Use Of A **Machine Learning** Algorithm For Real-time Detection Of **Gravitational Waves**

Master 2 PSA internship at IP2I

Outline

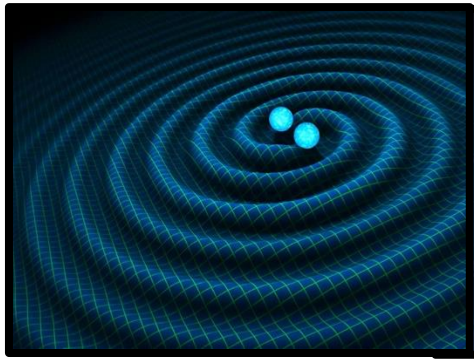
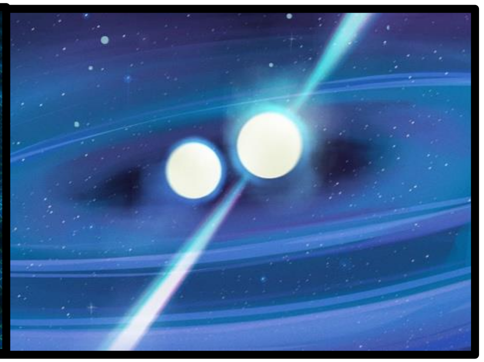
Introduction

- Context
- Goals

Introduction: Context

Gravitational Waves (GW)

- **Prediction** in 1916 by Albert Einstein
- First **indirect-detection** in 1974: Binary Pulsar (*Taylor-Hulse*)
- First **direct-detection** in 2015: LIGO interferometers

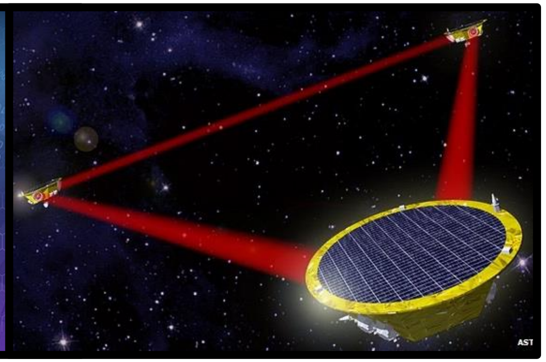



Current detection

- LIGO and Virgo **interferometers**
- Signal processing: **Matched Filtering** → Efficient but time-consuming
- **Increase in sensitivity**: ~1000 signals in Run O4


Future Detection

- **Einstein Telescope** → higher sensitivity, more signals
- **LISA Observatory** in space → signals with low masses
- **Machine Learning** (ML) → Faster detection and exotic signals
- Enable **Multi-Messenger astronomy**

Introduction: Goals

Evaluate the potential of ML for GW Detection

- **Classify** Data in two classes: *Signal+Noise & Noise*
- **Generation** of Data Sets
- **Implementation** of a Convolutional Neural Network (CNN)
- **Optimization**: Increase *sensitivity* and decrease *False Alarm Rate* (FAR)
- Find the **best training**

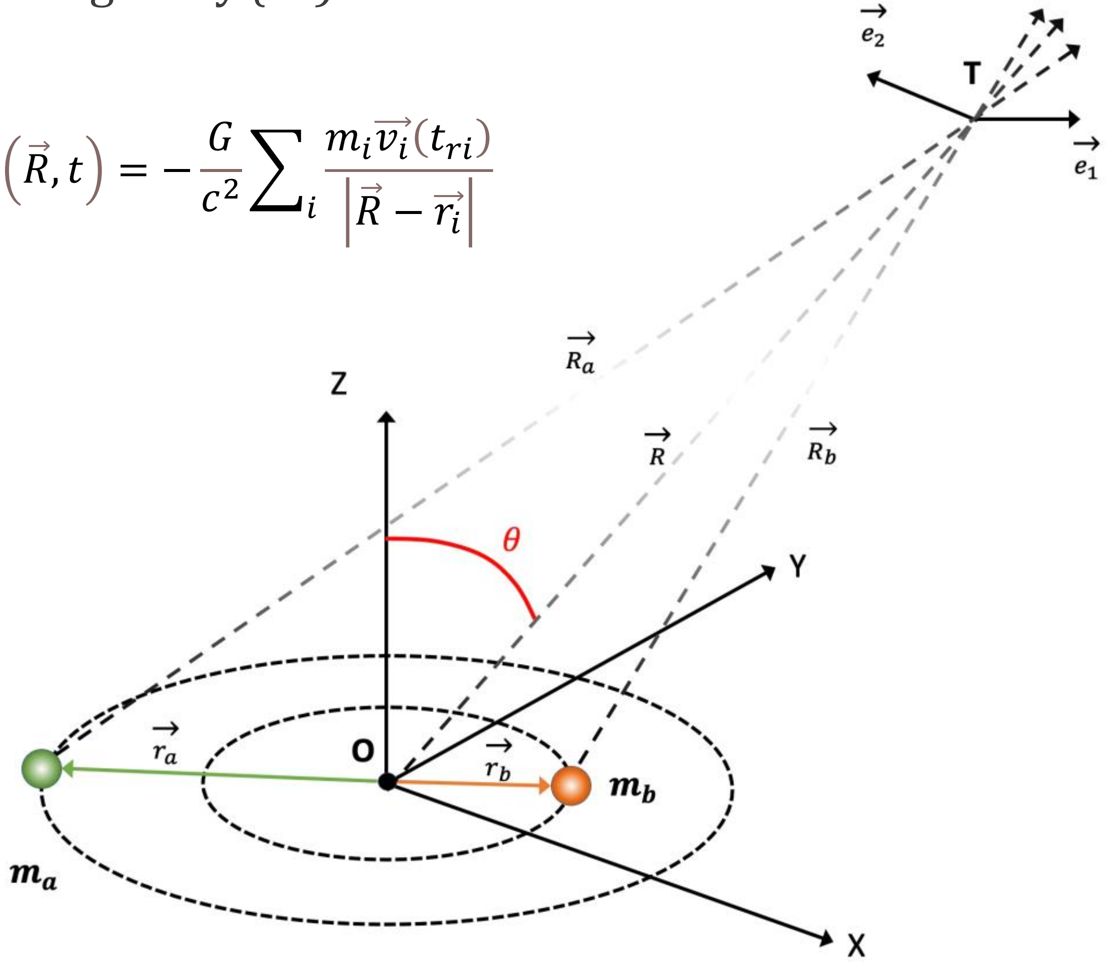
Scientific And Technical Background

- Gravitational Waves
- Interferometers
- Matched Filter
- Machine Learning

Background: Gravitational Waves

Simplified approach: Gravito-electromagnetism

- Assumption by Heaviside 1893: **Finite propagation speed** for gravity *(=c)* → Weak fields
- Scalar and vector **potentials** $\Phi_g\left(\vec{R},t\right) = -G\sum_i \frac{m_i}{\left|\vec{R}-\vec{r_i}\right|}$ $\quad \vec{A_g}\left(\vec{R},t\right) = -\frac{G}{c^2}\sum_i \frac{m_i\vec{v_i}(t_{ri})}{\left|\vec{R}-\vec{r_i}\right|}$
- Corresponding **fields** $\vec{E_g} = -\vec{\nabla}\Phi_g - \frac{\partial\vec{A_g}}{\partial t}$ $\quad \vec{B_g} = \vec{\nabla}\wedge\vec{A_g}$
- In the vacuum **propagation of waves** $\vec{\nabla^2}\vec{E_g} - \frac{1}{c^2}\frac{\partial^2\vec{E_g}}{\partial t^2} = 0$
- **Gravitational Force** $\vec{F_g} = m\left(\vec{E_g} + \vec{v}\wedge 4\vec{B_g}\right)$



Background: Gravitational Waves

Binary system

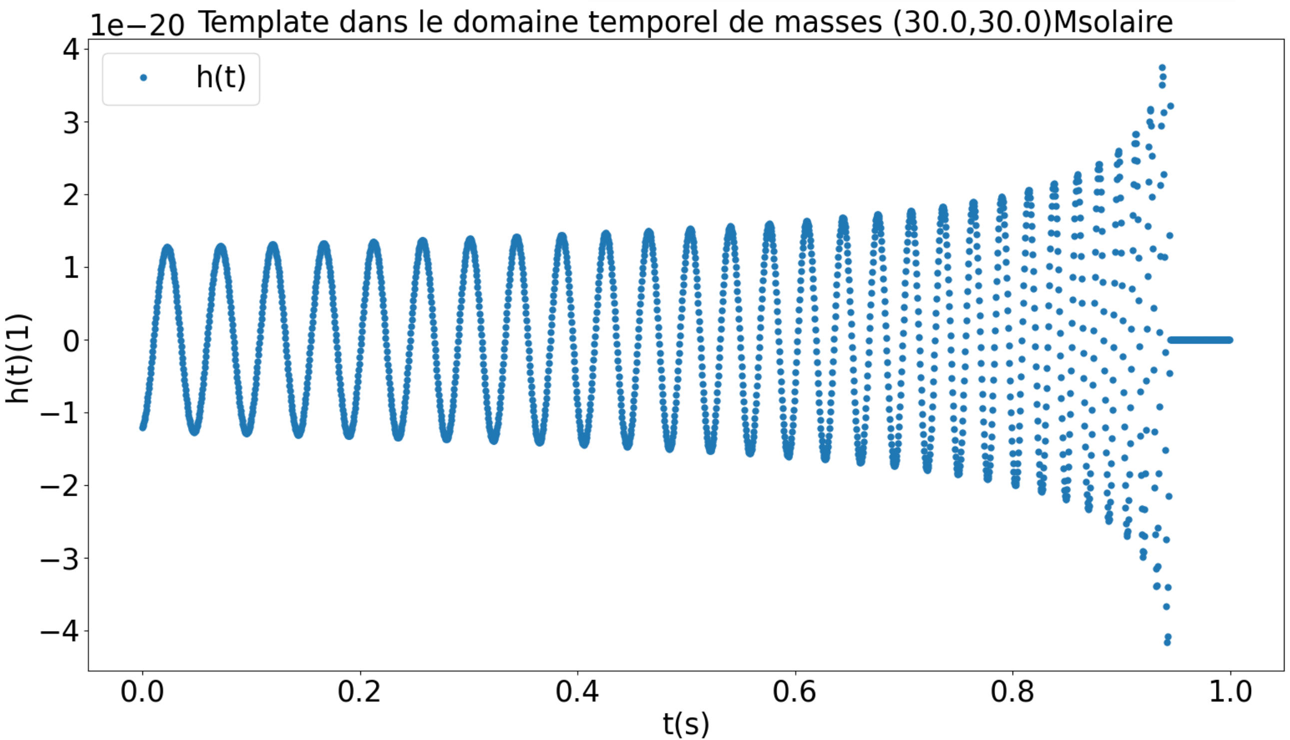
- **Newtonian Approximation:** $v_{a,b} \ll c$ and slightly distorted Minkowski metric
- System **losing Energy** $$\frac{dE}{dt} = -P$$
- **Pulsation evolution**

$$\omega(t) = \left(\frac{125}{128 \times N^3}\right)^{1/8} t_{SC}^{-5/8}(t_c - t)^{-3/8}$$

- **Gravitational Wave signal** in the detector

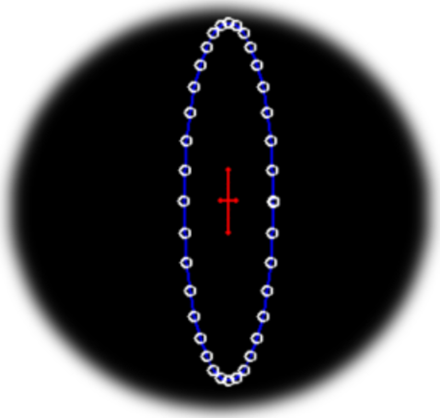
$$h(t) = \frac{\eta(GM)^{5/3}\omega^{2/3}(t)}{4Rc^4}\cos 2\Phi(t)$$

$$\Phi(t) - \left(\frac{2}{5}\right)^{5/8}\left(\frac{t_c - t}{t_{SC}}\right)^{5/8} + \Phi_c$$

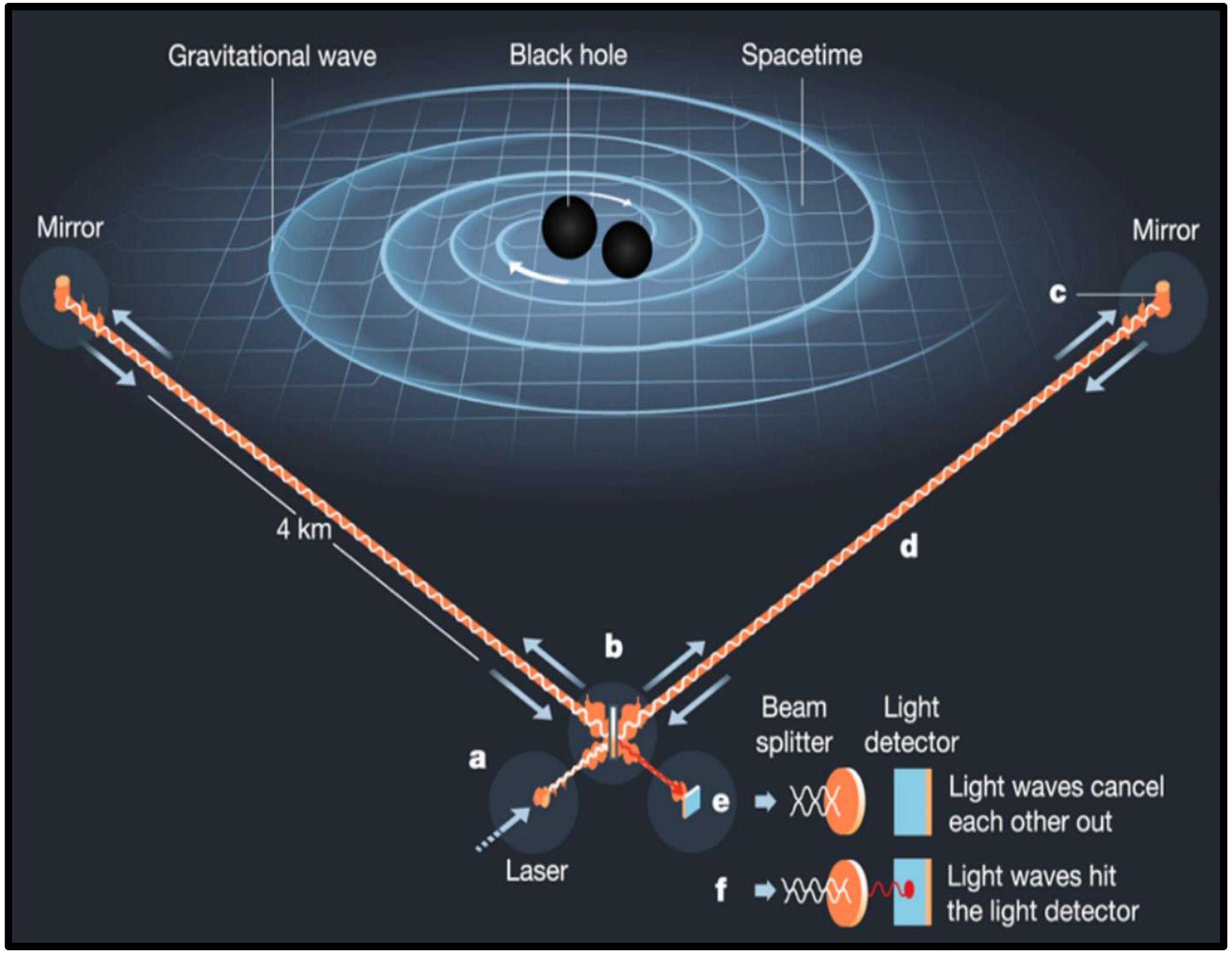


Background: Interferometer

- Effect on a **mass system**
- Gertsenshtein and Pustovoit **interferometer idea** in 1962
- Signal **very weak** *(proton size)*: A lot of noise

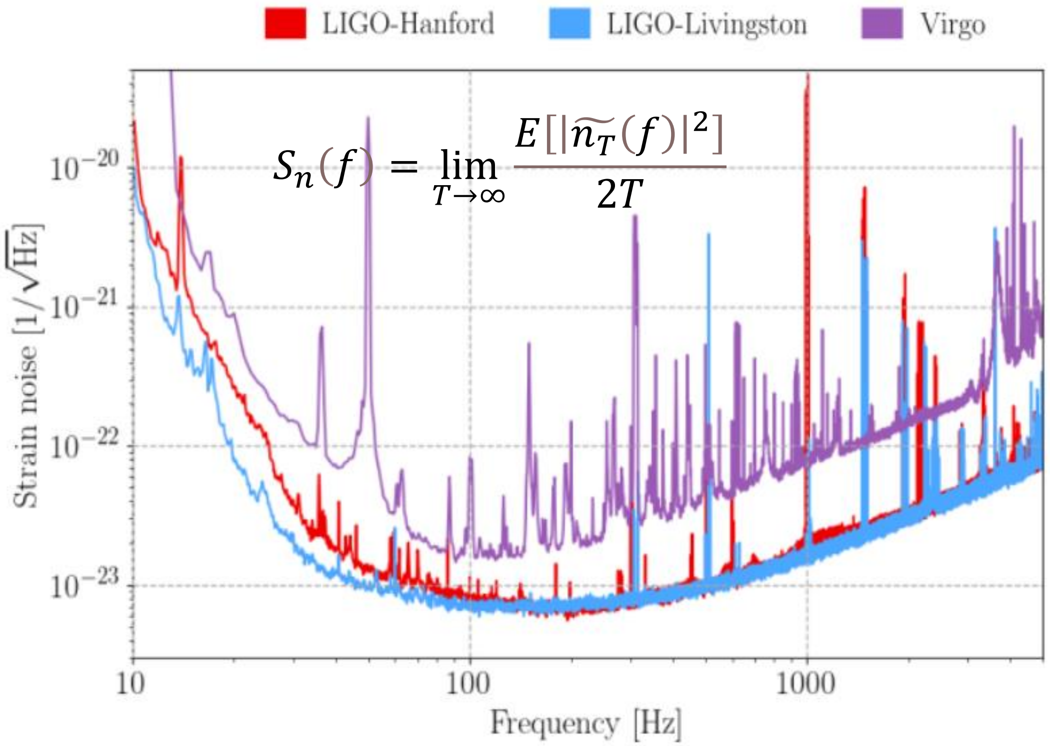


$$h(t) = \frac{\delta L}{L} \simeq 10^{-22}$$
$$\Leftrightarrow \delta L \simeq 10^{-18} m$$



Background: Matched Filter

- **Data layout** $s(t) = h(t) + n(t)$
- **Matched filter** $\tilde{g}(f) = 2\frac{\widetilde{h^*}(f)}{S_n(f)}$



$$S_n(f) = \lim_{T\to\infty} \frac{E[|\widetilde{n_T}(f)|^2]}{2T}$$

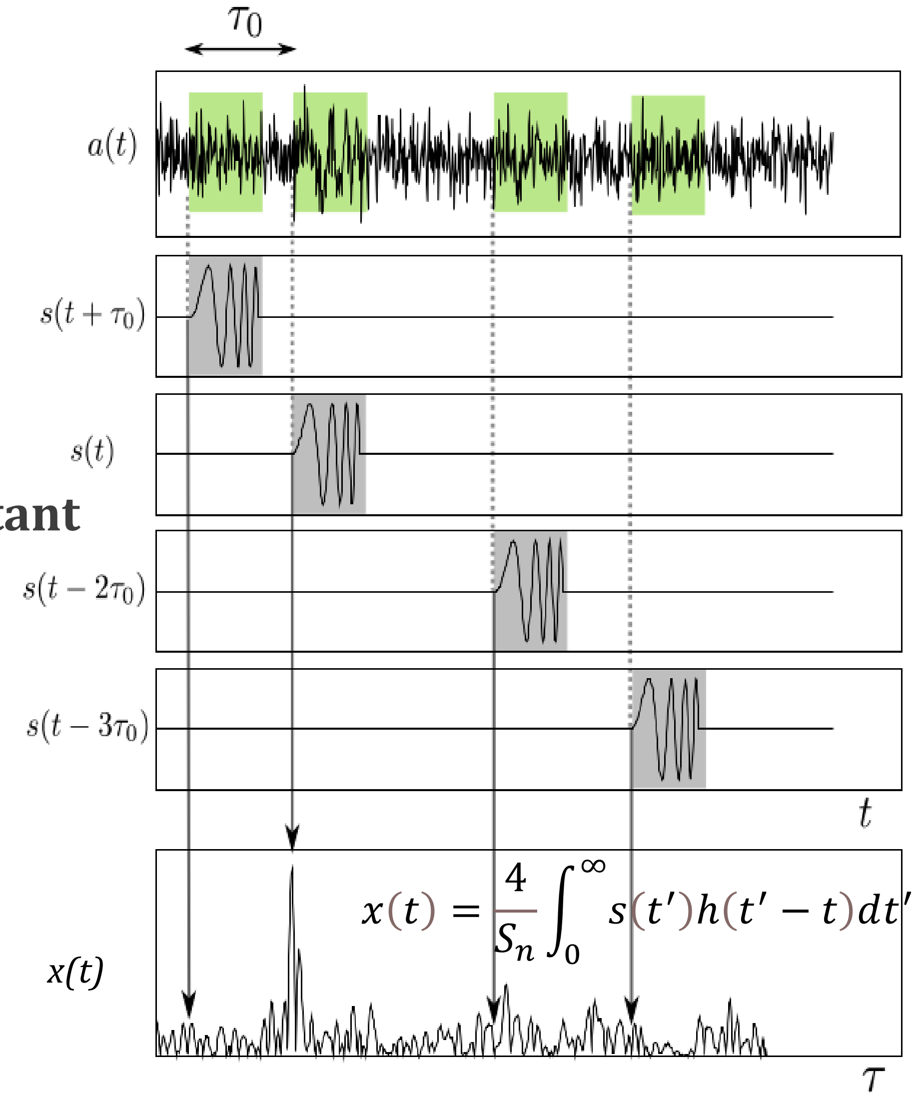
- **Filtered signal**

$$x(t) = \int_{-\infty}^{+\infty} \tilde{s}(f)\tilde{g}(f)e^{+2i\pi ft}df$$

- **Normalization constant** (Optimal SNR)

$$\rho_{opt}^2 = \langle x(t)x^*(t)\rangle = 4\int_0^{\infty} \frac{|\tilde{h}(f)|^2}{S_n(f)}df$$

$$\rho(t) = \frac{|x(t)|}{\rho_{opt}}$$

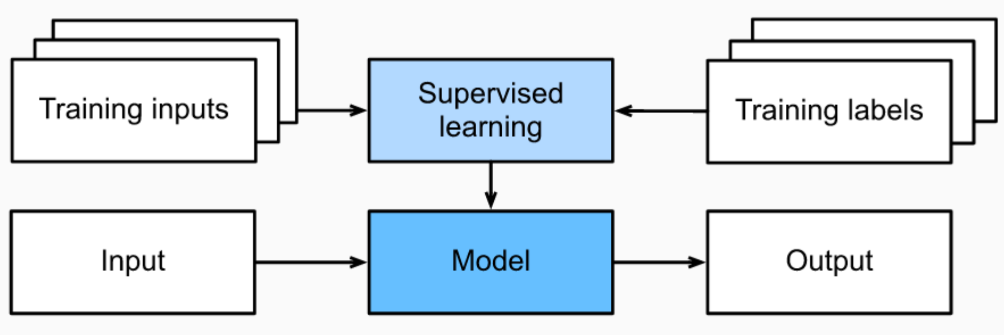


$$x(t) = \frac{4}{S_n}\int_0^{\infty} s(t')h(t'-t)dt'$$

Background: Machine Learning

Regression

- **Supervised learning** principle



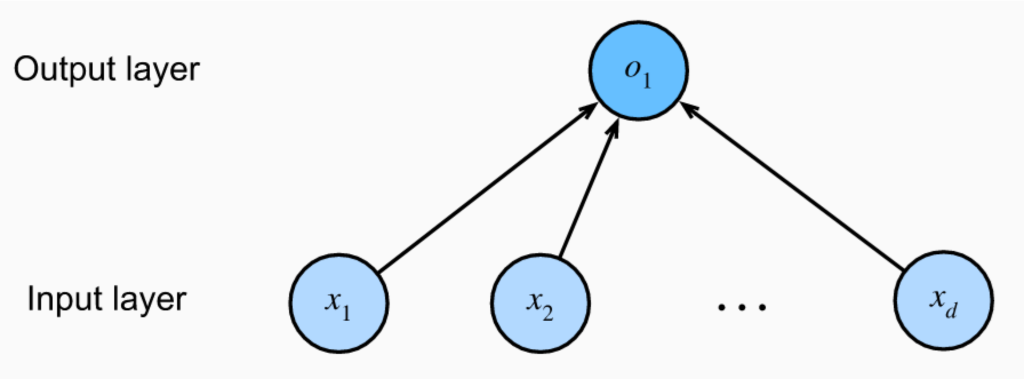
- **Linear regression** $\hat{y} = Xw + b$

- **Root mean square error** to be minimized $L(w,b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(w,b)$ with $l^{(i)}(w,b) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$

- **Analytical solution**: Normal equations $w^* = (X^\top X)^{-1} X^\top y$

- **Stochastic gradient descent** algorithm

$$(w,b) \leftarrow (w,b) - \frac{\eta}{|\mathcal{B}_k|}\sum_{i\in\mathcal{B}_k} \partial_{(w,b)} l^{(i)}(w,b)$$

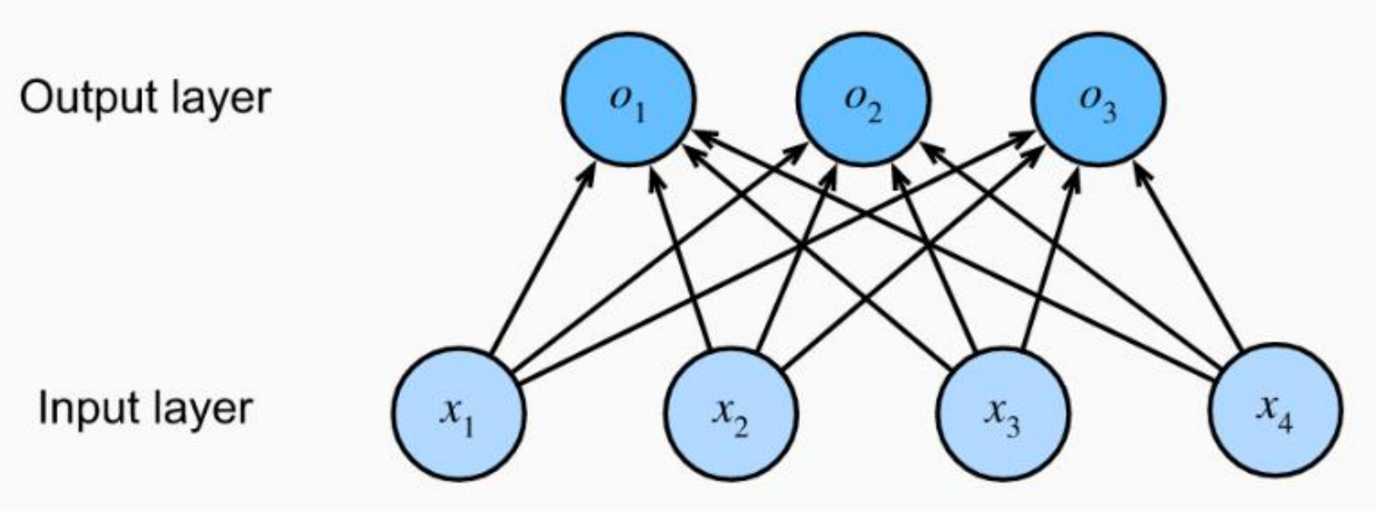


Background: Machine Learning

Classification

- **Classification model**: several parallel Linear Regressions

$$\hat{O} = XW + b \qquad \hat{Y} = softmax(\hat{O})$$



- **Softmax operation**: membership probability

$$\hat{y} = softmax(\hat{o}) \qquad \widehat{y_k} = \frac{\exp(o_k)}{\sum_i \exp(o_i)}$$

- Maximization of the **likelihood**

$$P(Y|X) = \prod_{i=1}^{n} P(y^{(i)}|x^{(i)})$$

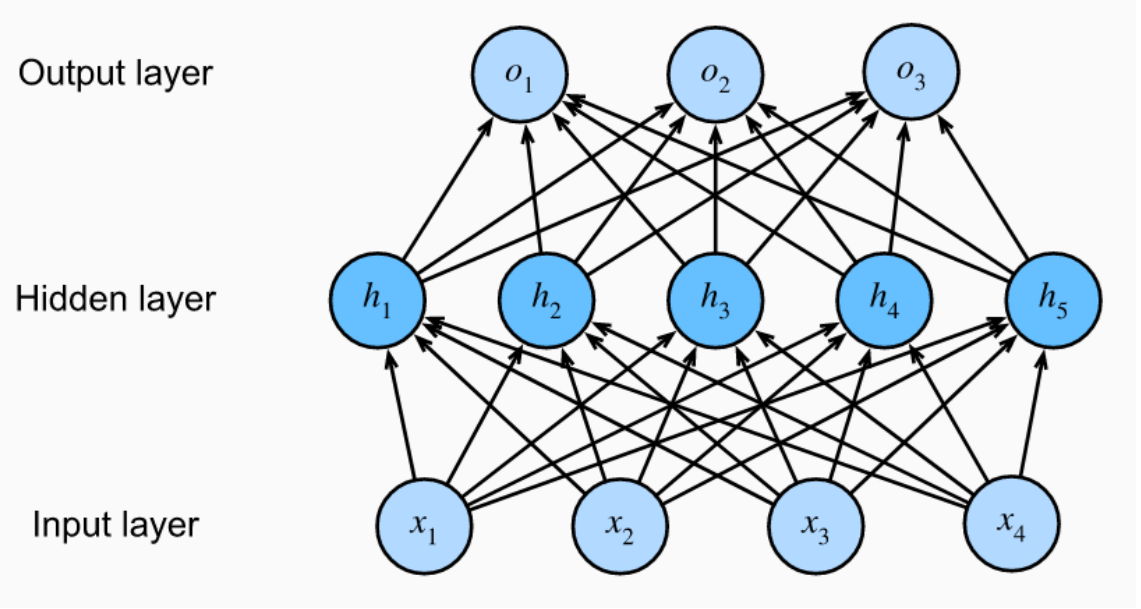
Background: Machine Learning

Multilayers Perceptrons

- **Multi-layered model** adding intermediate layers

$$\hat{H} = \sigma\left(XW^{(1)} + b^{(1)}\right) \qquad \hat{O} = \hat{H}W^{(2)} + b^{(2)}$$

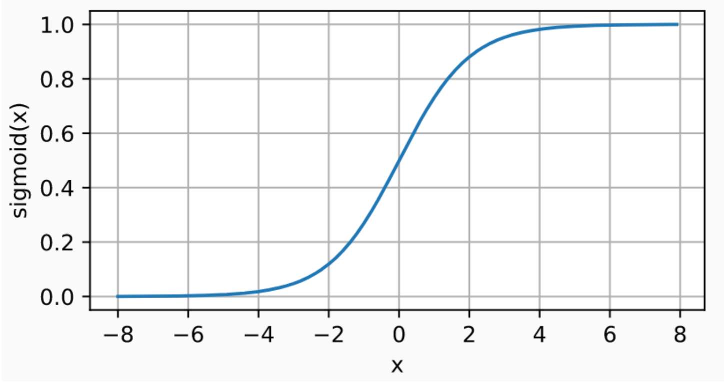
- **Explosion** of the **number of parameters**

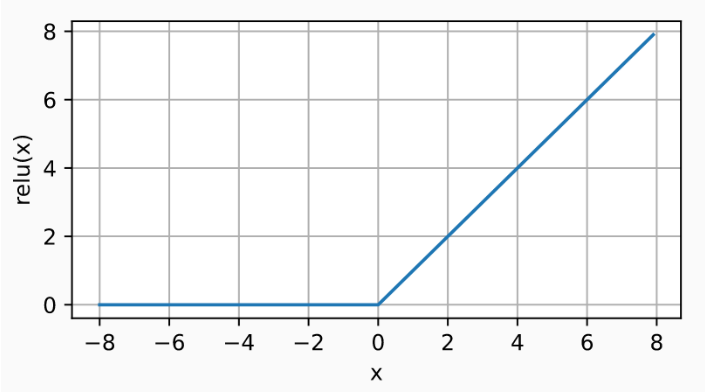


- Need to add **non-linear activation functions**

$$sigmoid(x) = \frac{1}{1 + \exp(-x)}$$

$$ReLU(x) = \max(0, x)$$





Background: Machine Learning

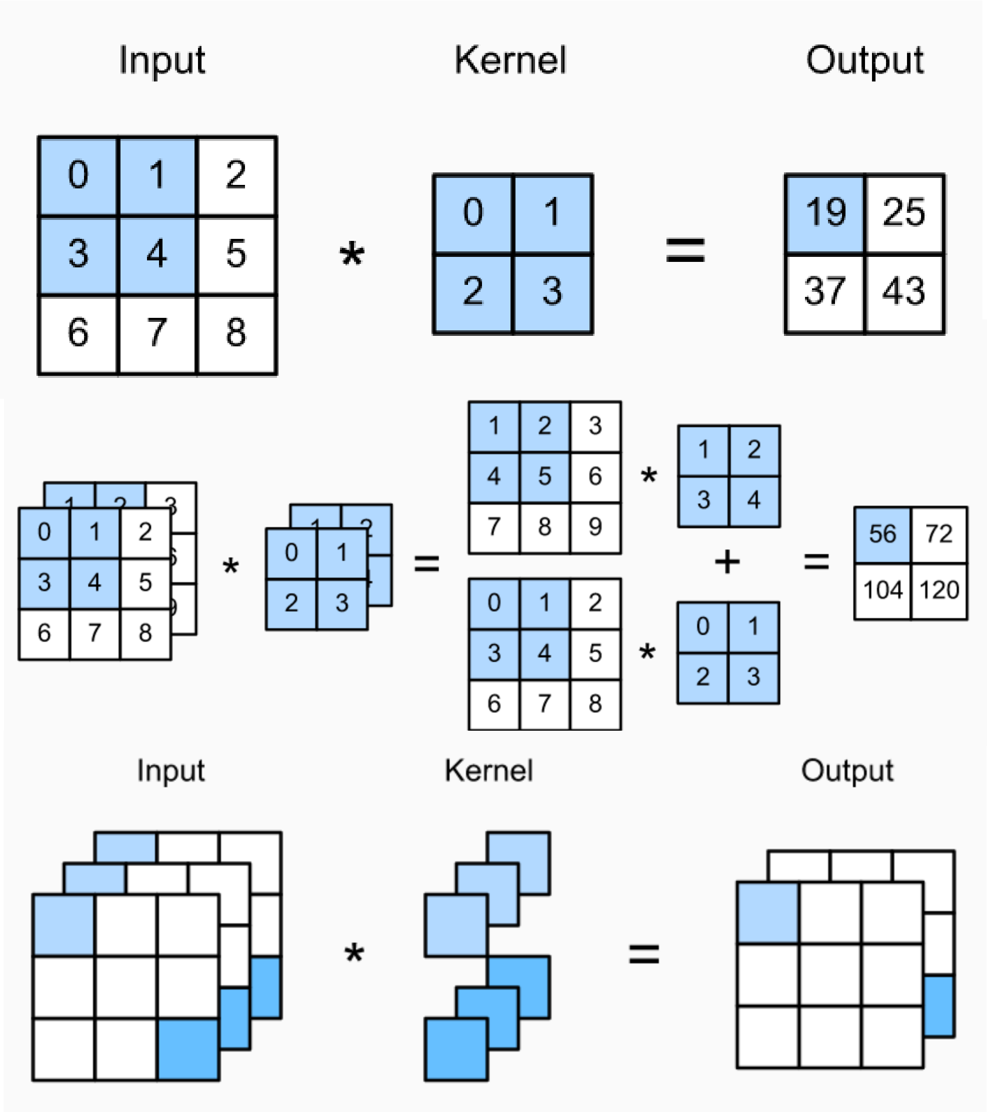
Convolution

- Principle of convolution: Taking into account the **correlation between the data** (temporal and/or spatial)

→ **reduce** the number of **parameters**

$$[H]_{k,l,s} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \sum_{i} [V]_{a,b,i,s} [X]_{k+a,l+b,i}$$

- **Multiple inputs**: choice of the number of kernels and the size
- **Multiple outputs**: As in multilayer networks
- **Pooling** layer



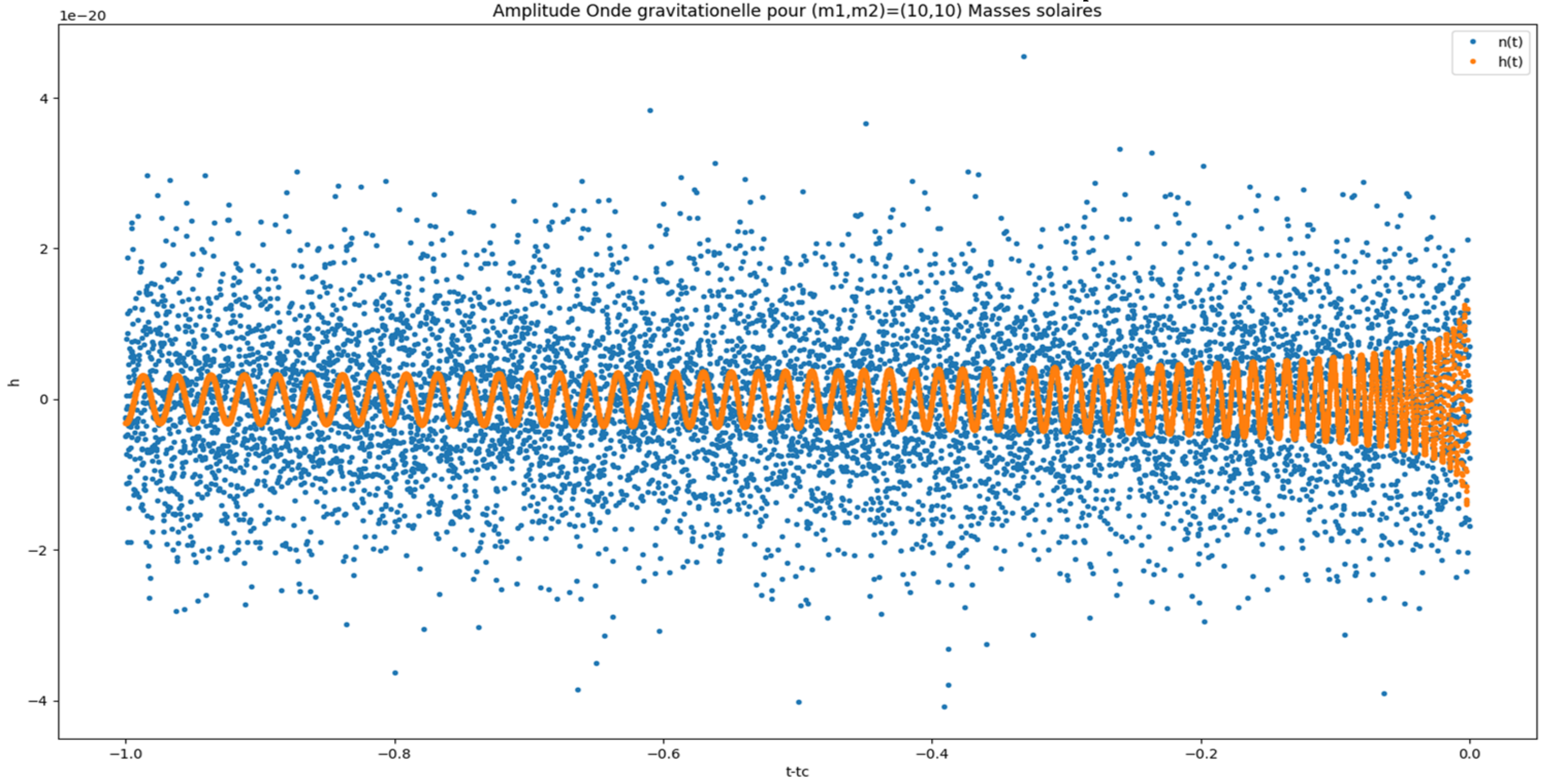
Data Generation

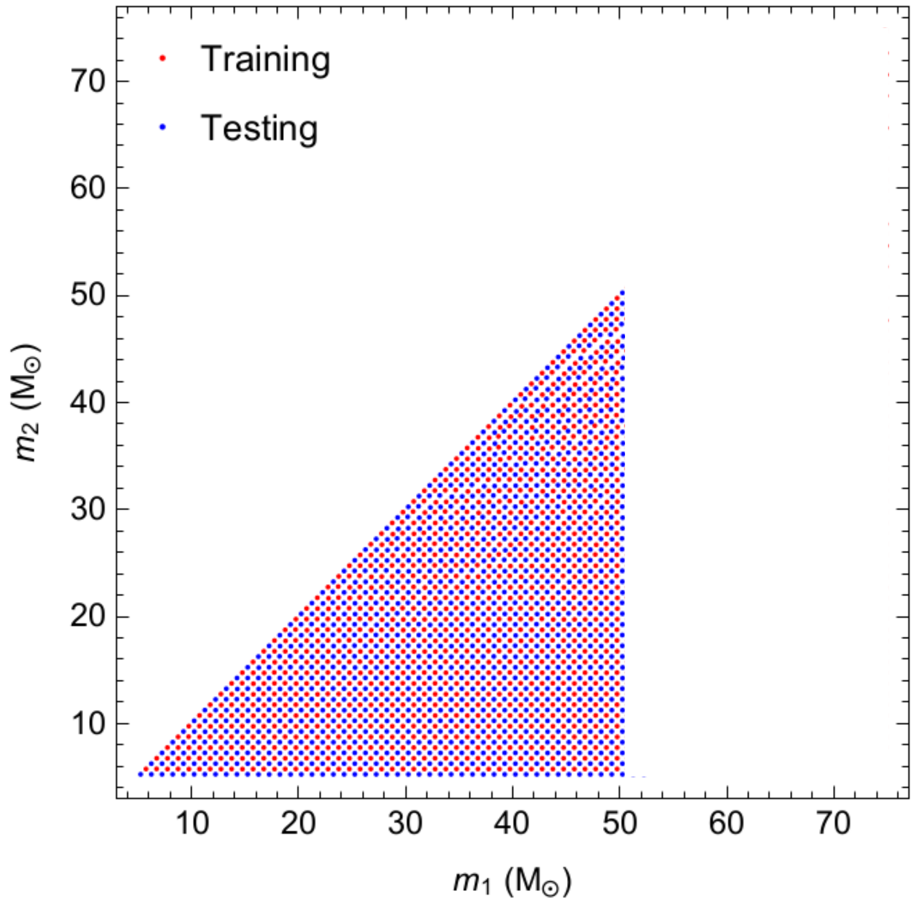
- First Data
- Whitening and colored noise

Data Generation: First Data

- **White** Gaussian Noise
- **1s signals** with $m_1 \geq m_2$
- **Template Normalization** $h_{norm}(t) = \frac{h(t)}{\rho_{opt}}$

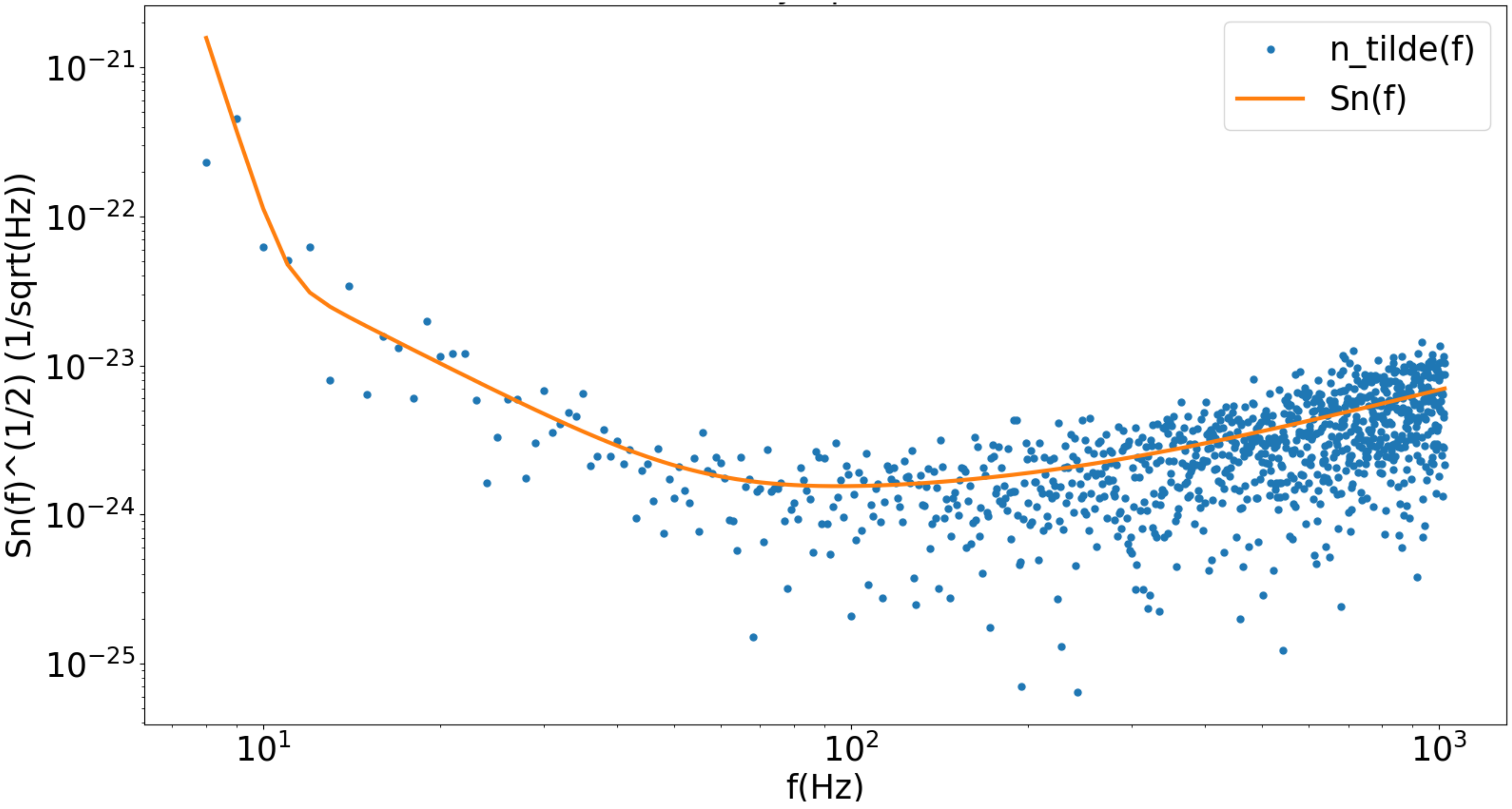
f_e	T_{tot}	D	t_c	Φ_c	f_{Dmin}	$Nb_B/template$	$Masses$
$2048\ Hz$	$1s$	$1Mpc$	$[.75,.95]s$	0	$20Hz$	10	$[\|10,50\|]\ M_\odot$





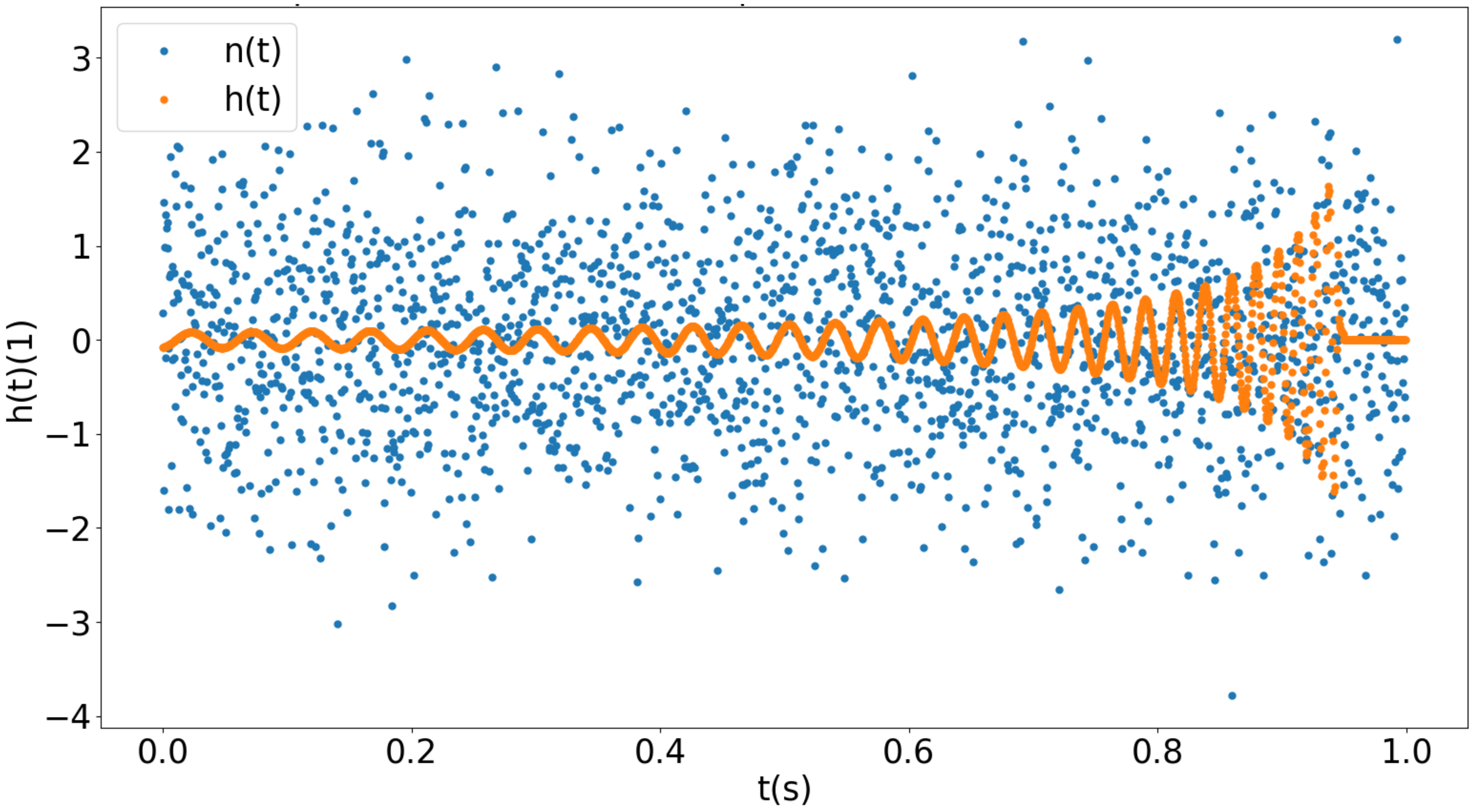
Data Generation: Whitening and colored Noise

- **Whitening** to scale Data around unity
- **Colored** Gaussian Noise
- **Analytic PSD**
- Change in **signal shape**



Data Generation: Whitening and colored Noise

- **Whitening** to scale Data around unity
- **Colored** Gaussian Noise
- **Analytic PSD**
- Change in **signal shape**

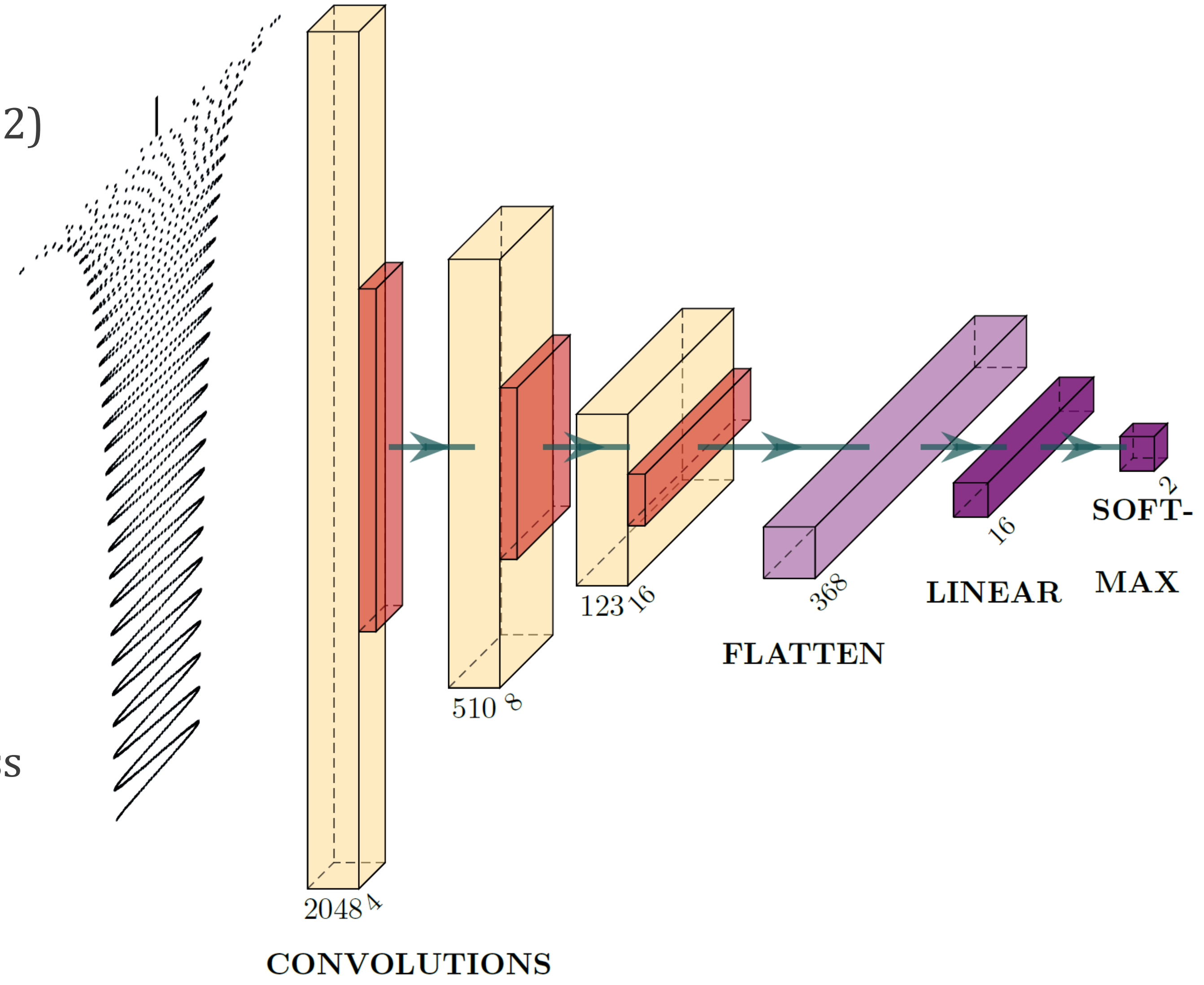


Neural Network And Training

- Neural Network
- Hyper-parameters and SNR

NN And Training : Neural Network

- **Convolution layers** (kernel sizes=8,16,32)
- **Pooling layers** (kernel sizes=4)
- **Activation functions**: ReLU
- **Stochastic Gradient Descent**
- **Loss function**: SoftmaxCrossEntropyLoss



NN And Training : Hyper-parameters and SNR

- Two **kinds of training** and **hyper-parameters** to optimize

	Fix SNR		Decreasing SNR	
	Scalar	Interval	Scalar	Interval
ρ_{opt}	8	(10-6)	[36,24,16,12,8]	[(48-24),(36-16),(24-12),(16-8),(10-6)]
Epochs Table	[0,300]		[0,4,8,20,40,300]	
η	3.10^{-3}			
$\lvert B_k \rvert$	250			
$Nb_B/template$	10			
Kind of PSD	*flat*			

Results and Analysis

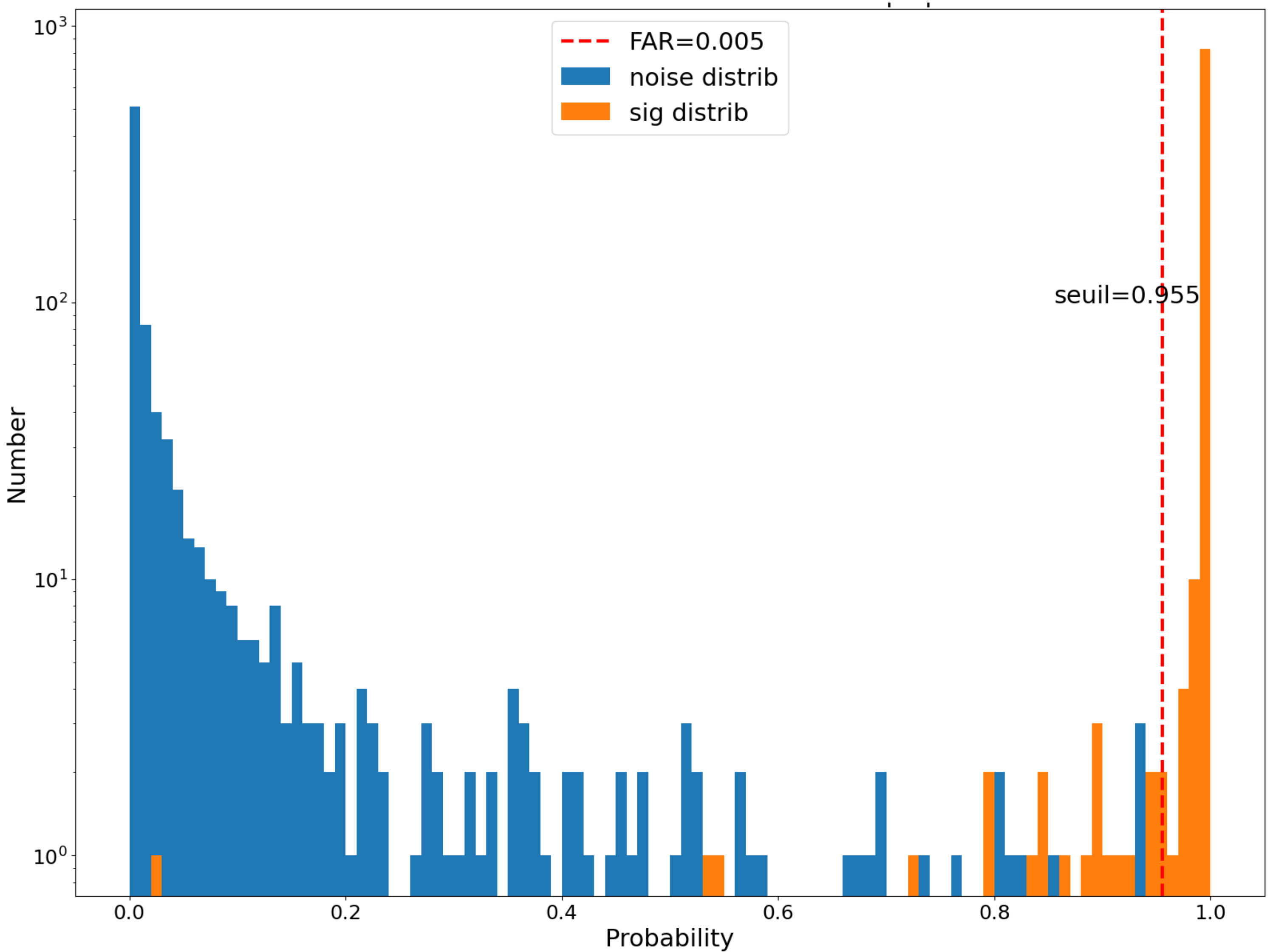
- Definitions and Metrics
- Example of training
- Training SNR Influence
- Learning-rate Influence
- Kind of Training Influence

Results and Analysis: Definitions and Metrics

- **Accuracy**: % of correct classification
- **Sensitivity**: % of correct classification among *Signal+Noise* sample
- **False Alarm Rate** (FAR): % of wrong classification among *Noise* sample
- **Threshold**

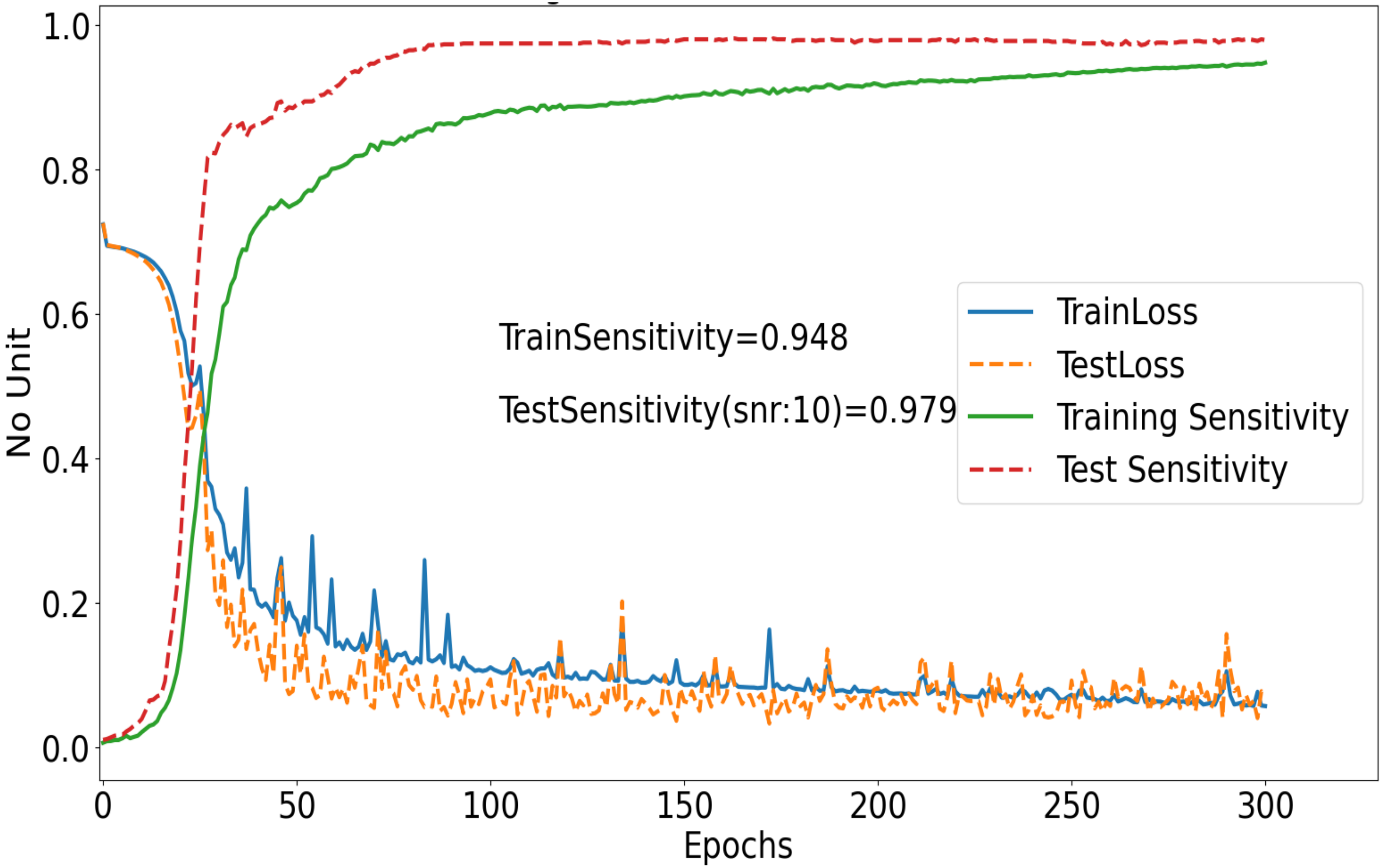
$$x = \lfloor \mathrm{N_{sample}} \times FAR \rfloor$$

$$p_{treshold} = p(x)$$



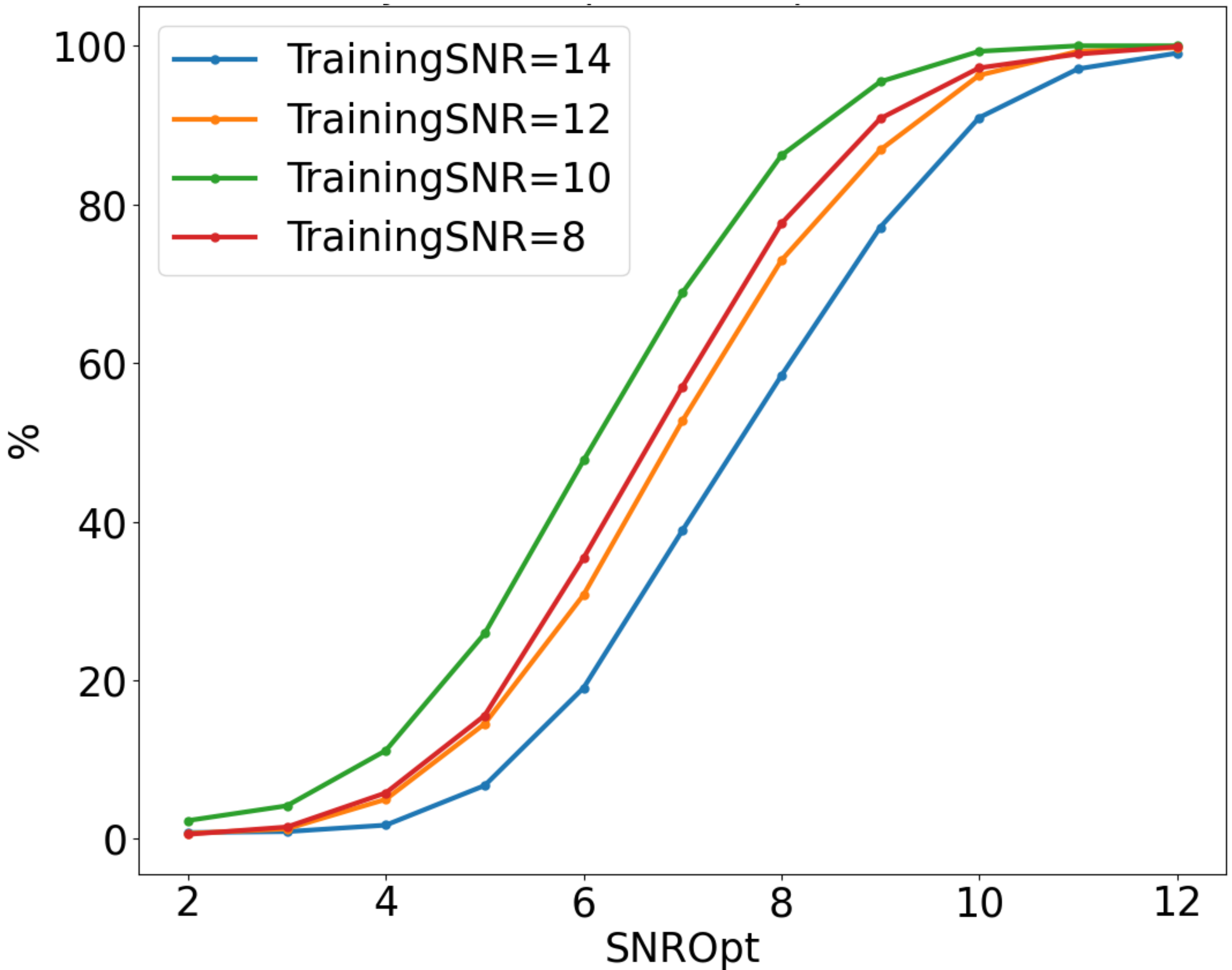
Results and Analysis: Exemple of training

- First plateau
- Significant improvement
- Last plateau
- Overfit
- Oscillations



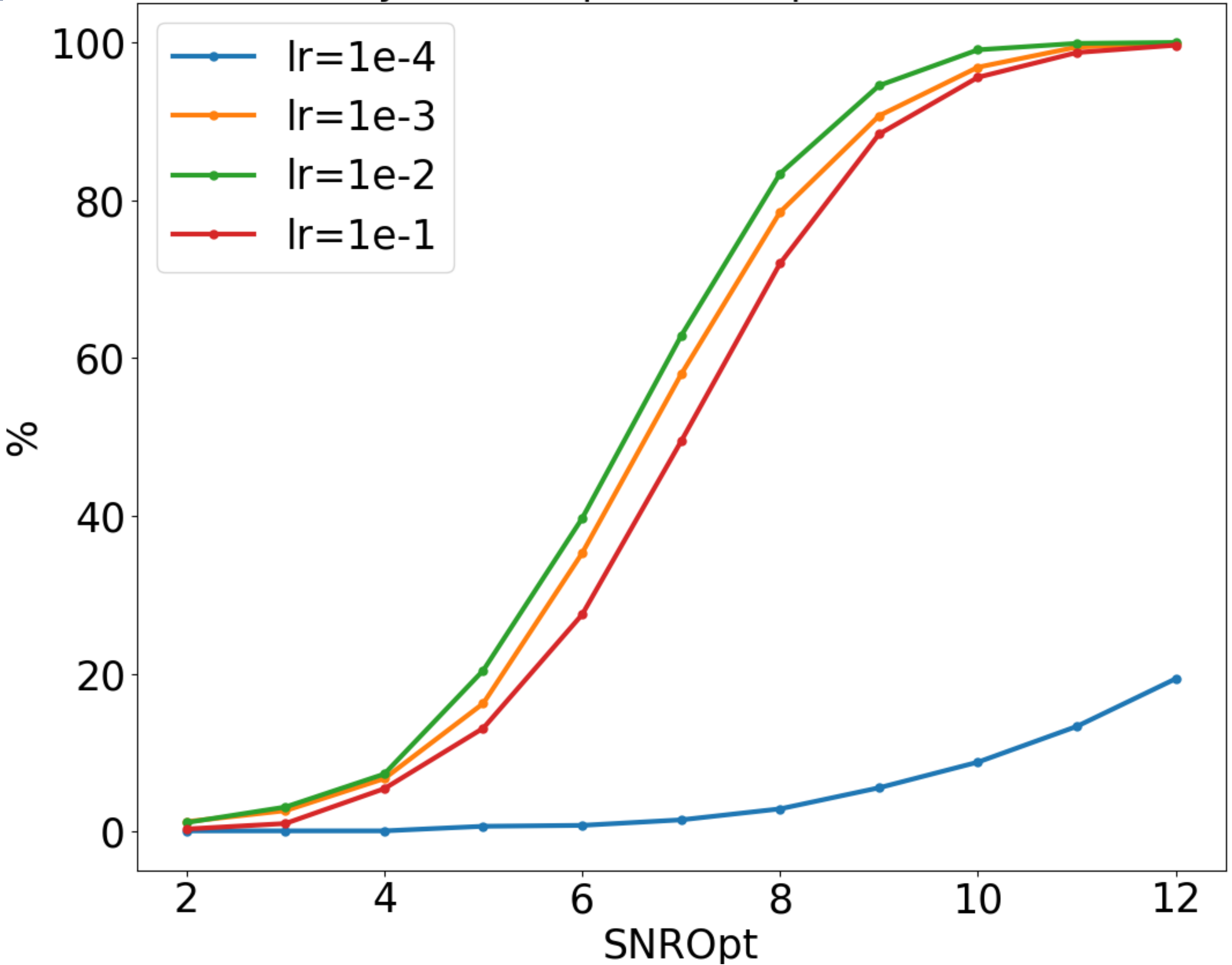
Results and Analysis: Training SNR Influence

- **300 epochs** and $\boldsymbol{l_r = 0.003}$
- Decrease the SNR → training harder
- Best Results for low SNR
- SNR too low → convergence too slow

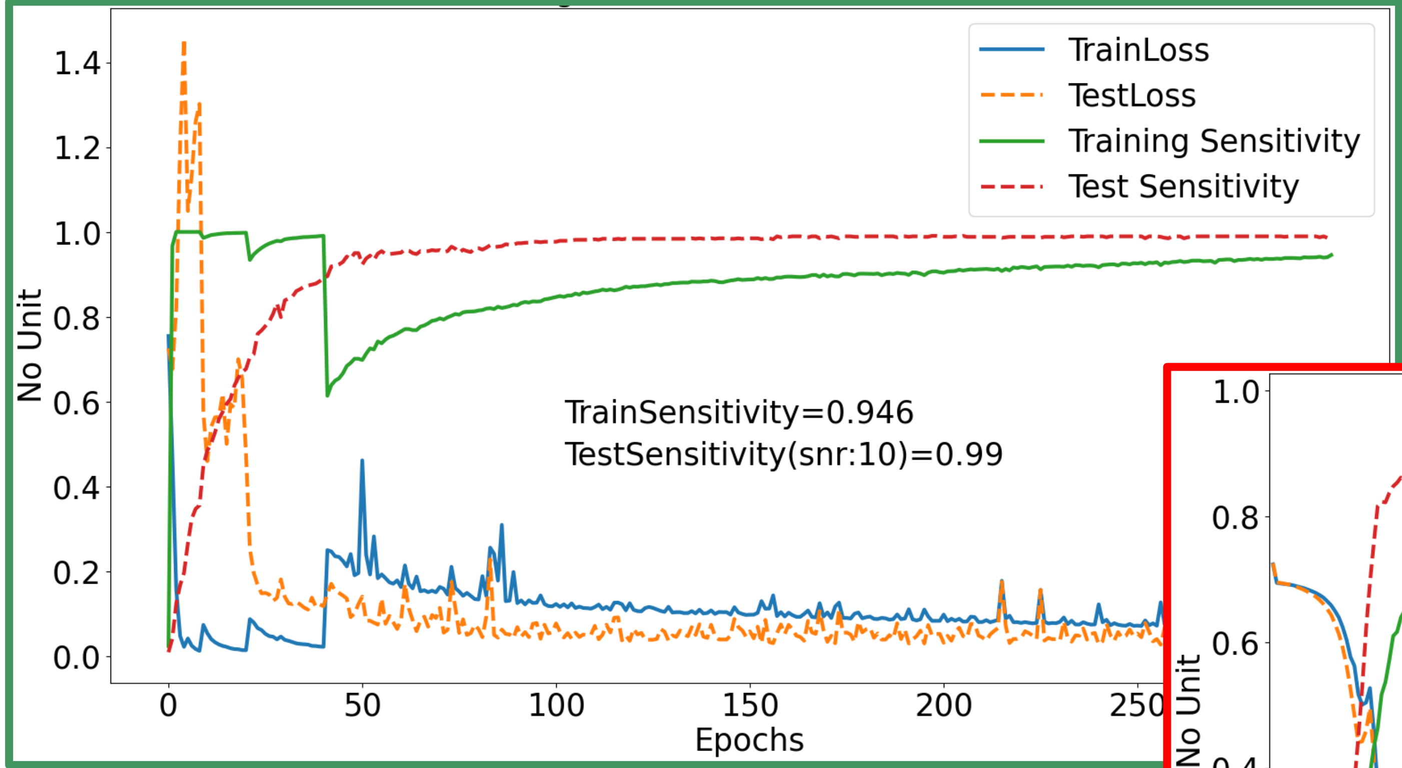


Results and Analysis: Learning-rate Influence

- **300 epochs** and Training $\boldsymbol{SNR} = \mathbf{10}$
- Increase the l_r → faster training
- Best Results for high l_r
- l_r too low → convergence too slow
- l_r too high → oscillations and overfit

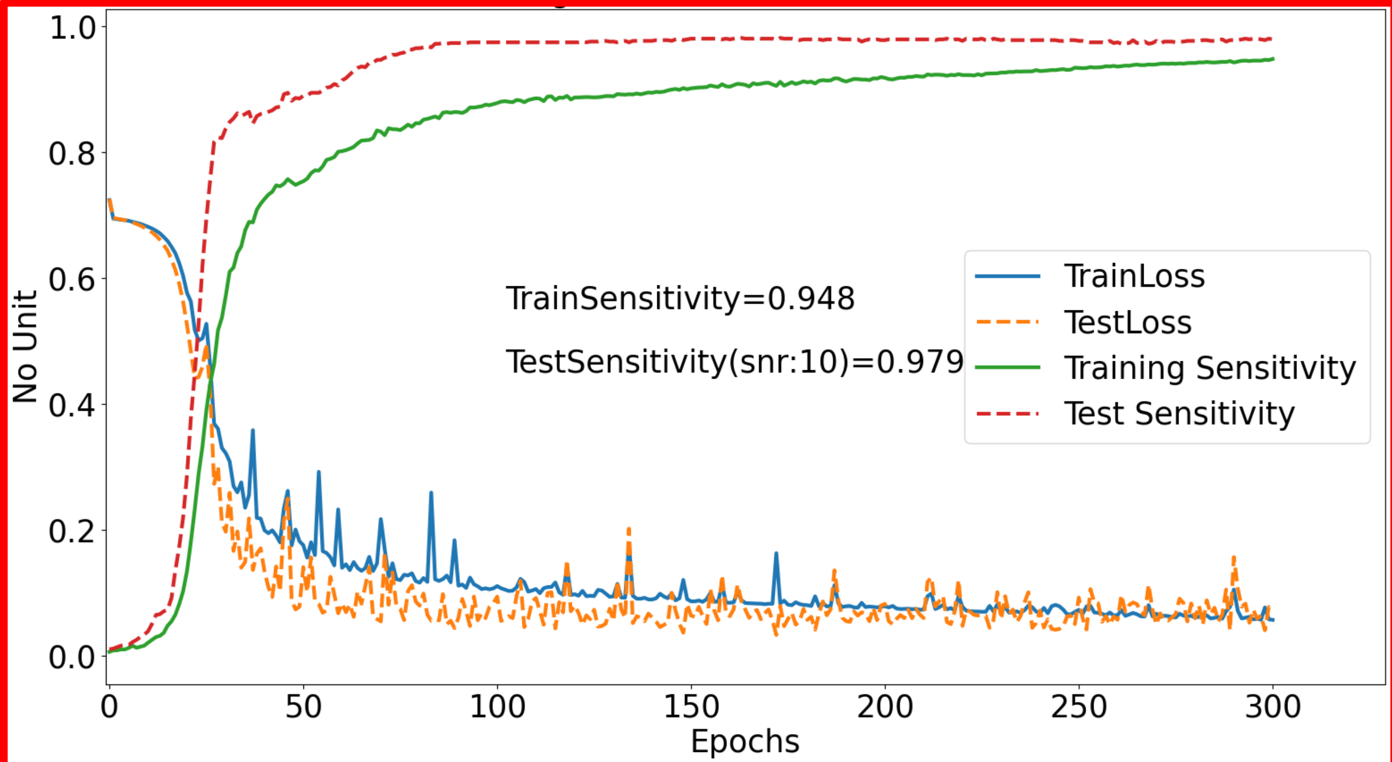


Results and Analysis: Kind of Training Influence



Fix < Decrease SNR

- Data change → Limitation of overfit
- Start Higher SNR → Limitation of first plateau
- End Lower SNR → Better Sensitivity



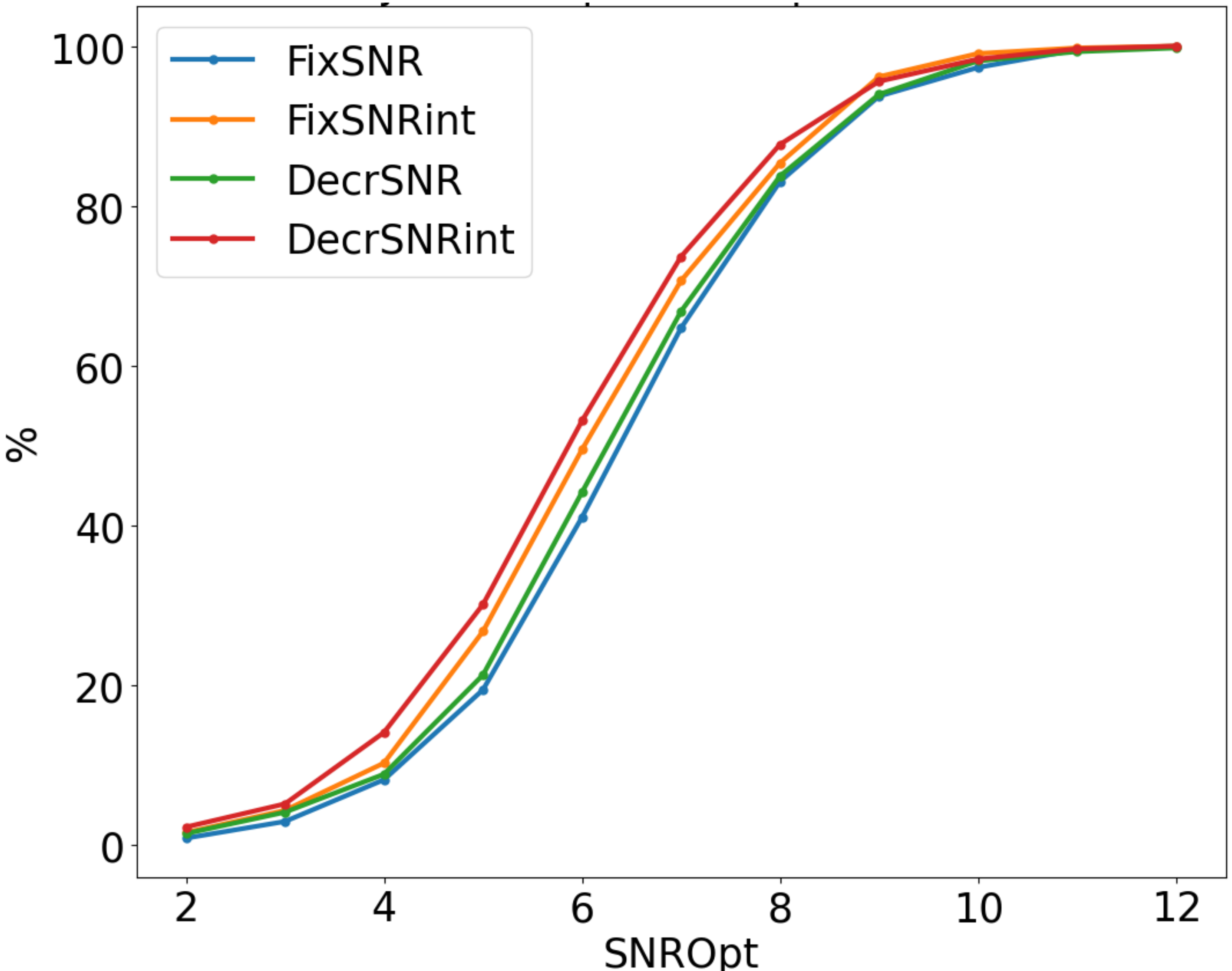
Scalar < Interval SNR

- Lower SNR access → Better sensitivity
- Interval → Limitation of overfit
- Higher SNR access → Limitation of first plateau

Results and Analysis: Kind of Training Influence

Best Training: Decrease and interval SNR

- Interval & Data change → Limitation of overfit
- Higher SNR access & Start very High SNR → No first plateau
- Lower SNR access & End very low SNR → Better sensitivity



Conclusion

- Synthesis
- Outlook

Conclusion: Synthesis

- Great **potential** for **ML detection** but **more Statistics** needed
- Big **importance** of **learning rate** and training SNR
- Best Training**: Interval decreasing SNR**
- **100% sensitivity** (for 0.5% FAR) for **SNR>=8**
- **Importance** of **SNR repartition**

Conclusion: Outlook

Data Generation

- Numerical relativity signals (with merger)
- Optimal template Bank
- Several Φ_c
- Long signals (before merger)

Neural Network

- Use of GPUs
- Add of a new class (glitch)
- Estimator for mass parameters: delete softmax and use mass labels

Training and Results

- Have for statistics
- Evolving Learning-rate
- Testing NN on exotic signals (spin precession)

Thank you !

Back-up : Tools

Plotting Results

matplotlib

Development platform

GitLab

Programming Language

Data Generation

NumPy

PyCBC

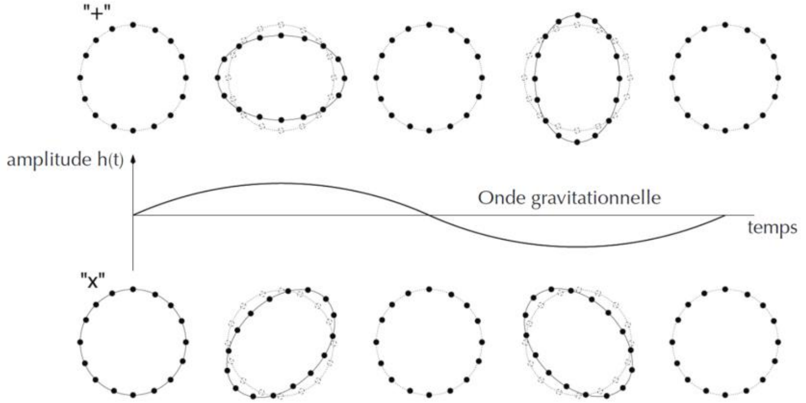
ML incubator

mxnet

Back-up : Relativité générale

Propagation dans le vide: similaire aux ondes EM

$$\Box \bar{h}_{\mu\nu} = 0 \iff (\nabla^2 - \frac{\partial^2}{\partial t^2}) h_{\mu\nu} = 0$$



Origine: masses accélérées, moment quadripolaire

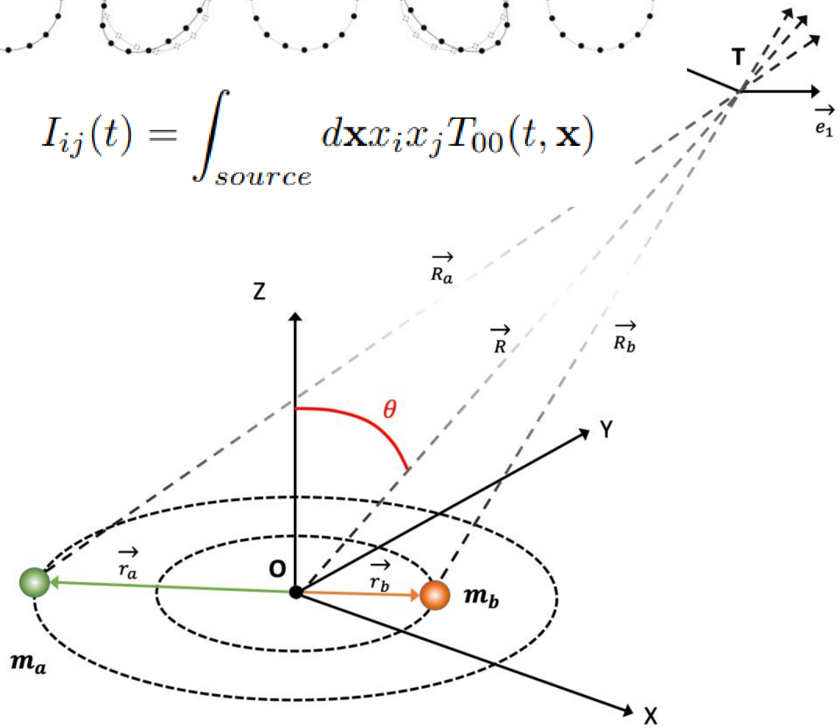
$$\bar{h}_{\mu\nu}(\mathbf{x}, t) = -\frac{4G}{c^4} \int_{source} \frac{T_{\mu\nu}(\mathbf{x'}, t - \frac{|\mathbf{x}-\mathbf{x'}|}{c})}{|\mathbf{x}-\mathbf{x'}|} \qquad \bar{h}_{ij}(t) = \frac{2G}{rc^4} \frac{d^2 I_{ij}(t - R/c)}{dt^2}$$

$$I_{ij}(t) = \int_{source} d\mathbf{x} x_i x_j T_{00}(t, \mathbf{x})$$

Système binaire: Calcul dans le cadre de l'EM (polarisation x)

$$h(t) = \frac{\eta (GM)^{5/3} \omega^{2/3}(t)}{4Rc^4} \cos 2\Phi(t)$$

$$\Phi(t) = \int_{t_0}^{t} \omega(u) du = -(\frac{2}{5})^{5/8} (\frac{t_c - t}{t_{SC}})^{5/8} + \Phi_c$$



Back-up : Kind of Noise Influence

- **Same total SNR**
- **Different SNR distribution**
- **Higher maximum** of **signal** for **colored noise** than flat noise
- Neural Network recognizes **better** a **high** expected value **shift** in **few distributions** than a **weak** expected value **shift** in **many distributions**