



# AI in nuclear and particle physics

Wojciech Broniowski

Jan Kochanowski U., Kielce, and  
Institute of Nuclear Physics PAN, Cracow, Poland

**PhD Forum, UJK**

2 June 2022

# Example of bombarding news...

ARTIFICIAL INTELLIGENCE

## AI has cracked a key mathematical puzzle for understanding our world

Partial differential equations can describe everything from planetary motion to plate tectonics, but they're notoriously hard to solve.

By Karen Hao

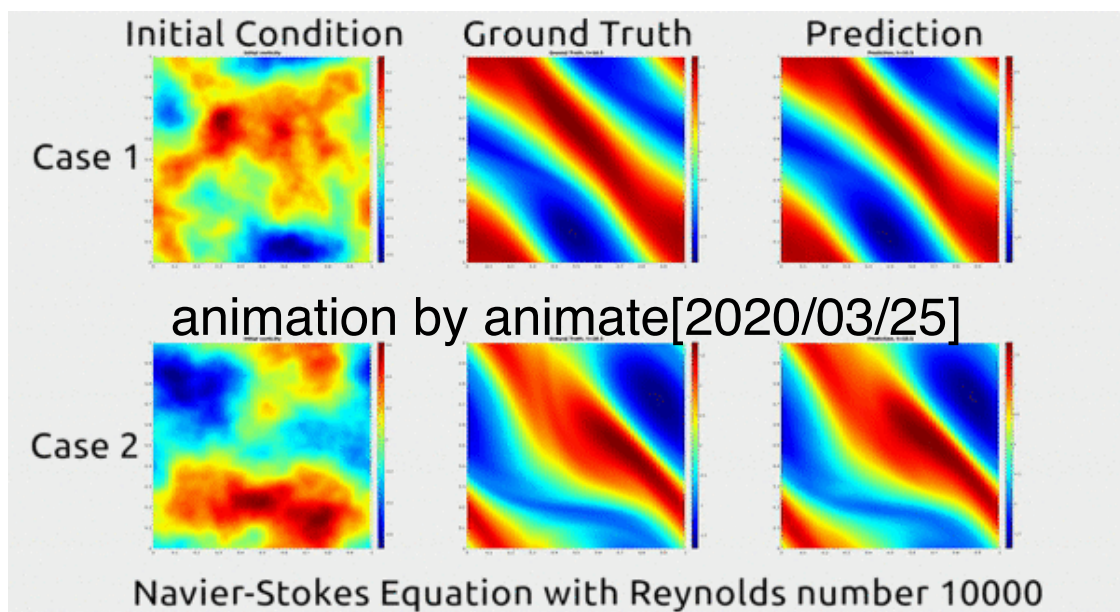
October 30, 2020



<https://arxiv.org/pdf/2010.08895.pdf>

<https://www.technologyreview.com/2020/10/30/1011435/ai-fourier-neural-network-cracks-navier-stokes-and-partial-differential-equations/>

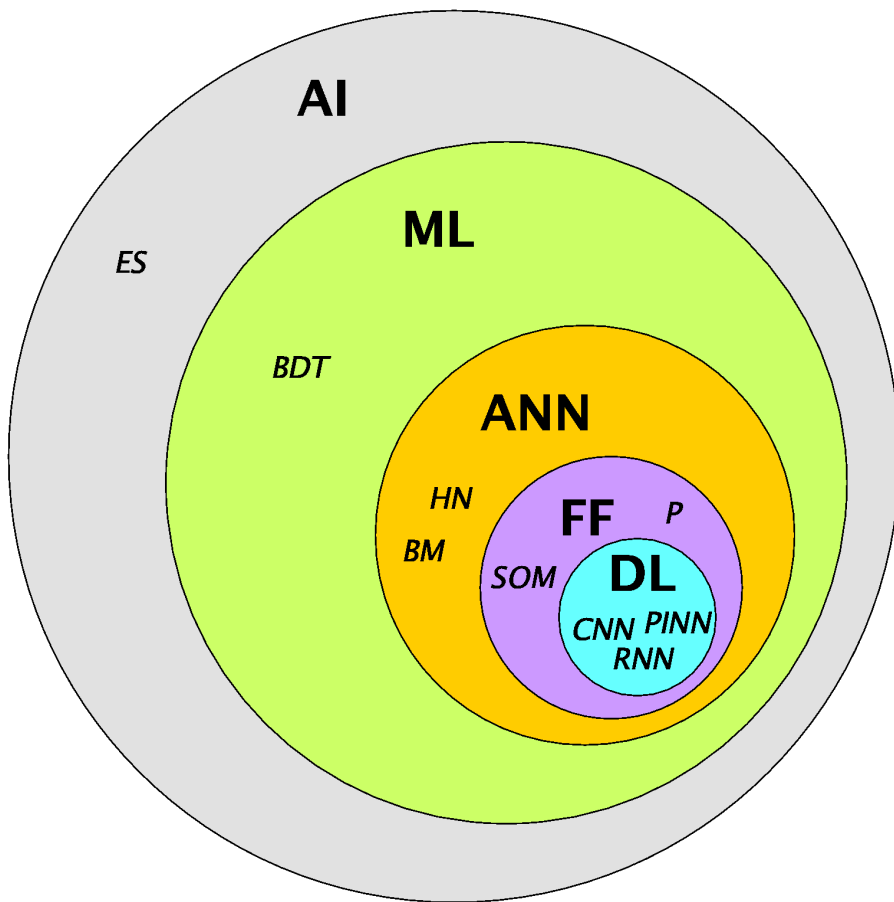
## Example of bombarding news...



<https://arxiv.org/pdf/2010.08895.pdf>

<https://www.technologyreview.com/2020/10/30/1011435/ai-fourier-neural-network-cracks-navier-stokes-and-partial-differential-equations/>

# "Classification"



Artificial Intelligence

Expert Systems

Machine Learning

Boosted Decision Trees

Artificial Neural Networks

Hopfield Network

Boltzmann Machine

Feed Forward

Perceptron

Self-Organizing Maps

Deep Learning

Convolutional NN

Recurrent NN

Physics-informed NN

...

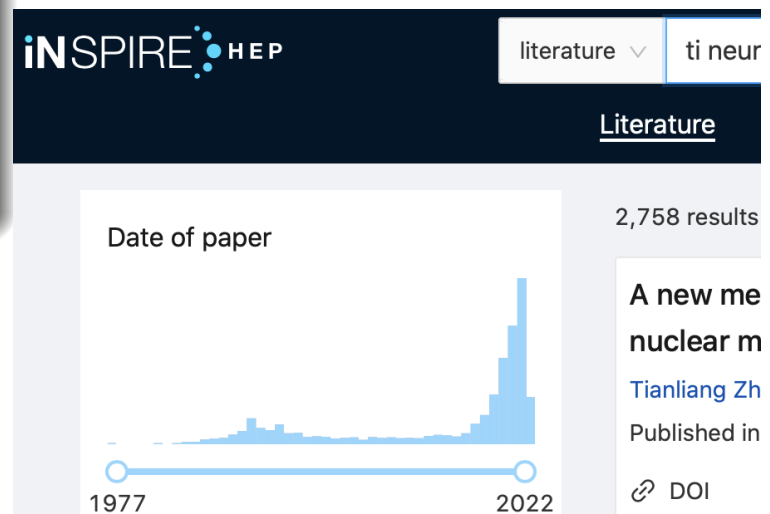
# Outline

- Physics questions
- Mini-review of ANN
- Classification and regression
- Deep NN
- Physics-informed NN
- Outreach

Revolution ~2010:

- Abundance of data
- Huge computational power
- Novel NN architectures and algorithms

Understanding the black box...



# ML in nuclear/particle physics

## Basic tasks

Reviews:

<https://arxiv.org/abs/2112.02309>

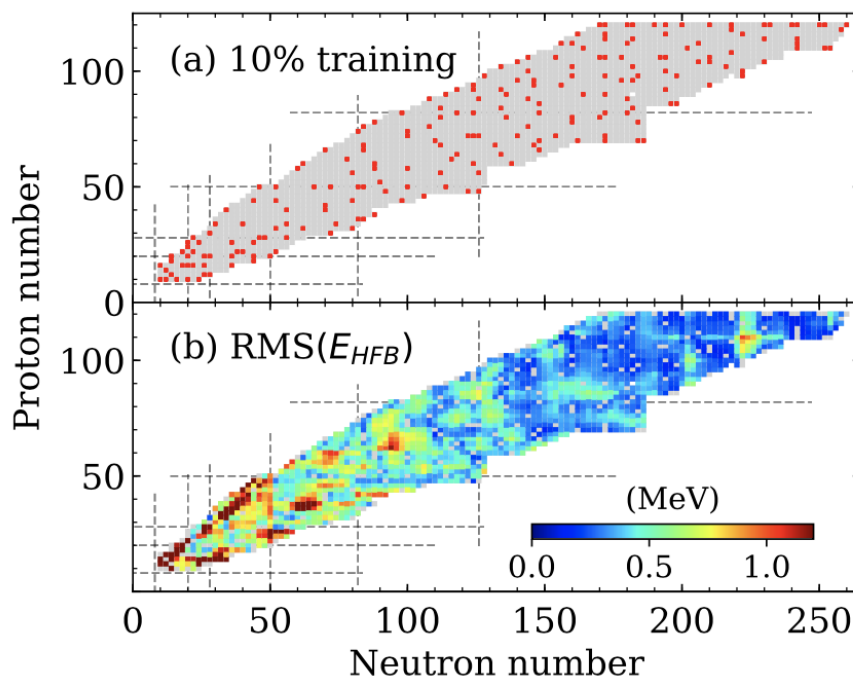
<https://doi.org/10.1140/epja/s10050-020-00290-x>

- Classification (data analysis, event identification, triggering, jets, dedicated searches, ...)
- Regression (efficient fitting of models, solving differential equations (!), inverse problems, ...)
- Clusterization (data analysis, efficient representation of data, ...)
- Generation (simulating events, data augmentation, detector physics, ...)
- ...

Teaching slow and expensive, using fast and cheap!

## Example: Nuclear structure (Hartree-Fock-Bogolyubov)

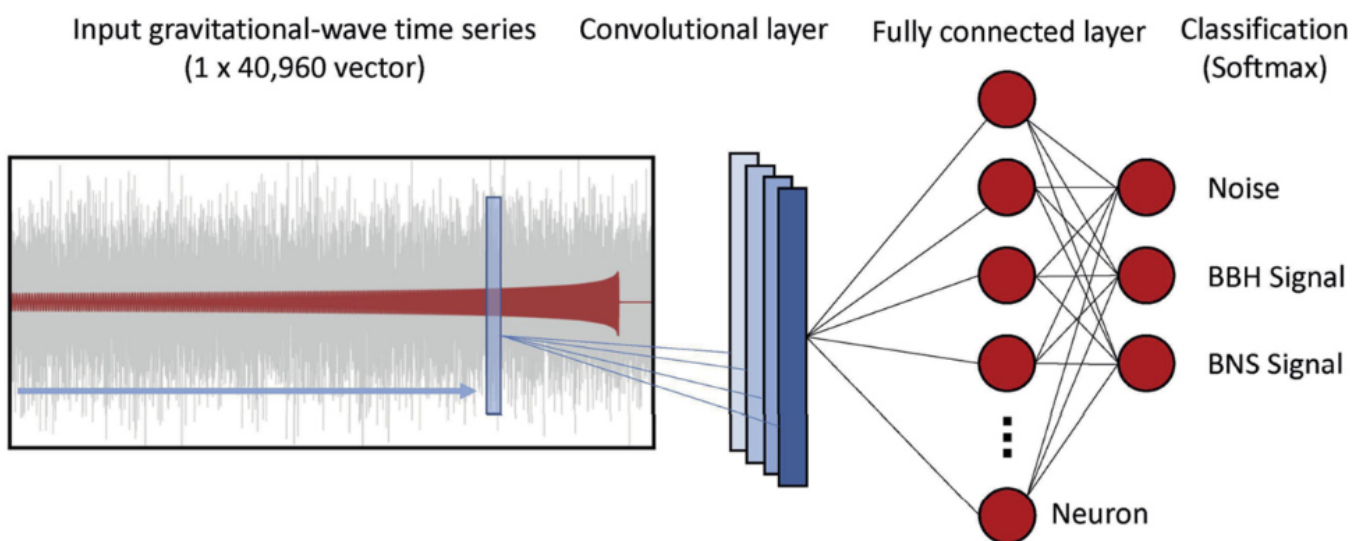
<https://doi.org/10.1103/PhysRevLett.124.162502>: "deep neural networks is capable of predicting the ground-state and excited energies of more than 1800 atomic nuclei with an accuracy akin to the one achieved by state-of-the-art nuclear energy density functionals and with significantly less computational cost"





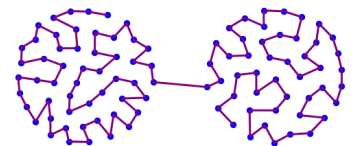
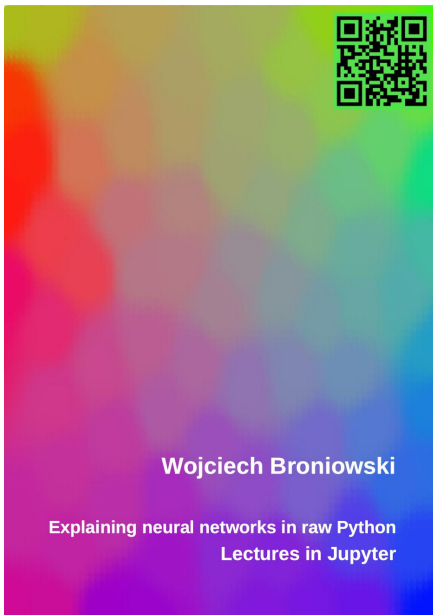
## Example 2: Gravitational waves from binary merger

<https://doi.org/10.1016/j.physletb.2020.135330>



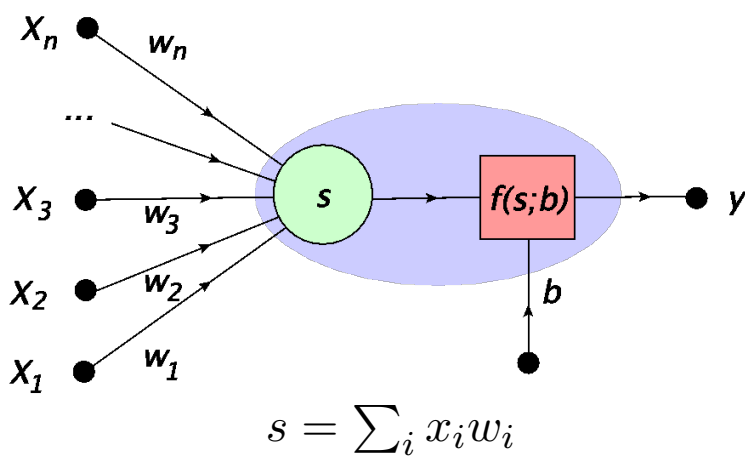
# Mini-review of ANN

Simplest Pythonic introduction ever!



Executable Jupyter Book

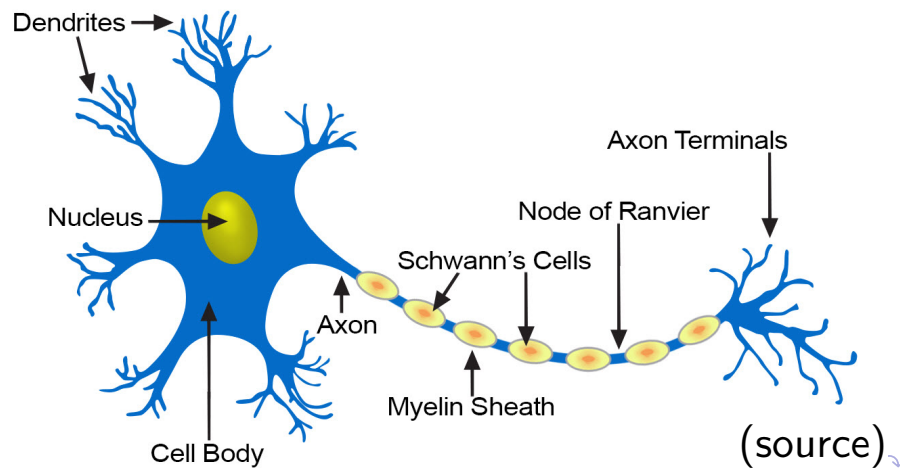
# McCulloch-Pitts neuron (MCP), 1943



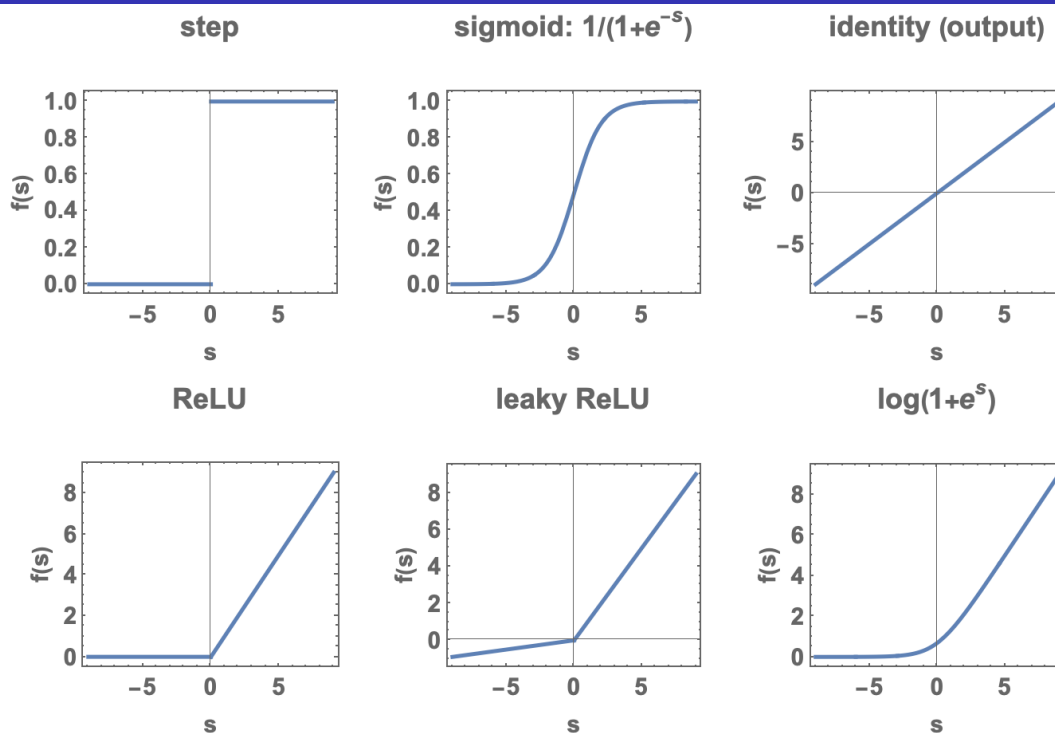
$x_i$  – inputs  
 $w_i$  – weights (hyperparameters)  
 $s$  – signal (weighed sum of inputs)  
 $f$  – activation function  
 $b$  – bias (threshold)  
 $y$  – output

- Fires, if the signal is sufficiently strong,  $s > b$
- Mimics (approximately) the biological neuron

## Structure of a Typical Neuron



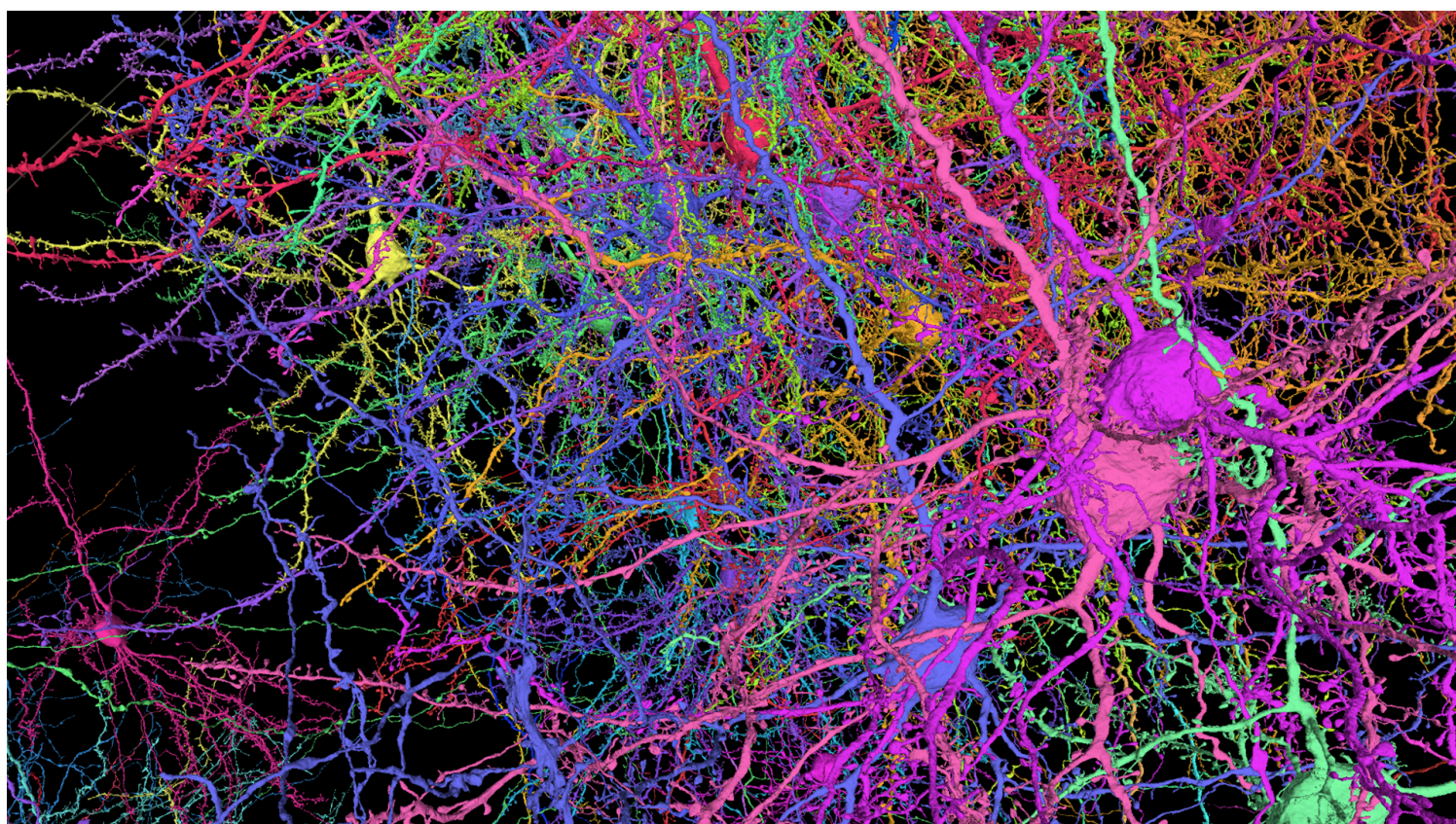
# Activation functions



Nonlinearity absolutely essential!

Without nonlinearity just matrix multiplication! Nonlinearity  $\rightarrow$  dissipative characteristics, attractor-based dynamics, allows NN to better catch non-trivial correlations

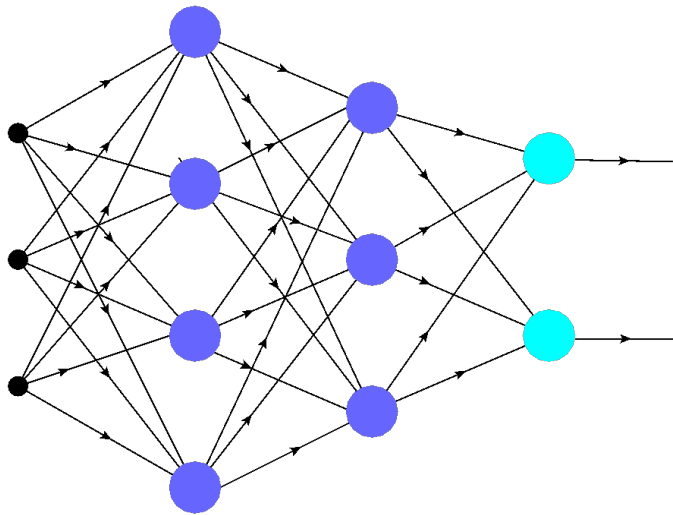
1mm<sup>3</sup> of the mouse brain



(MICrONS)



# Artificial neural network (ANN)



input  
intermediate neuron layers ( $\geq 0$ )  
output neuron layer

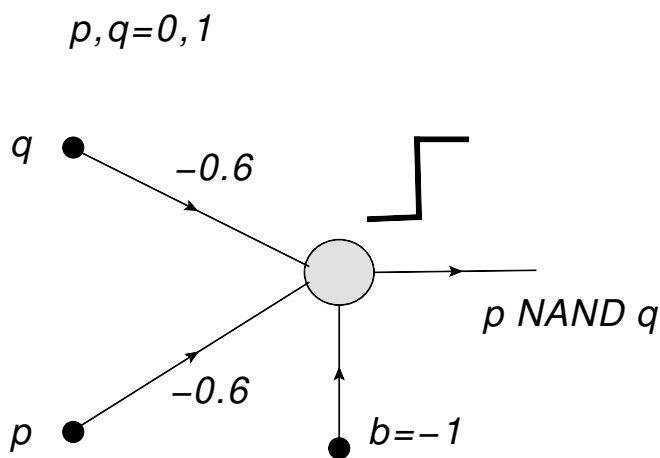
here:

- feed-forward network
- fully connected
- 2 intermediate neuron layers

- Edges carry weights  $w_{ij}$  (many hyperparameters)

## Logical networks

With proper selection of weights, the MCP neuron becomes a logical gate



- Completeness of NAND (or NOR) gates: any logical network (boolean function) can be constructed as a feed-forward (multilayer) ANN
- → Completeness of ANN in the Church-Turing sense: any computation possible
- Fuzzy logic with smooth step functions

## Inequality conditions from MCP neurons

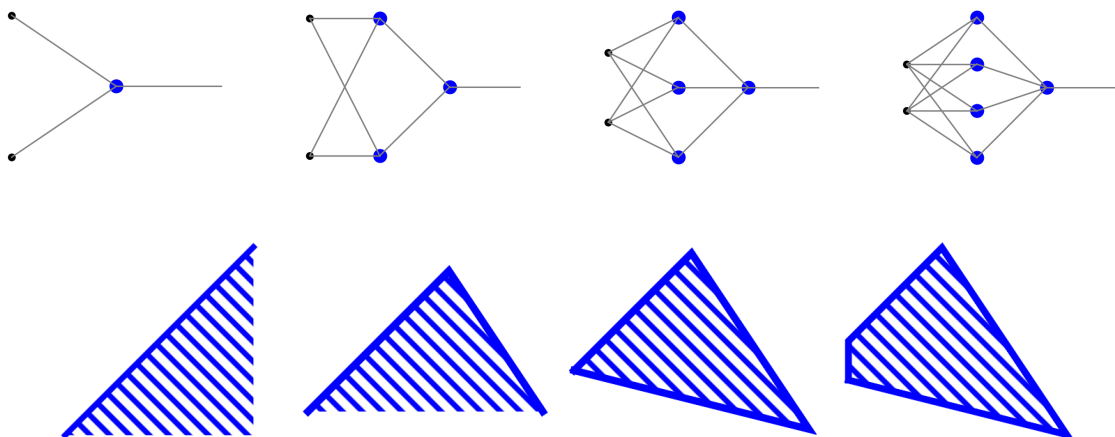
Example in 2 dimensions:

(Binary classifier)

$x_1w_1 + x_2w_2 > b$  – half-plane condition (in more dim. hyper half-space)

linearly separable sets

first neuron layer – half-plane conditions, second (output) – multiple conjunction



Any convex polytope region can be classified with a 2-neuron layer ANN

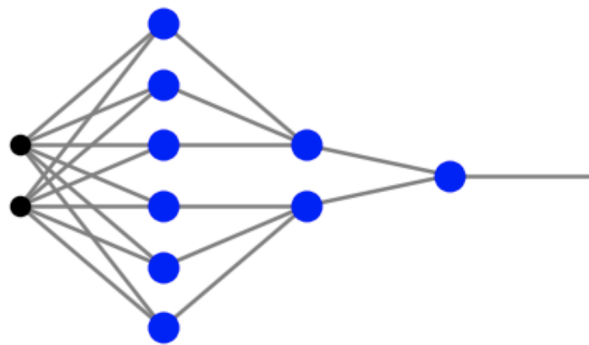
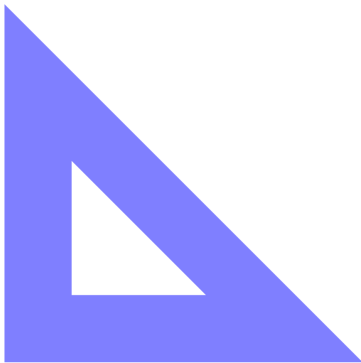


## More layers

half-plane conditions

$p$  – in big triangle  
 $q$  – in small triangle

$$p \wedge q$$



Any polytope region in any number of dimensions can be classified with an at least 3-neuron layer ANN. The number of needed neurons reflects the complexity of the region

# Setting hyperparameters by hand

In our example one can easily (from geometry) infer the weights and biases *a priori*. The geometric conditions  $\rightarrow$  the first neuron layer:

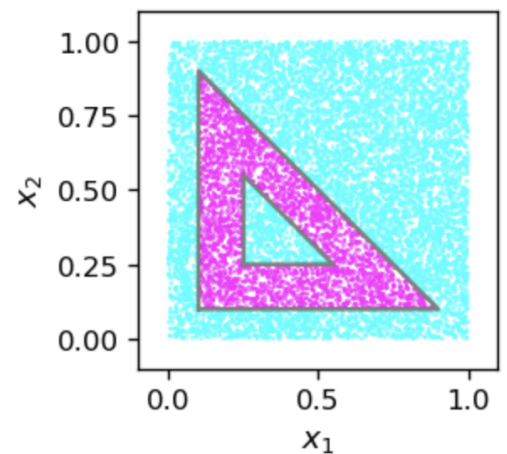
$\alpha$	inequality condition	$b_\alpha^1$	$w_{1\alpha}^1$	$w_{2\alpha}^1$	
1	$x_1 > 0.1$	0.1	1	0	big triangle
2	$x_2 > 0.1$	0.1	0	1	
3	$x_1 + x_2 < 1$	-1	-1	-1	
4	$x_1 > 0.25$	0.25	1	0	small triangle
5	$x_2 > 0.25$	0.25	0	1	
6	$x_1 + x_2 < 0.8$	-0.8	-1	-1	

Conjunctions in the second neuron layer:

$\alpha$	$b_\alpha^2$	$w_{1\alpha}^2$	$w_{2\alpha}^2$	$w_{3\alpha}^2$	$w_{4\alpha}^2$	$w_{5\alpha}^2$	$w_{6\alpha}^2$
1	1	0.4	0.4	0.4	0	0	0
2	1	0	0	0	0.4	0.4	0.4

Finally, in the output layer we need the  $p \wedge \sim q$  gate:

$\alpha$	$b_\alpha^3$	$w_{1\alpha}^3$	$w_{2\alpha}^3$
1	1	1.2	-0.6



This is not what we want to do in ML!

# Concept of learning

## Learning

A suitable algorithm sets the hyperparameters based on the data alone, "automatically", without a direct programmers interference!

- **Supervised** learning:  
Data points have **features** and **labels**  
Example: features – coordinates, labels – belongs to an area, i.e., **has a property** (1) or not (0)  
**Training** and **test** data samples, **loss function**
- **Unsupervised** learning:  
No labels, looking for correlations/clusterization in the data
- Semi-supervised, online, ...
- **Reinforcement** learning (not covered):  
Agent actions to maximize reward, interaction with the environment (**robotics**)

## Back-propagation [Bryson, Ho, 1969]

Goal: minimize the loss function between (feed-forward) NN answers and labels

Notation:  $x_0^j = 1$ ,  $w_{0m}^j = -b_m^j$ ,  $p \in$  training sample

$$E(\{w\}) = \sum_p \sum_{\alpha_l=1}^{n_l} [y_{o,\alpha_l}(\{w\}) - y_{t,\alpha_l}]^2 = \sum_p \sum_{\alpha_l=1}^{n_l} \left[ f \left( \sum_{\alpha_{l-1}=1}^{n_{l-1}} f \left( \dots f \left( \sum_{\alpha_0=0}^{n_0} x_{\alpha_0}^0 w_{\alpha_0 \alpha_1}^1 \right) w_{\alpha_1 \alpha_2}^2 + x_0^1 w_{0 \alpha_2}^2 \dots \right) w_{\alpha_{l-1} \alpha_l}^l + x_0^{l-1} w_{0 \alpha_l}^l \right) - y_{t,\alpha_l} \right]^2$$

Chain rule, steepest gradient (differentiability of  $f$ )  $\rightarrow$

### Backprop

$$D_{\alpha_l}^l = 2(y_{o,\alpha_l} - y_{t,\alpha_l}) f'(s_{\alpha_l}^l)$$

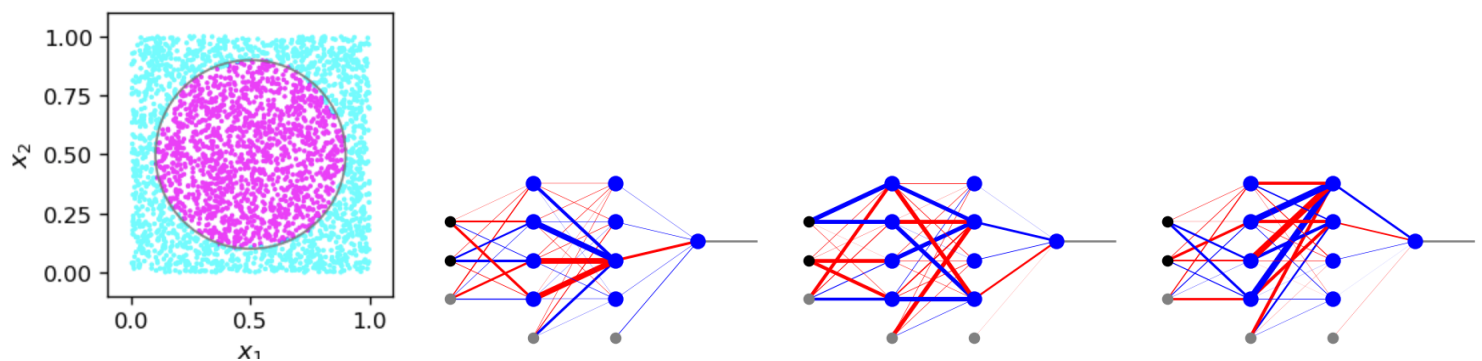
$$D_{\alpha_j}^j = \sum_{\alpha_{j+1}} D_{\alpha_{j+1}}^{j+1} w_{\alpha_j \alpha_{j+1}}^{j+1} f'(s_{\alpha_j}^j), \quad j = l-1, l-2, \dots, 1$$

$$w_{\alpha_{j-1} \alpha_j}^j \rightarrow w_{\alpha_{j-1} \alpha_j}^j - \varepsilon x_{\alpha_{j-1}}^{j-1} D_{\alpha_j}^j \quad (\text{recursion goes backwards})$$

## Example, local minima

- Methodology: the program "writes itself" by finding the parameters via minimization of the loss function
- Black box - different paradigm, works, many local minima are fine!

Example:



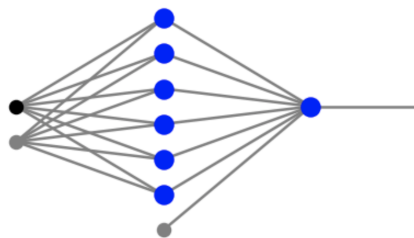
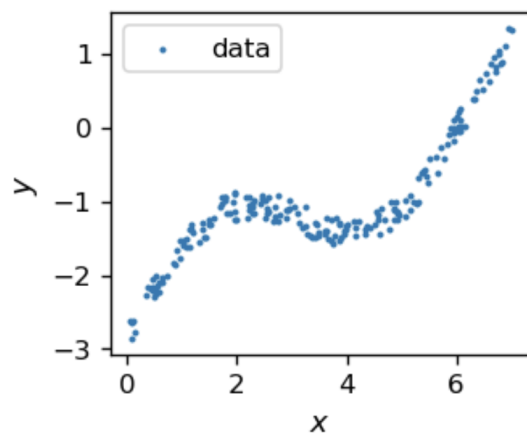
ANN does not have any idea about geometry!

Learn on mistakes!

# Regression

# Regression

We want to represent (interpolate) some noisy experimental data

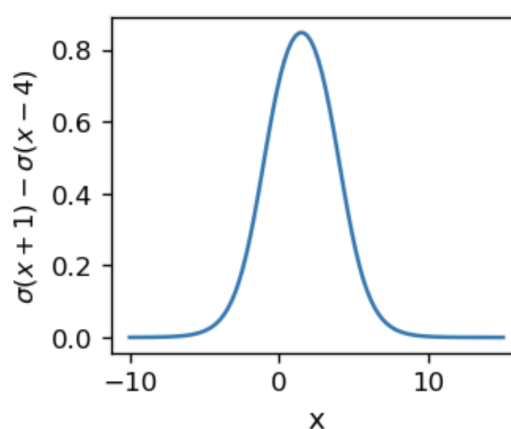


Instead of using a specific function (e.g. polynomial) we use ANN. The intermediate layer has sigmoid activation functions, but the last layer has an **identity** activation to cover the values range

$x$  - feature,  $y$  - label

## Sigmoid basis

To see that a basis of sigmoid can be used to represent a function, consider a difference of two sigmoids with different thresholds:



We can "fill" any function with such bell-shaped functions, or more generally with (sufficiently many) sigmoids

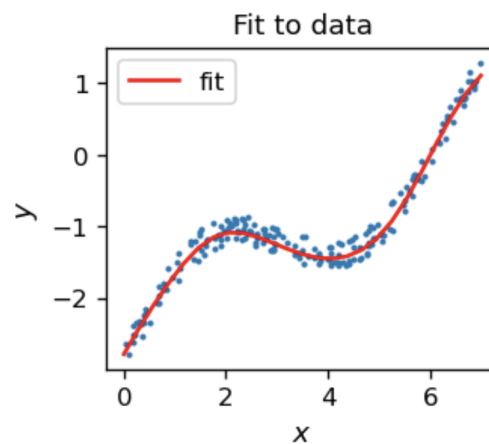
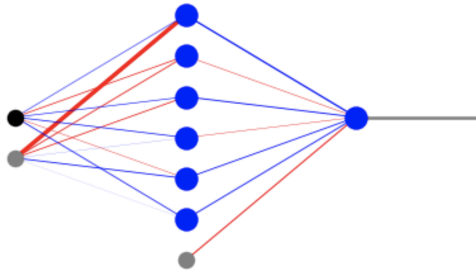
The first neuron layer forms the basis, the second provides "amplitudes"

The loss function = least squares, or other choices



# Training regression

Use backprop to train (19 parameters!)



- $\Delta E_{test} \geq \Delta E_{train}$ , possible overfitting problem (regularization), optimum number of neurons
- Adaptiveness

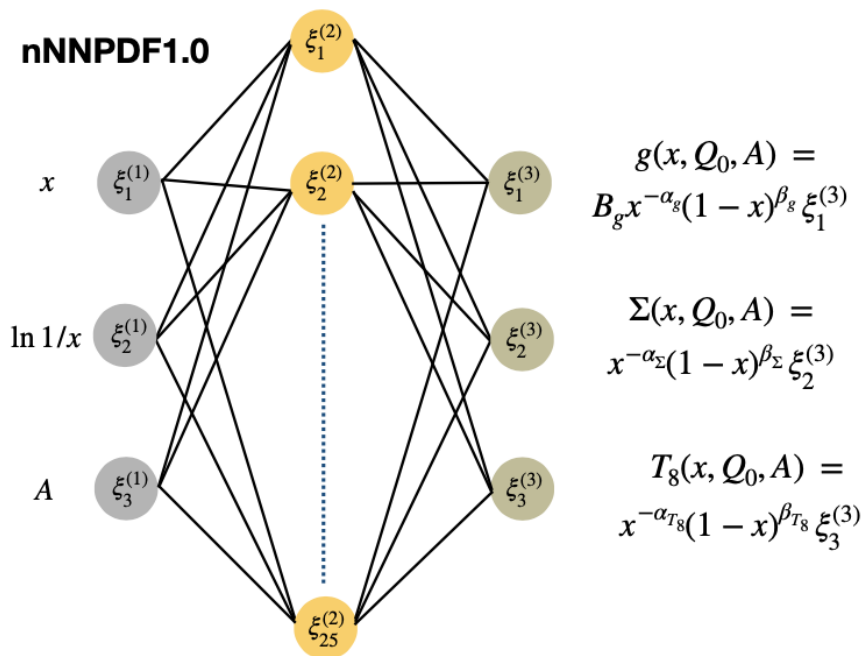
To interpolate in 2 or more dim. one needs ANNs with at least 3 neuron layers

A pair of neurons in the first neuron layer can form a hump in  $x_1$ , another pair – in  $x_2$ , and so on for the remaining  $x_i$ . Conjunction of these humps in the second neuron layer yields a “basis” function focused in a region around a certain point in the  $n$ -dimensional input space. The last layer provides amplitudes.

[cf. Müller, Reinhardt, Strickland]

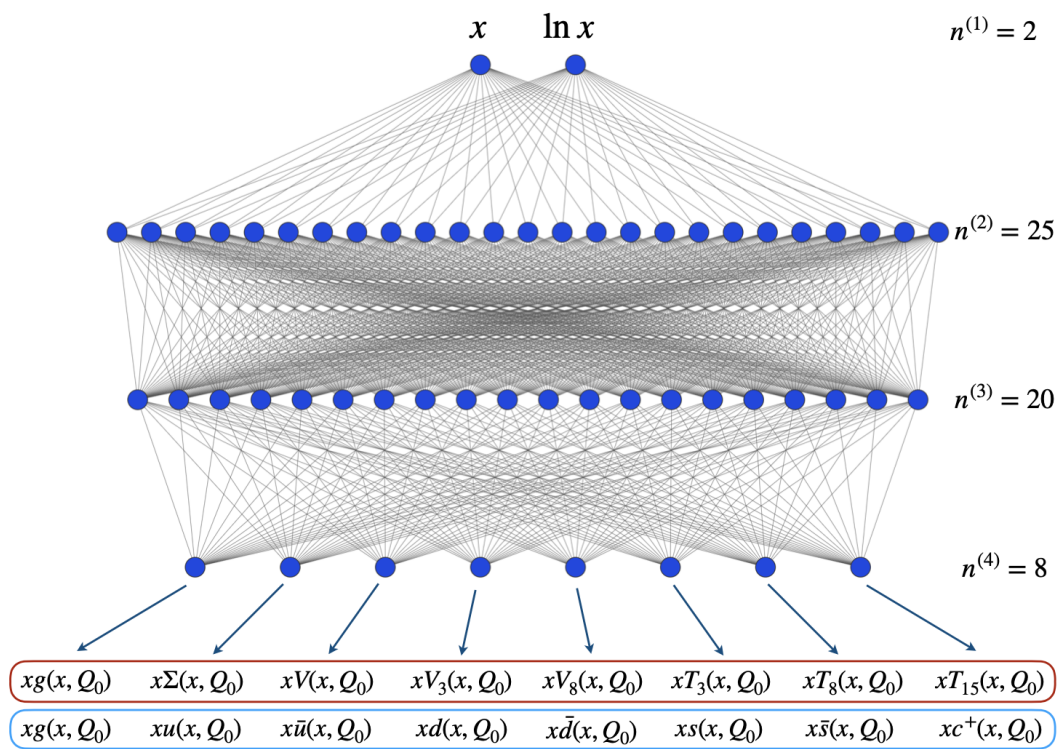


# Example: NNPDF



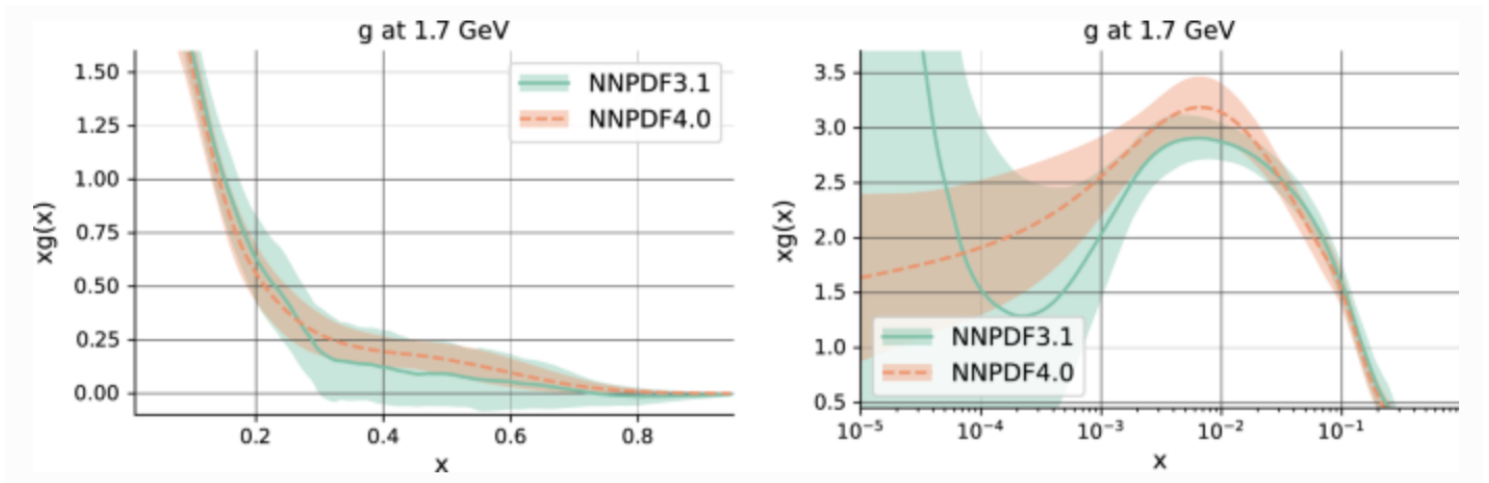
[NNPDF Collaboration, Eur. Phys. J. C (2019) 79:471]

# NNPDF 4



[arXiv:2109.02653]

## Sample result: glue from NNPDF

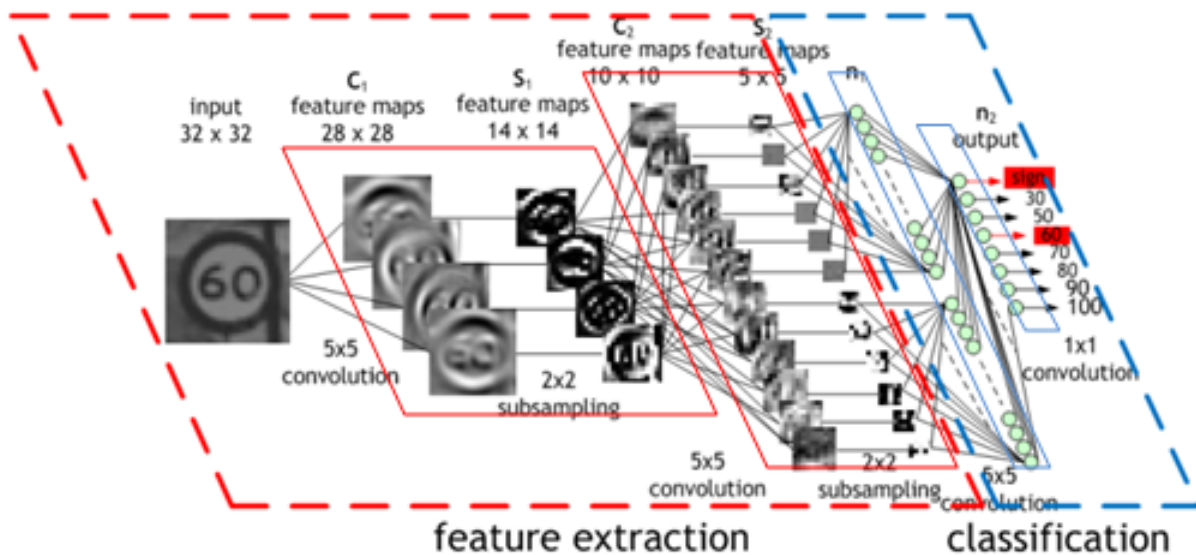


<https://doi.org/10.1140/epjc/s10052-021-09863-6>

# Going deep

# Convolutional NN (CNN)

Limited (**local**) connectiveness (as in our biological vision system), action of various **filters** producing **feature maps**, run on **GPU**



(source)

Composing ANNs as "Lego blocks", backprop, partially-trained networks  
Local features are extracted and correlated, longer and longer range

# CNN for gravitational waves classifier from intro

**Table 1**

**Architecture of the deep neural network.** The architecture of the deep one-dimensional convolutional network consist of input layer, followed by 19 hidden layers, and output *softmax* layer. The size of the network is about 83 MB.

	Layer	Array Type	Size
	Input	Vector	40960
1	Reshape	Matrix	$1 \times 40960$
2	Convolution (1D)	Matrix	$32 \times 40945$
3	Pooling	Matrix	$32 \times 10236$
4	ReLU	Matrix	$32 \times 10236$
5	Convolution (1D)	Matrix	$64 \times 10229$
6	Pooling	Matrix	$64 \times 2557$
7	ReLU	Matrix	$64 \times 2557$
8	Convolution (1D)	Matrix	$128 \times 2550$
9	Pooling	Matrix	$128 \times 637$
10	ReLU	Matrix	$128 \times 637$
11	Convolution (1D)	Matrix	$256 \times 623$
12	Pooling	Matrix	$256 \times 155$
13	ReLU	Matrix	$256 \times 155$
14	Flatten	Vector	39680
15	Dense Layer	Vector	128
16	ReLU	Vector	128
17	Dense Layer	Vector	64
18	ReLU	Vector	64
19	Dense Layer	Vector	3
	Output (Softmax)	Vector	3

(taught on 100000 augmented data)

## Modern ML

Some thoughts from a review by Matthiew D. Schwartz (Harvard)

<https://arxiv.org/abs/2103.12226>

*The modern approach is to feed raw, minimally-processed data, rather than high-level physically-motivated variables, into a deep neural network. The network is then free to find what it thinks is most valuable in the data*

*In the past, physicists ... would devote their efforts to understanding signatures of particular particles or processes from first-principles: why should a stream of pions coming from a  $W$  boson decay look different than a stream coming from an energetic gluon? Now we simply simulate the events, and let neural network learn to tell the two samples apart. Even a relatively simple dense network with 10 lines of python code can blow the traditional discriminants out of the water. Progress ... has come from taking algorithms masterly engineered for other applications, like convolutional neural networks, and shoehorning the collider data into a format that these algorithms can process. Unfortunately doing so can inhibit the extraction of any kind of physical understanding from the network itself*

*While it is easy to train an algorithm to reproduce the majority of a model's output, it is much harder to train it to reproduce every nuance of the model*



# Other developments

# Detector simulation with Generative Adversarial NN (GAN)

GAN uses one NN to generate (noisy) data samples, and a second *adversary* NN (trained on a training sample) to assess if the generated data are OK

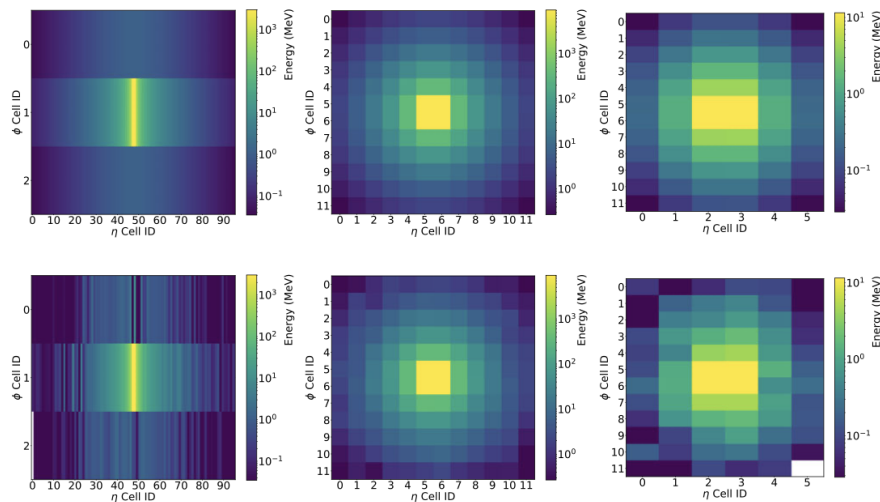


FIG. 6. Average  $e^+$  GEANT4 shower (top), and average  $e^+$  CALOGAN shower (bottom), with progressive calorimeter depth (left to right).

Orders of magnitude faster!

10.1103/PhysRevD.97.014021

# Physics-informed NN (PINN)

Include physics information in the architecture and/or loss function (symmetries, constraints, physics laws)

→ learn (partial) differential equations with boundary/initial conditions

Lu, Meng, Mao, Karniadakis <https://epubs.siam.org/doi/pdf/10.1137/19M1274067>

Karniadakis et al. <https://www.nature.com/articles/s42254-021-00314-5>

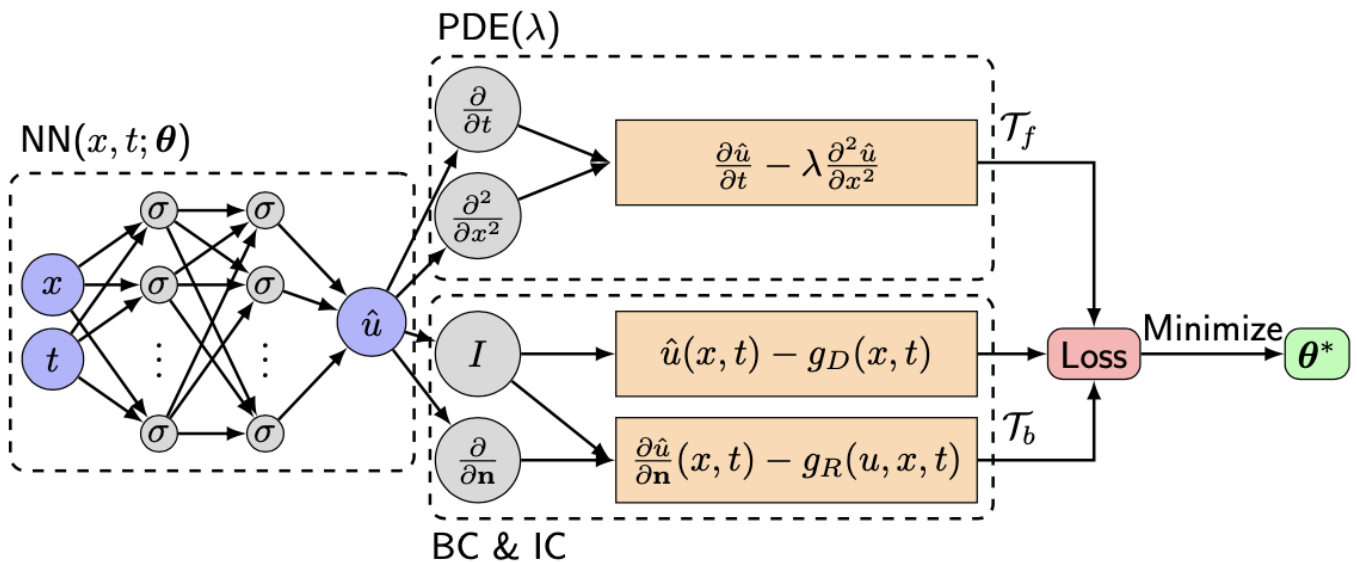
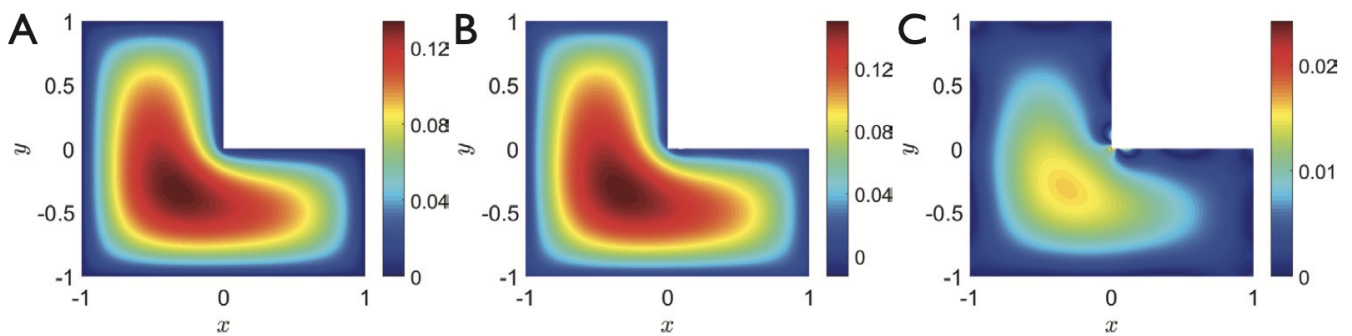


FIG. 1. Schematic of a PINN for solving the diffusion equation  $\frac{\partial u}{\partial t} = \lambda \frac{\partial^2 u}{\partial x^2}$  with mixed boundary conditions (BC)  $u(x, t) = g_D(x, t)$  on  $\Gamma_D \subset \partial\Omega$  and  $\frac{\partial u}{\partial \mathbf{n}}(x, t) = g_R(u, x, t)$  on  $\Gamma_R \subset \partial\Omega$ . The initial condition (IC) is treated as a special type of boundary conditions.  $\mathcal{T}_f$  and  $\mathcal{T}_b$  denote the two sets of residual points for the equation and BC/IC.

Lu, Meng, Mao, Karniadakis <https://epubs.siam.org/doi/pdf/10.1137/19M1274067>  
 Karniadakis et al. <https://www.nature.com/articles/s42254-021-00314-5>

Laplace eqn.



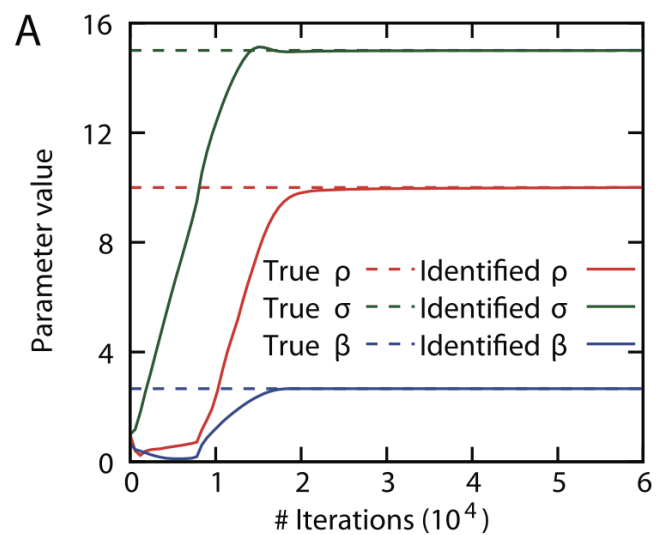
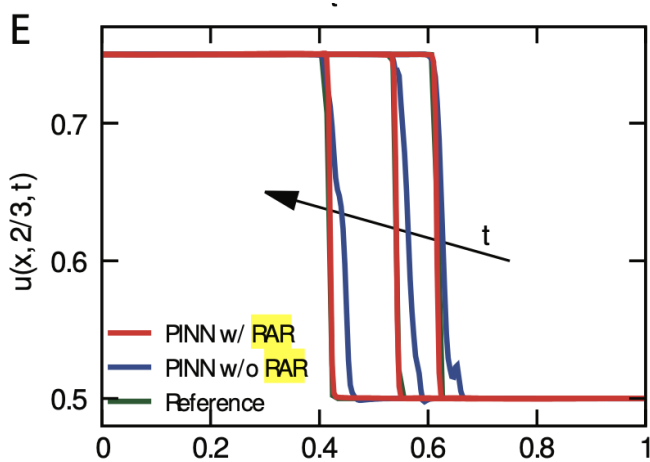
**Fig. 7** Example in subsection 4.1. Comparison of the PINN solution with the solution obtained by using the spectral element method (SEM). (A) The SEM solution  $u_{SEM}$ , (B) the PINN solution  $u_{NN}$ , (C) the absolute error  $|u_{SEM} - u_{NN}|$ .

(SEM – conventional Spectral Element Method)

Lu, Meng, Mao, Karniadakis <https://epubs.siam.org/doi/pdf/10.1137/19M1274067>  
Karniadakis et al. <https://www.nature.com/articles/s42254-021-00314-5>

Burgers eqn. (shock waves)

Inverse problem of the Lorentz system



(RAR – adaptive improvement)

# Conclusions

## Status of ML in physics

- Not long ago: novelty, curiosity, fashion → absolute standard (as Simpson's method, least squares, MC, ... before)
- In physics, ideas and methods largely transplanted from other fields
- Continuous improvements
- Active on-going development: architecture, Hopfield, Boltzmann... – interesting mathematically
- Sea of practical applications, "production phase" of simple to use libraries and dedicated codes
- Libraries and platforms: PyTorch, Keras/TensorFlow (Google), fastai (DL in Python), AutoML (Google cloud), Watson (IBM), SageMaker (Amazon), ...

We need to HL (human-learn) this stuff!



<https://cerncourier.com/a/designing-an-ai-physicist>

Jesse Thaler (MIT):

*... We have gained trust in AI decisions through careful studies of “control regions” and painstaking numerical simulations. As our physics ambitions grow, however, we are using “deeper” networks with more layers and more complicated architectures, which are difficult to validate in the traditional way. And to mitigate 10 to 100-fold increases in computing costs, we are planning to fully integrate AI into data collection, simulation and analysis at the high-luminosity LHC ...*

*To build trust in AI, I believe we need to teach it to think like a physicist ...*

*If we don't exploit the full power of AI, we will not maximize the discovery potential of the LHC and other experiments*

# Thanks!