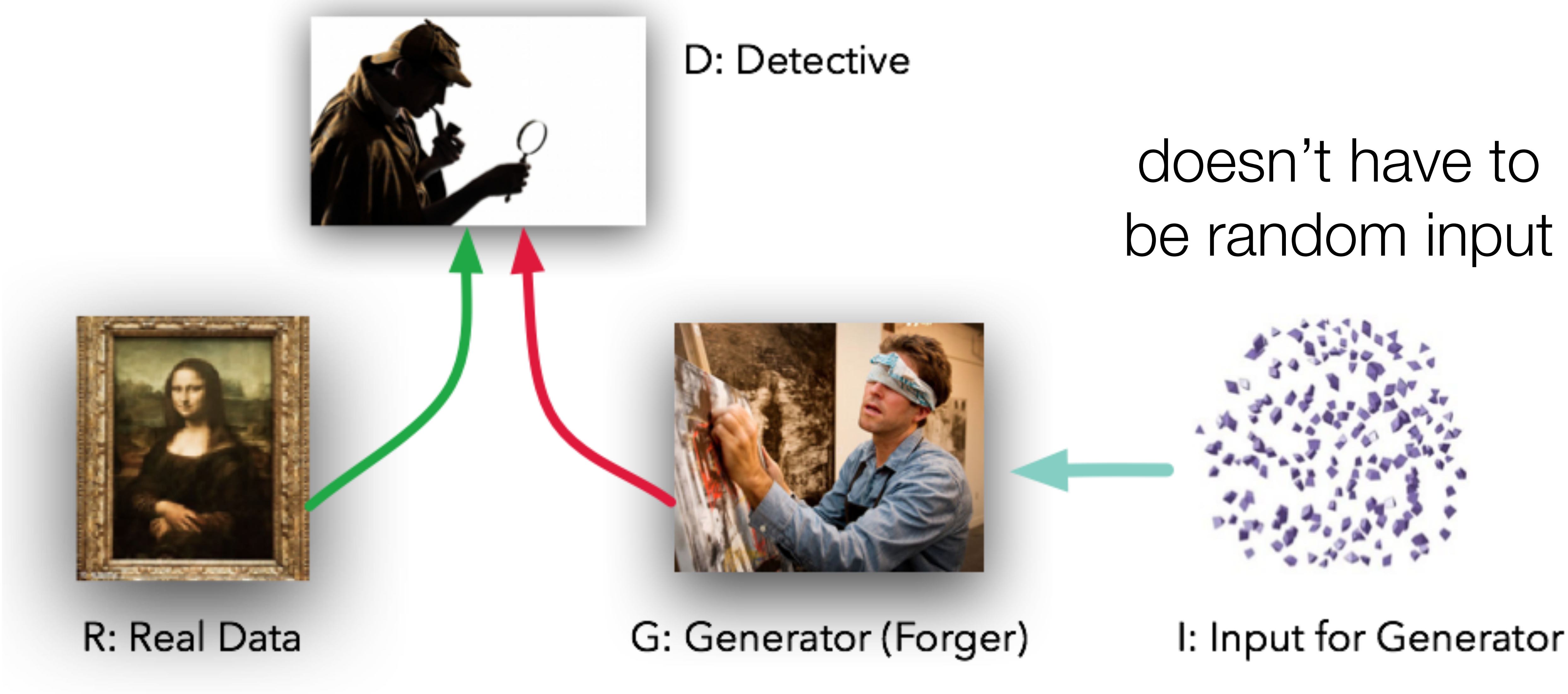
- Max pooling
- Dropout
- Batch normalisation
- Data augmentation
- Transfer learning
- and many more …

Generative Adversarial Networks (GANs)

Getting better through competition



Fighting networks

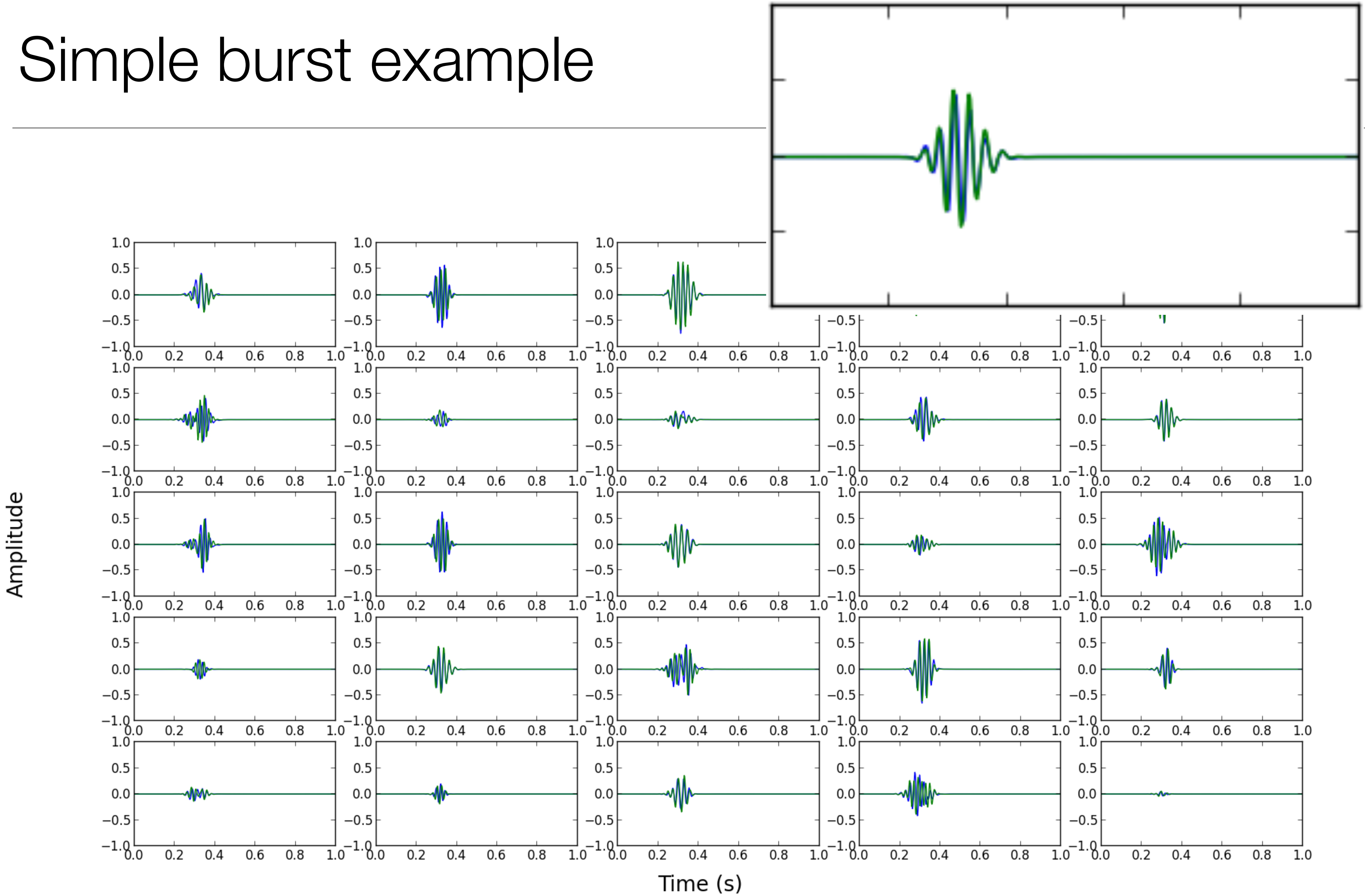


doesn’t have to be random input

Why might GANs be useful?

- In general they are very good at generating new data (hence “generative”)
- They don’t need much training data
- To be honest, for GWs, we’re not sure yet
- Useful for generating fake signals - bursts, NR
- Also useful for distinguishing real from fake - maybe a search algorithm

Simple burst example



McGinn, Heng & Messenger in prep (2018)

Interior design

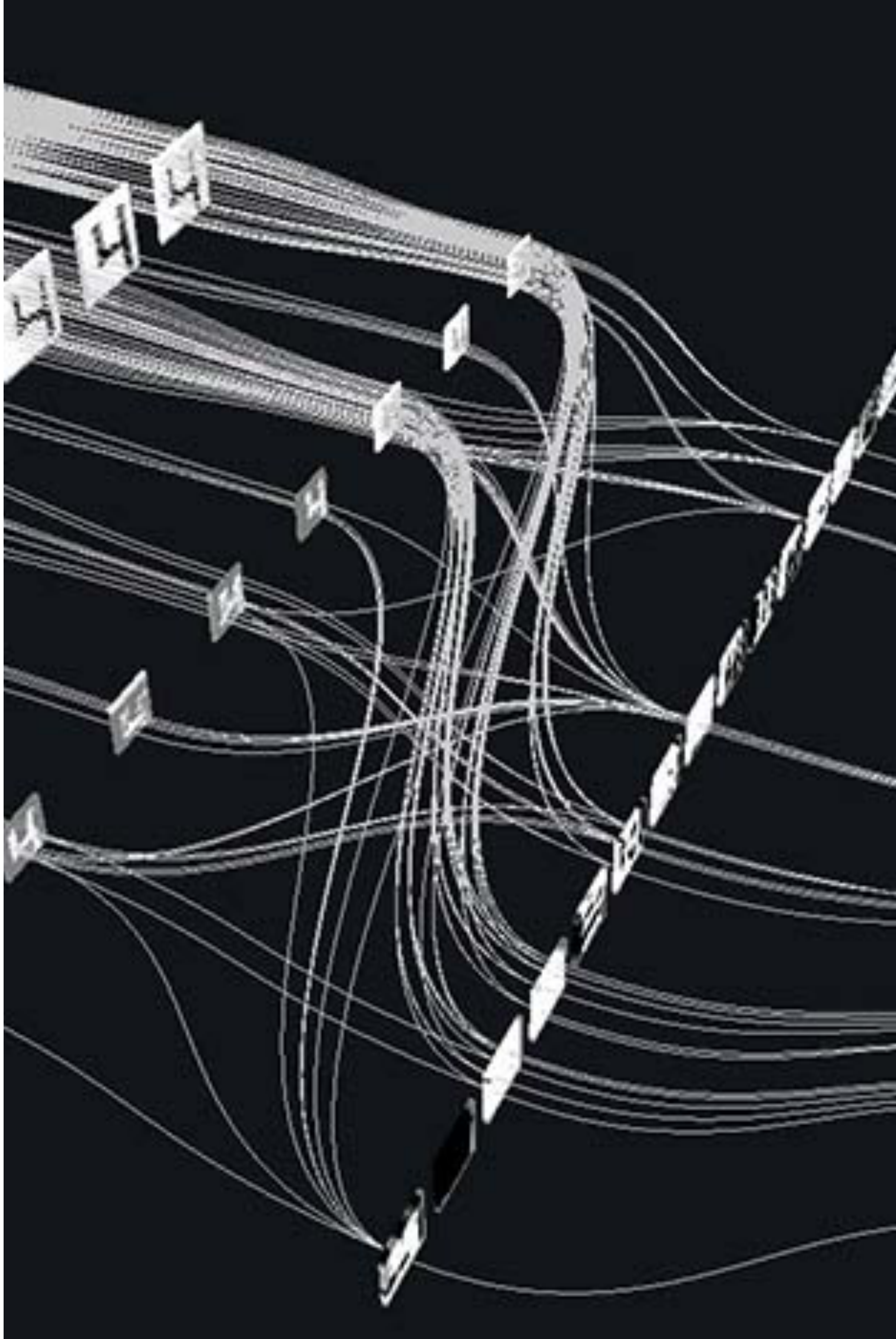
Radford, Metz, Chintala, arXiv:1511.06434 (2015)

Conclusions

Where do we go from here?

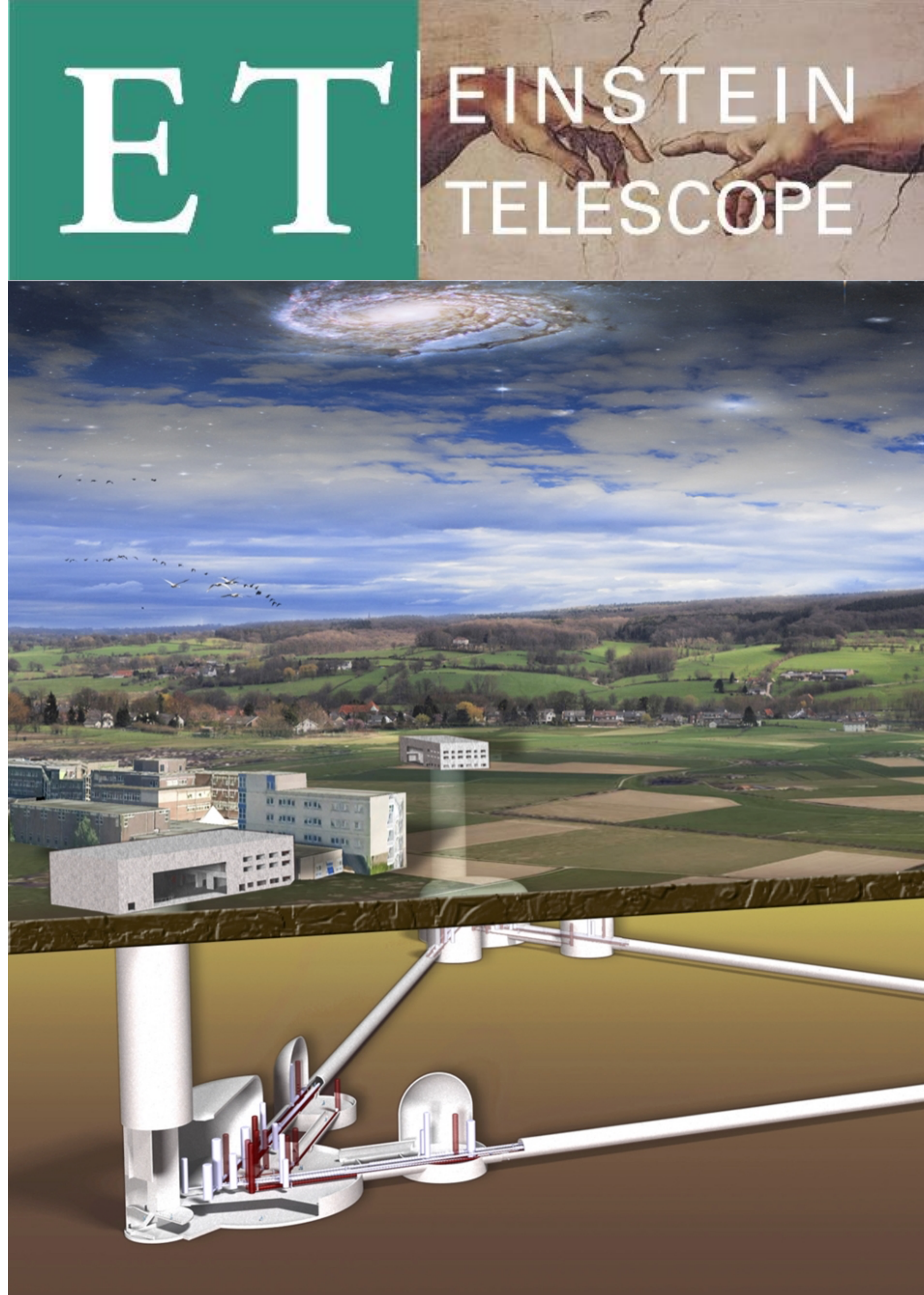
Obvious things

- BBH searches could be done quite simply (now)
- Parameter estimation is possible - but only get point estimate - still very fast
- Push harder for longer waveforms - BNS with LSTM?
- Detector characterisation - requires labelled data (or include in the noise model)
- Start thinking about unsupervised approaches



The future

- Speed
  - Detection and PE of systems prior to merger - EM warning
- Un-modelled
  - Generalised GW transient search - hinged on time and waveform consistent signals
- Non-Gaussian noise
  - Trained on the real data



Thanks