Machine learning in space physics

Savvas Raptis

Space Physics I
KTH Royal Institute of Technology
10 / 10 / 2022 | 13:00 – 15:00

Introduction to Neural Networks & Applications in Space Physics

Savvas Raptis

Space Physics I
KTH Royal Institute of Technology
10 / 10 / 2022 | 13:00 – 15:00

What is machine learning & A.I ? (1)

Making the computer "learn" without being explicitly programmed

What is machine learning & A.I ? (2)

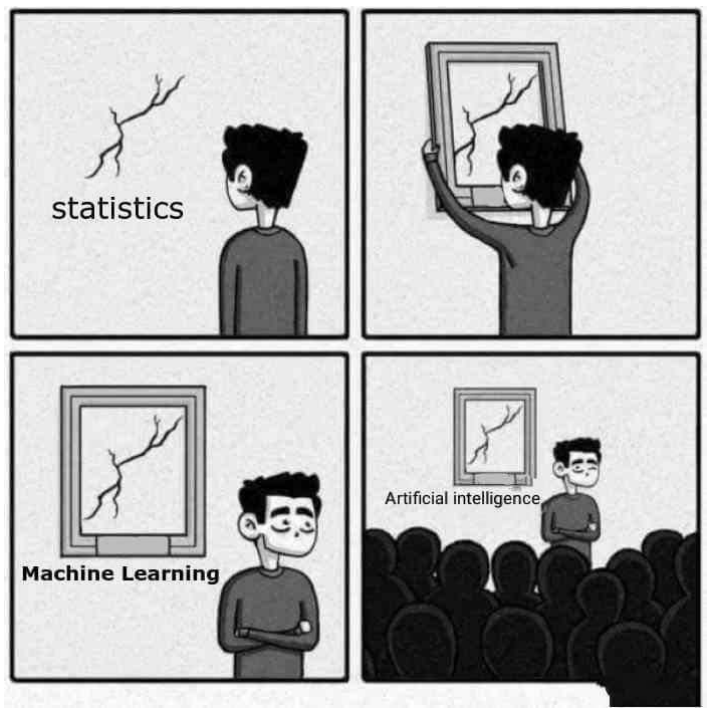
AI (Artificial Intelligence) : A term most of the times used to draw attention (at least for the moment).

ML (Machine Learning) : A program designed to “learn” something based on a set of rules and/or data.

Difference between machine learning and AI:

If it is written in Python, it's probably machine learning

If it is written in PowerPoint, it's probably AI

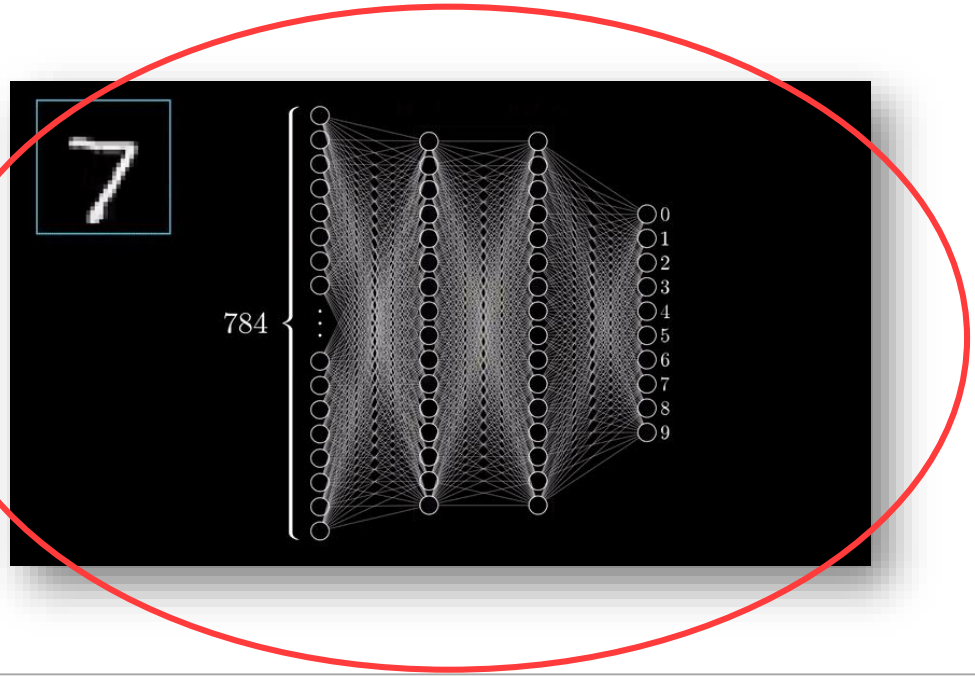


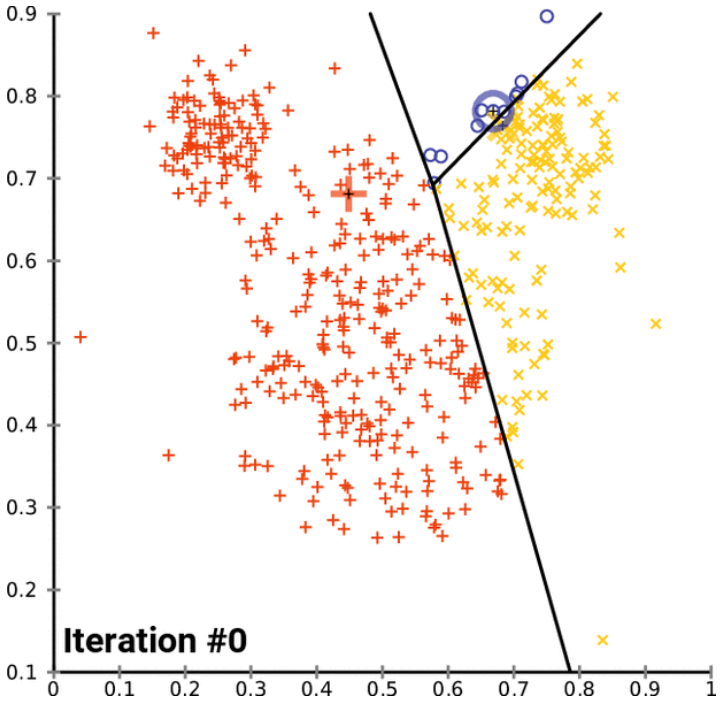
Different categories of Machine Learning

Supervised learning : Given Input and Output - learn the mapping by minimizing an error between the initially random output with the given output. Obtaining a relation.

Unsupervised Learning: Given Input but not output - Either find features or discover patterns.

Reinforcement learning: No data needed, just a value that we want to maximize (e.g. score in game) and a set of actions/environment. The goal is to find a sequence of actions that maximizes the reward.

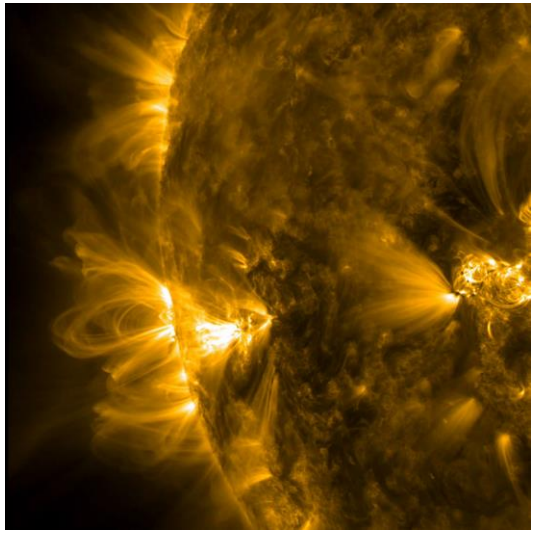


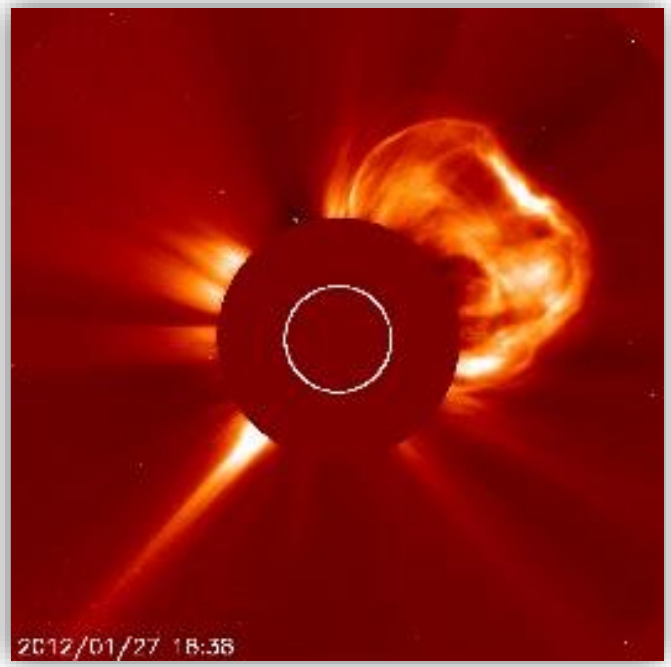


Relevance to Space Physics

Can we predict SEPs
(Solar Energetic Particles) ?

Can we reconstruct coronal loops?

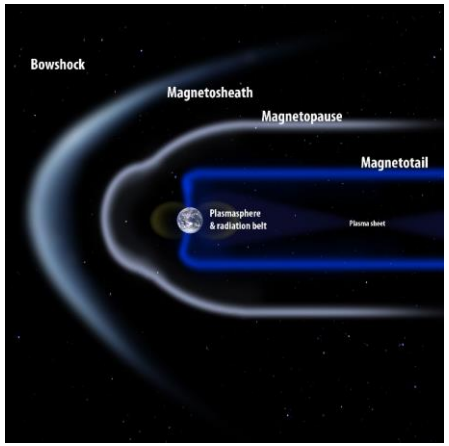


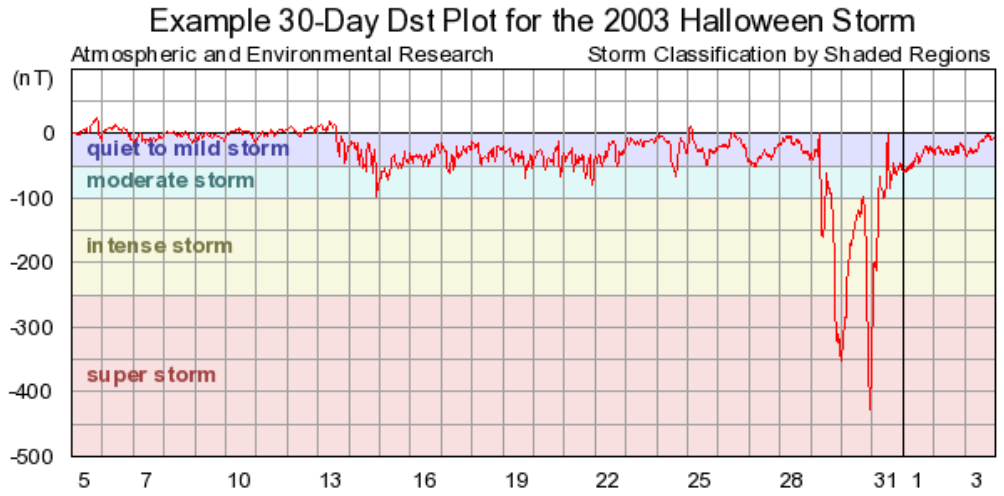


Can we predict CMEs?

Can we predict
Geomagnetic storms?

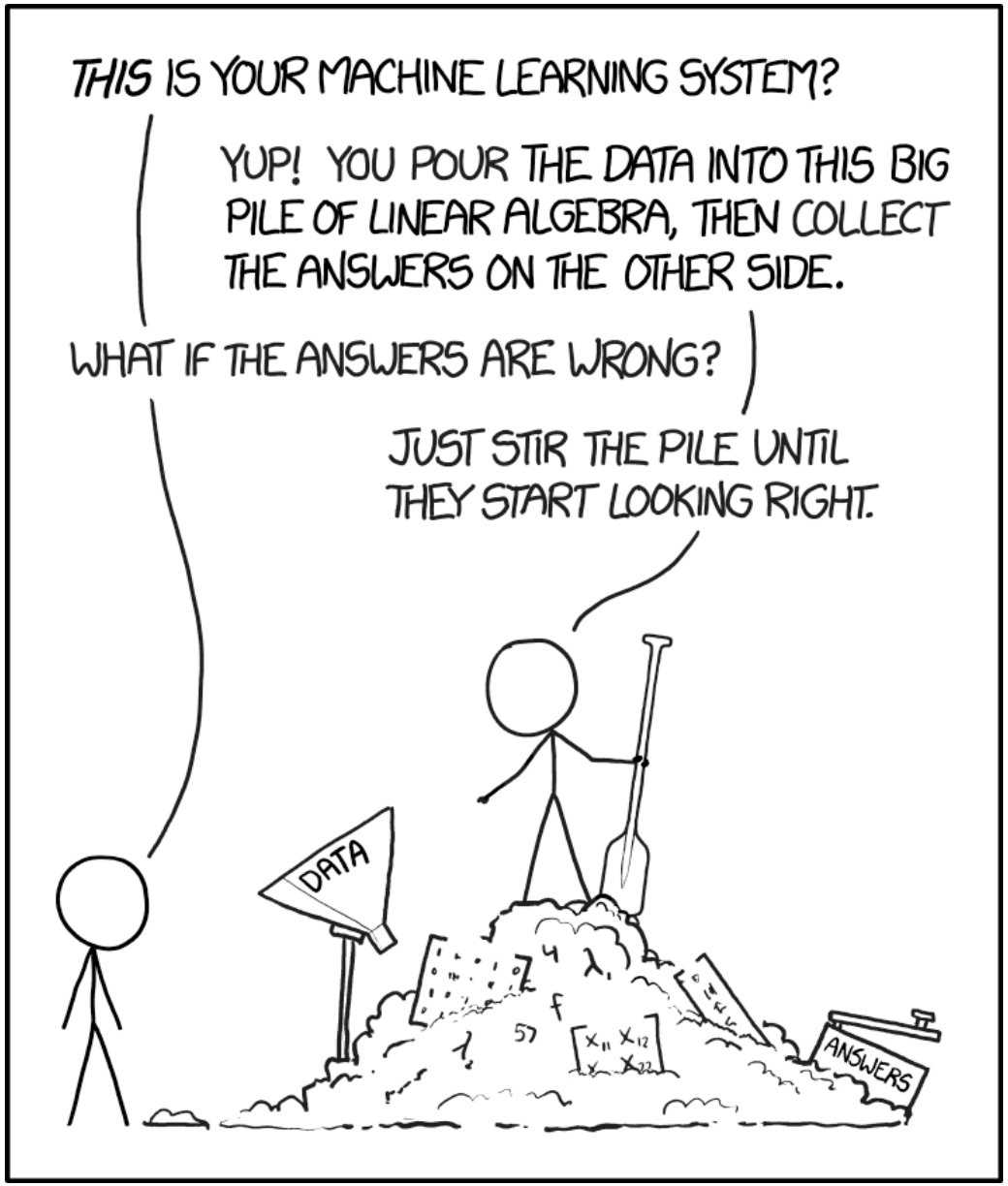
Can we classify plasma
environments?





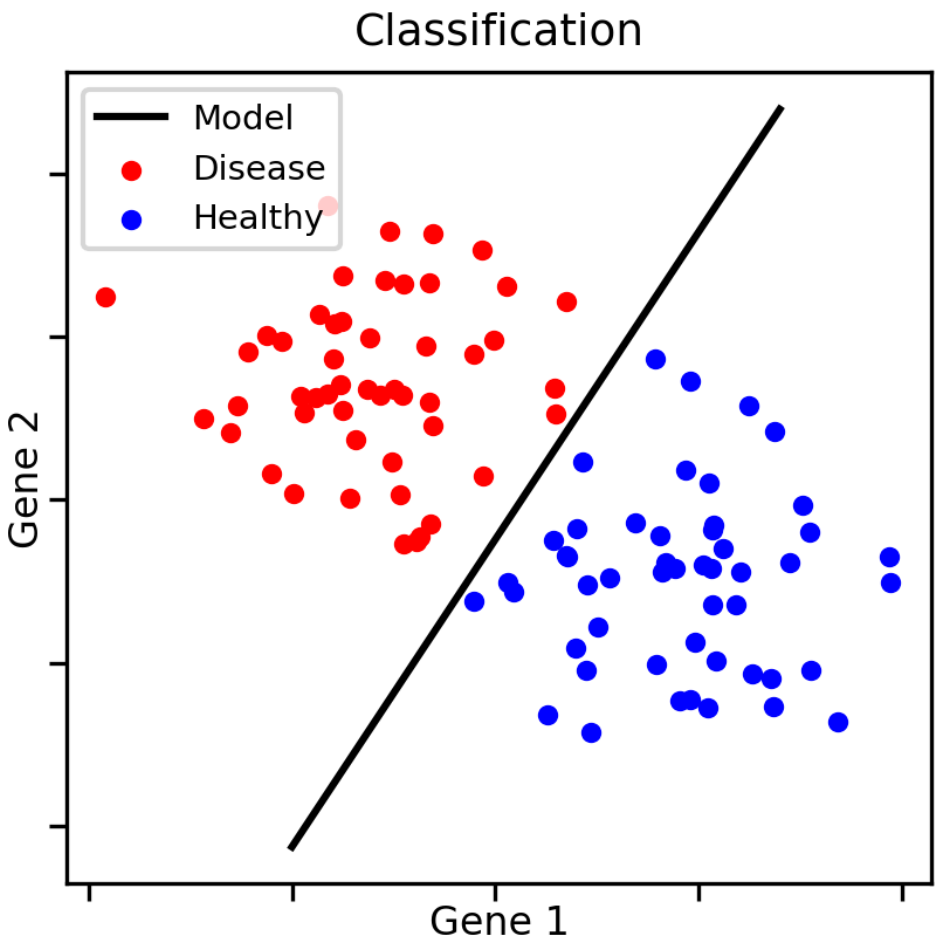
Supervised Learning
using
Neural Networks

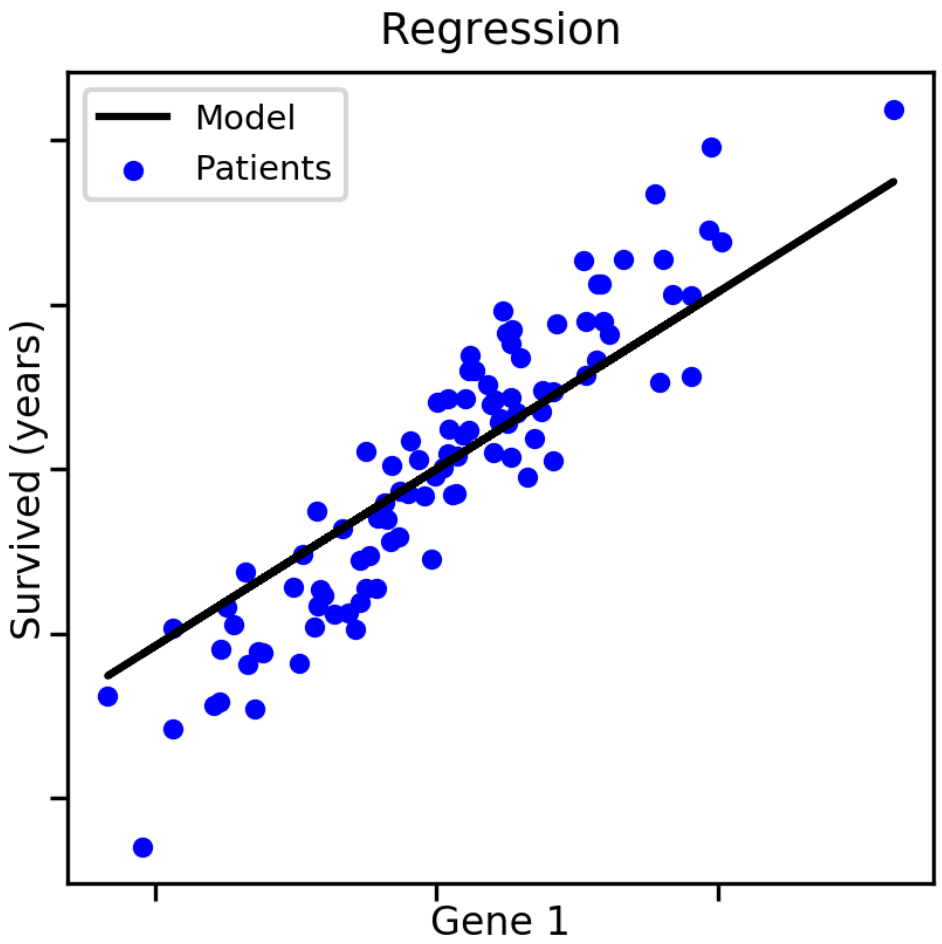
TLDR version



https://xkcd.com/1838/

Type of Machine Learning Problems





Let’s think of a problem (classification)

Data: Thousands of labeled numbers 1-9

Problem: Can we make a model that given the image can recognize the number ?

(one) **Solution**: Neural Networks!

Another problem to think of (regression)

Data: characteristic of a house (e.g., # of rooms)

Problem: What should the price be ?

(one) **Solution**: Neural Networks!

Outline

How to understand Neural Network modeling:

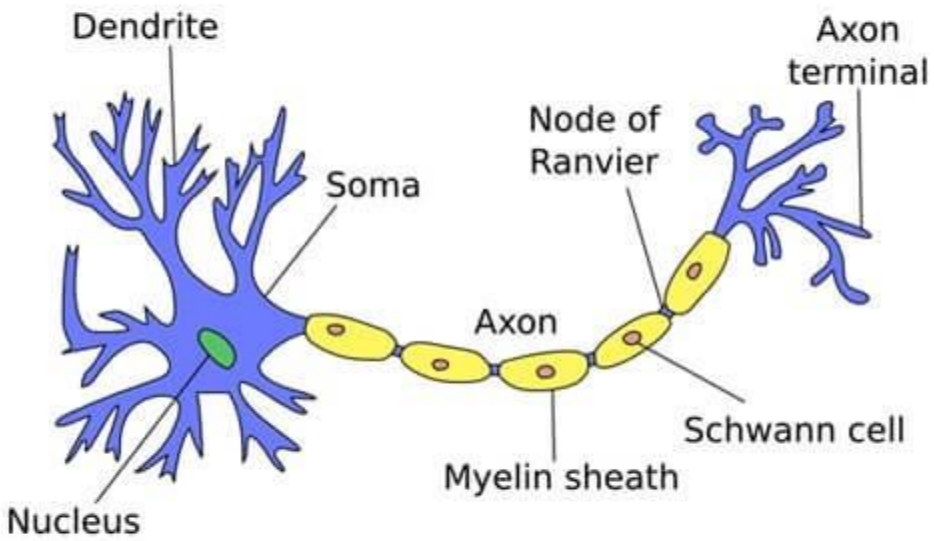
- Concept (basic principles) & Building (what are the components?)
- Train (How does it learn?)
- Evaluate (Did it really learn anything?)

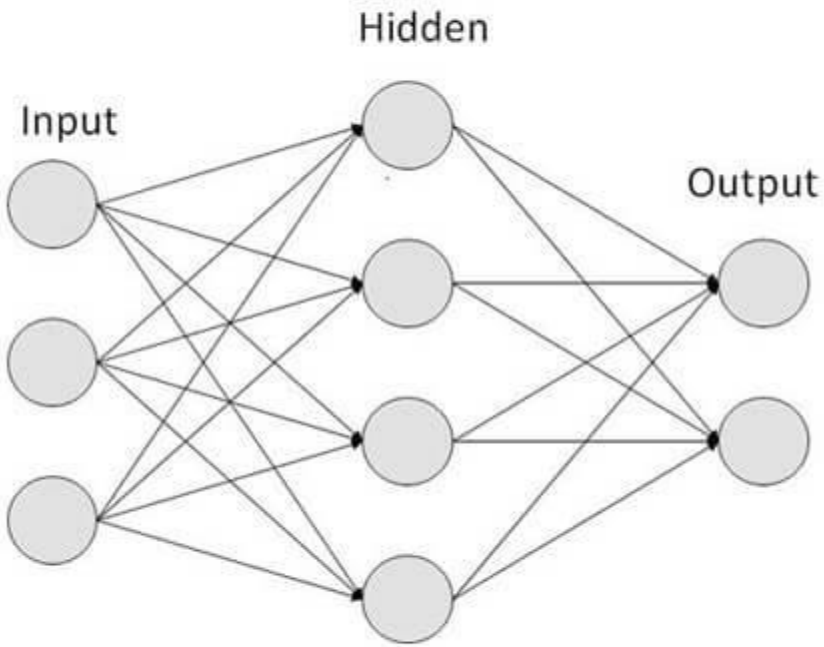
Outline

How to understand Neural Network modeling:

- Concept (basic principles) & Building (what are the components?)
- Train (How does it learn?)
- Evaluate (Did it really learn anything?)

Biological vs Artificial Neural Networks





source: https://medium.com/@abhyudayj.isme1820/neural-networks-42b34deacc1b

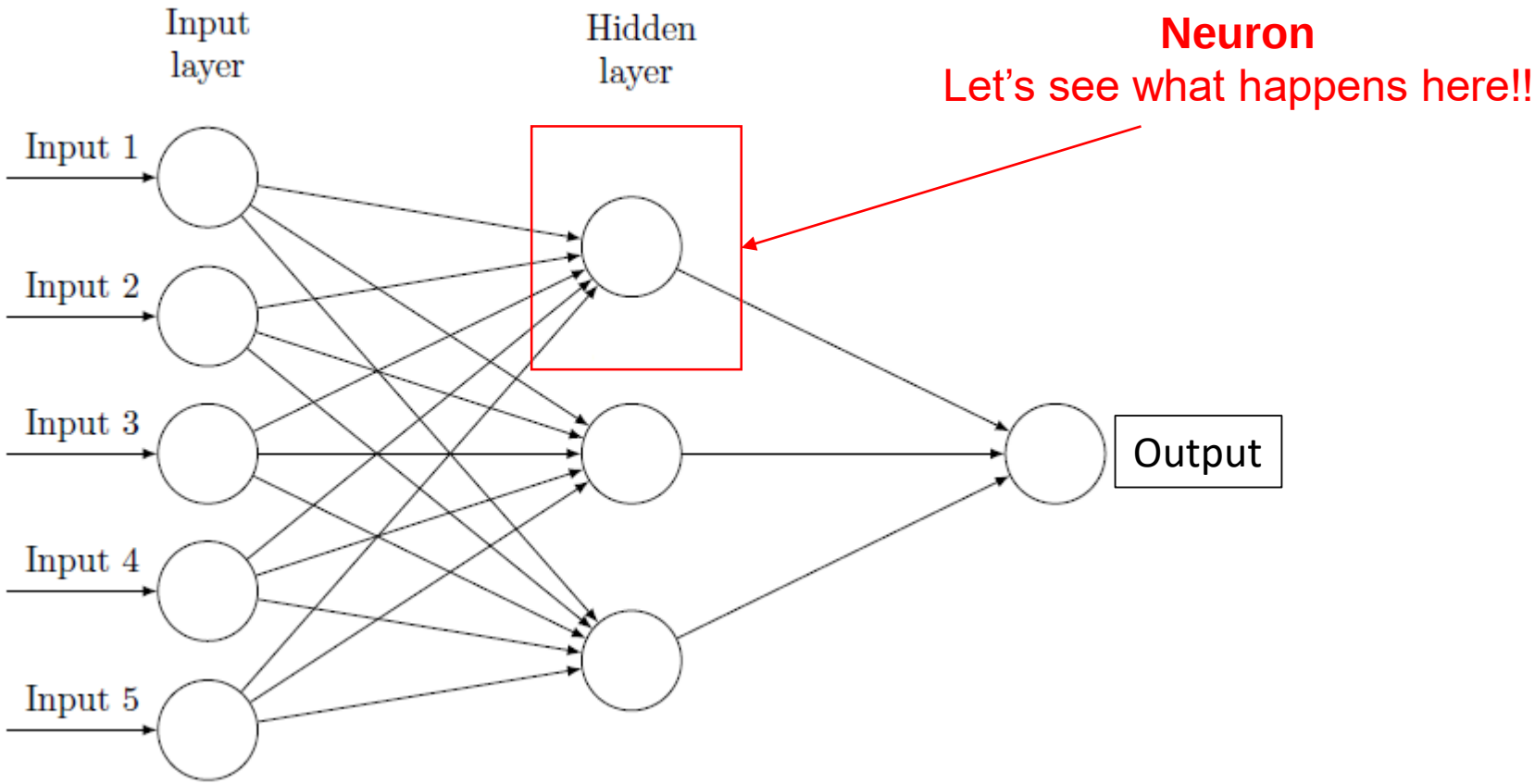
Neural Networks

<u>Input/Output pairs</u>

e.g.

Output = Price of house

Input1 = Square meters
Input2 = # rooms
Input3 = Proximity to center
Input4 = Age it was built
...



A Neural Network Input and Output

$$y = f(x_1 w_1 + x_2 w_2 + \cdots + x_n w_n + b)$$

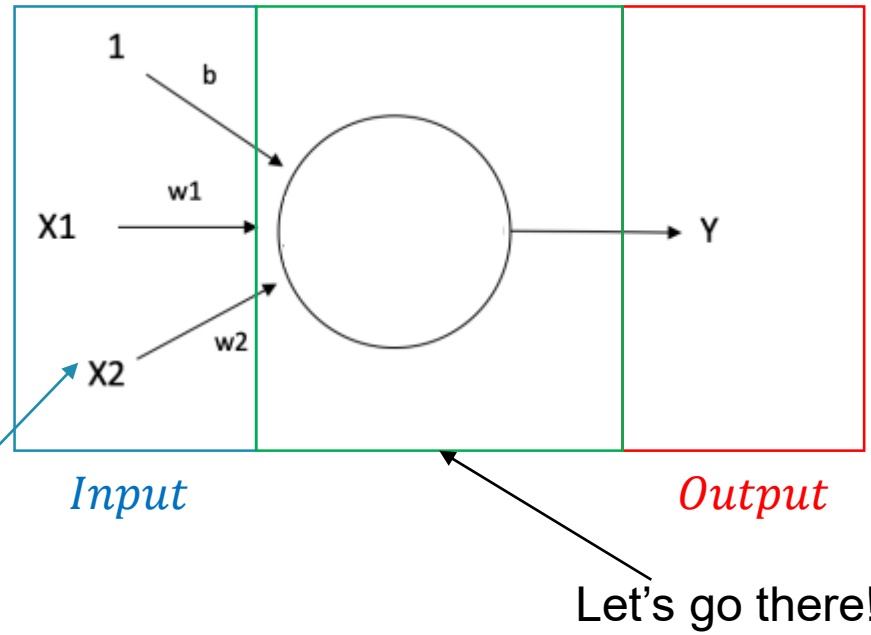
y : Output of Neuron

x_i: Inputs of Neuron

w_i: Weights of each Input

b : Bias for each neuron

} Random Numbers

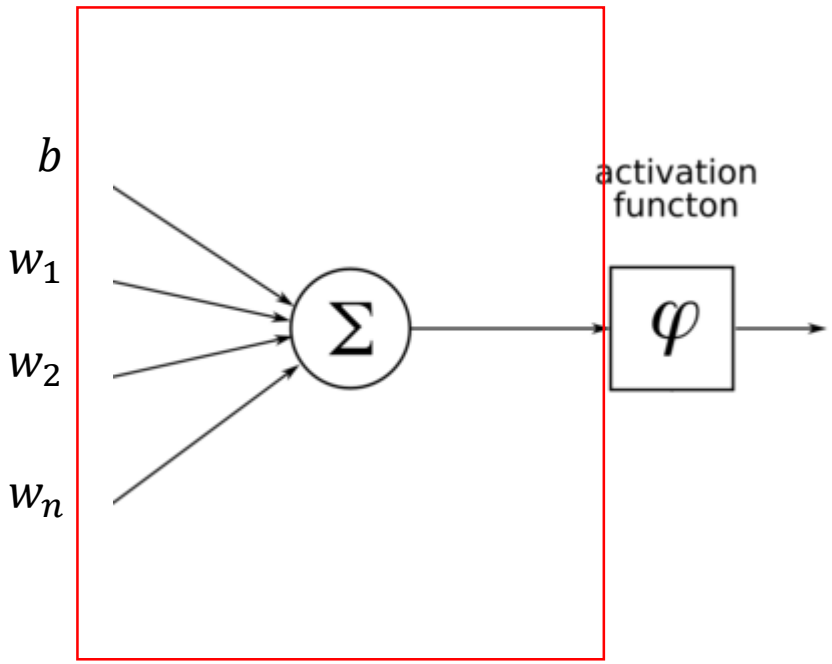


For example

x_1= Square meters - $w_1 = 0.7$ (more important)

x_2= Proximity to center - $w_2 = 0.3$ (less important)

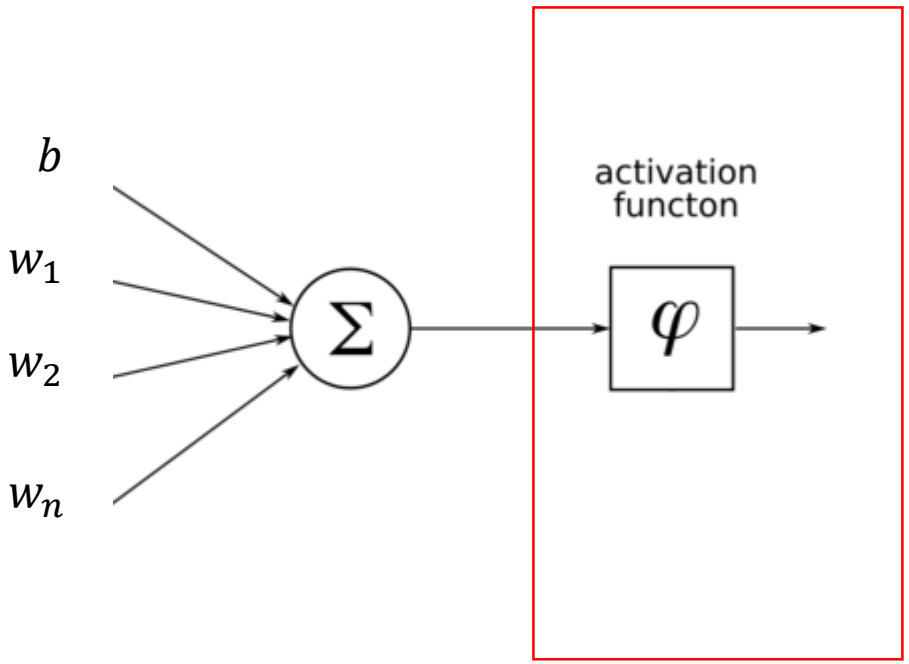
Inside the Neuron



Sum of all Data(x_i), Weights (w_i) and Biases (b)

$$y = \sum x_i w_i + b$$

Activation Function



Apply $f(y)$ depending on **goal**

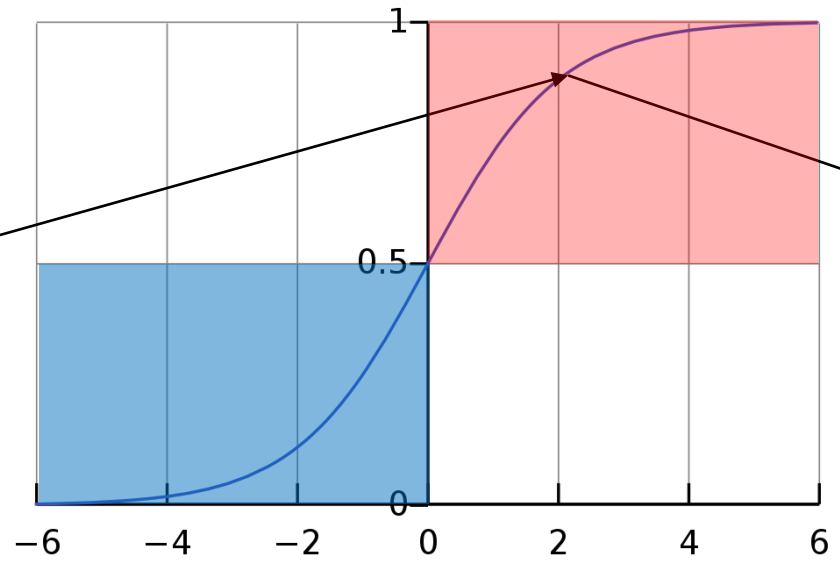
Activation function Examples

Goal: Classification (e.g., 0/1)

e.g. Should I buy the house?

Yes (class 1)
No (class 0)

$$\sum x_i w_i + b = 2$$



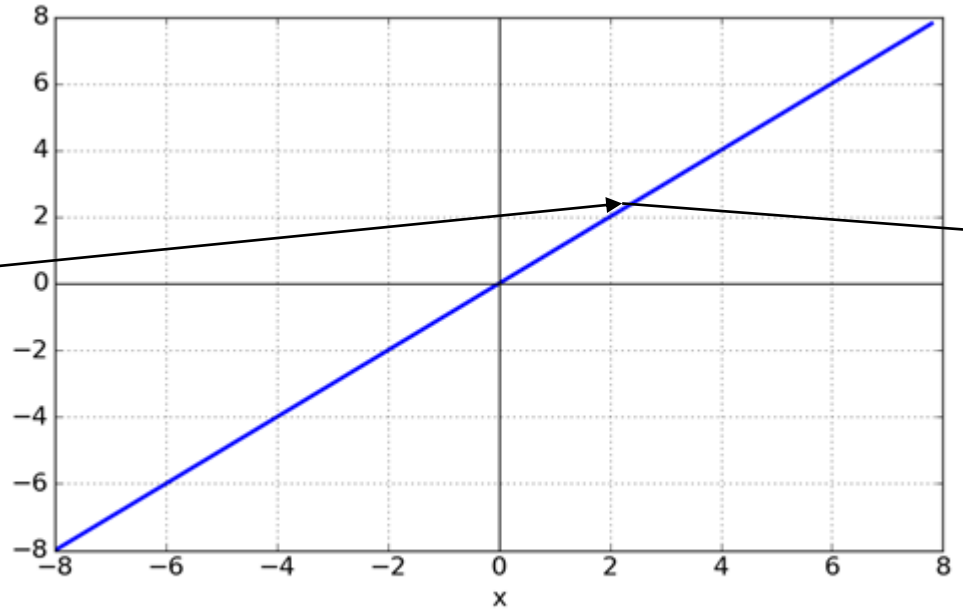
$$y = 0.8 \rightarrow class\ 1\ (\text{buy})$$

Activation function Examples

Goal: Regression (Continuous value)

e.g. How many million SEK will a house cost ?

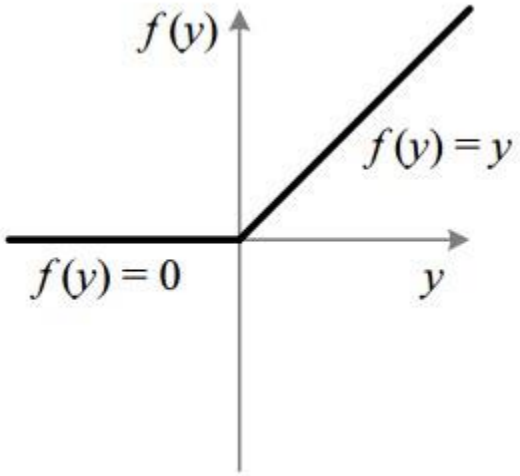
$$\sum x_i w_i + b = 2$$



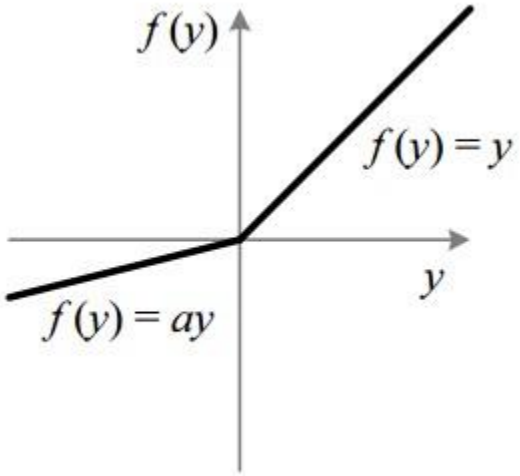
$y = 2$ million SEK price

Hidden layer activation functions

Goal: Add complexity to our mapping



Rectified Linear Units (ReLU)

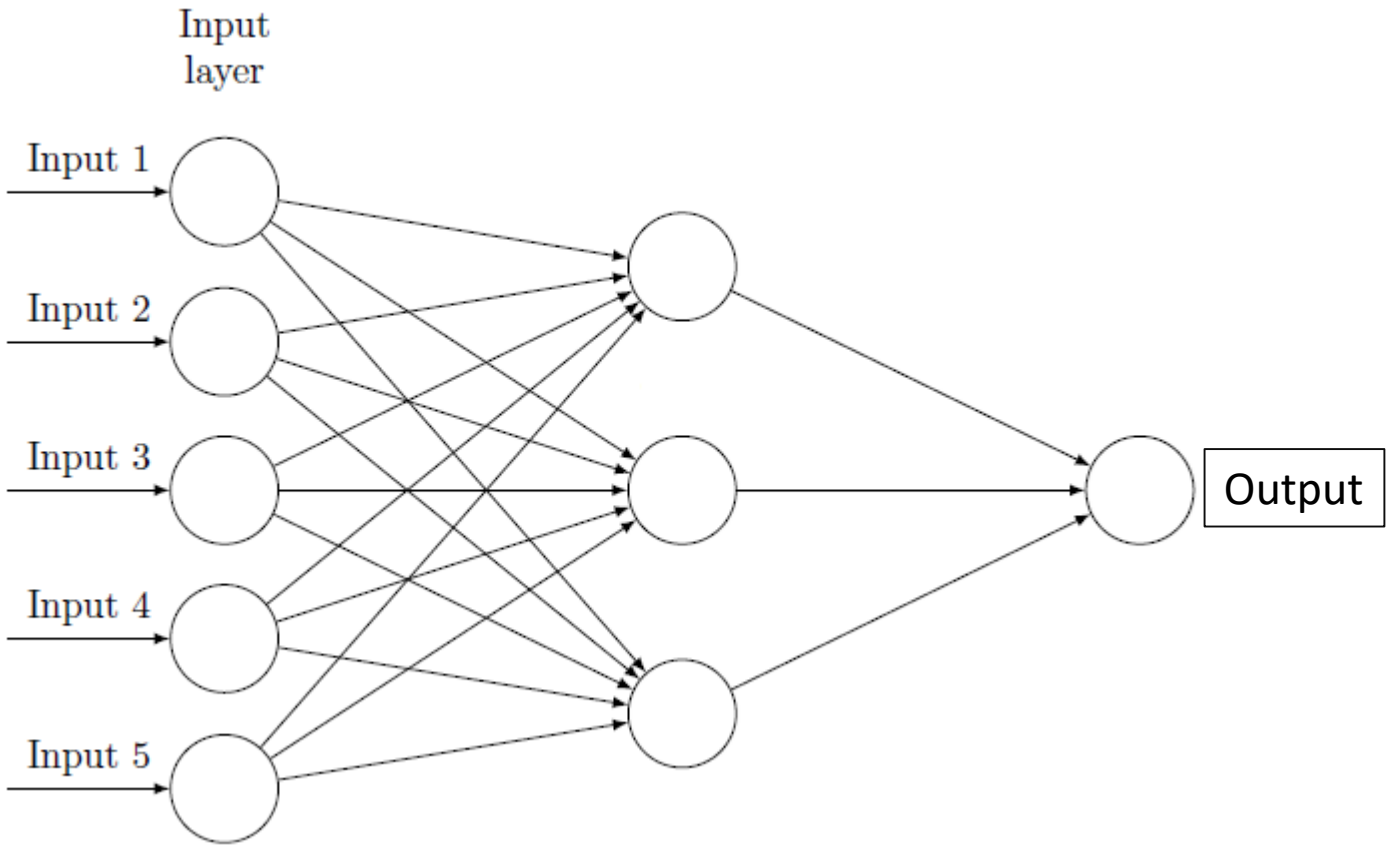


LeakyReLU

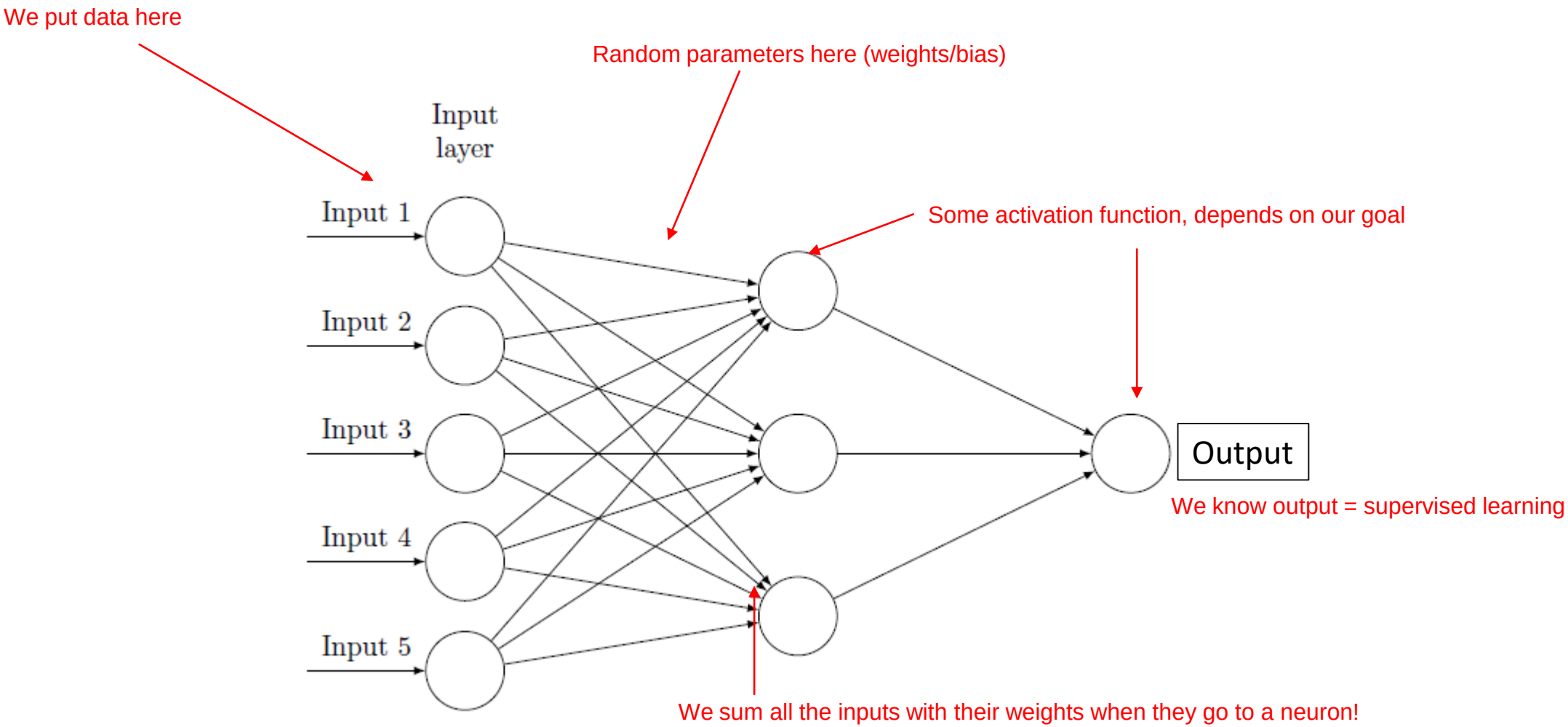
Question time!

What do we know so far ?

Neural Networks



Neural Networks



Outline

How to understand Neural Network modeling:

- Concept (basic principles) & Building (what are the components?)

Answer: ….

- Train (How does it learn?)
- Evaluate (Did it really learn anything?)

Outline

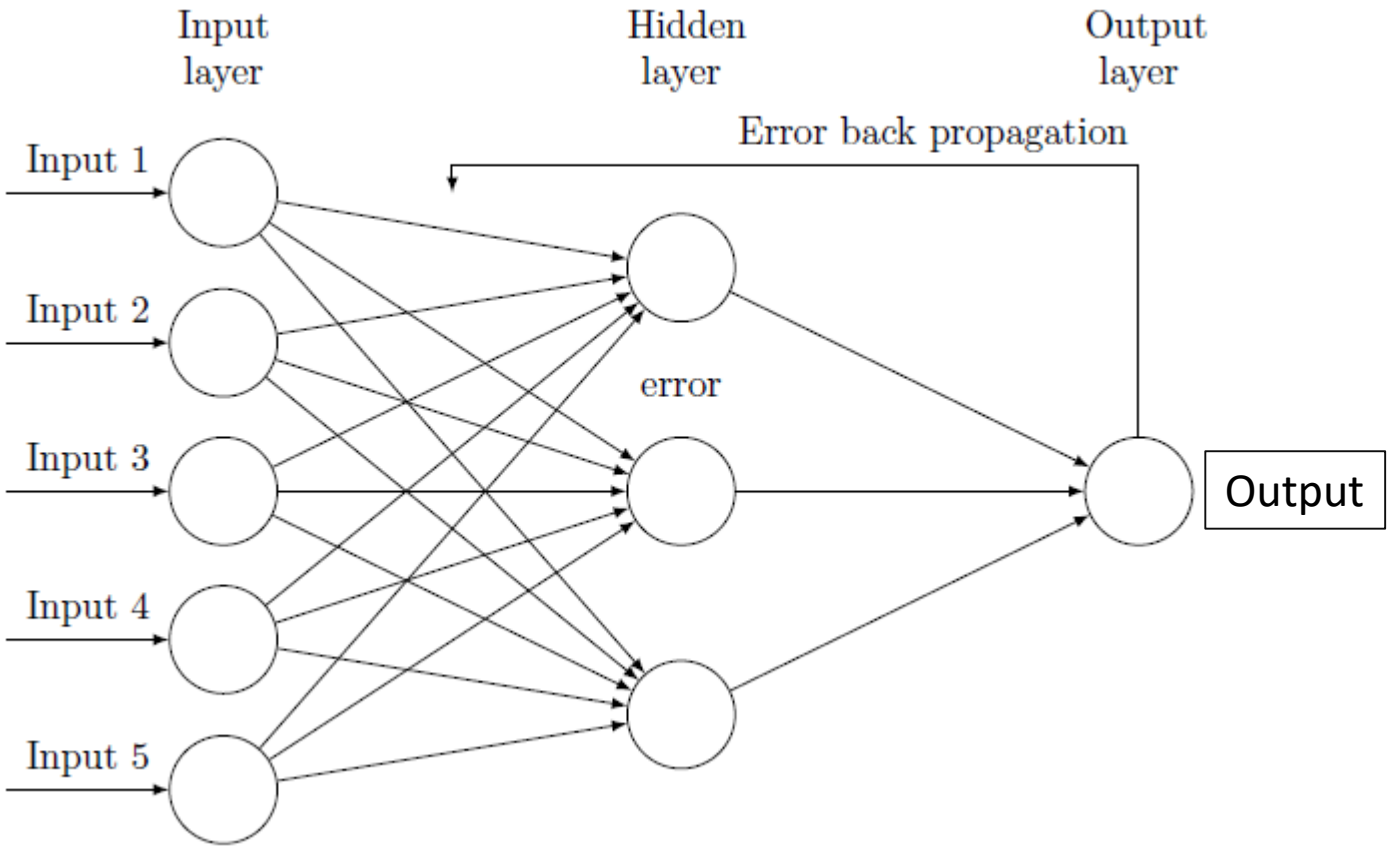
How to understand Neural Network modeling:

- Concept (basic principles) & Building (what are the components?)

Answer: Yes! We know that now! We have a randomly parametrized model….. But ok, then what ? It should just spit out random results now right ?

- Train (How does it learn?)
- Evaluate (Did it really learn anything?)

Neural Networks & Backpropagation



- *Random parameters = random result*
- *Error back propagation = change parameters based on data = modeling*

Cost (error) function

For example, mean squared error:

Output (initially random, since w, b are random)

desired output (known)

$$J = \frac{1}{N}\sum_{i=1}^{N}(Y' - Y)^2$$

The goal is to minimize/maximize a cost function.

Question: How do we minimize this? What do we have to do when we are asked to find the minimum of a function ?

With an algorithm that uses the minima of the function: **Gradient Descent.**

2 things we care about

Where to go and how fast to go there ?

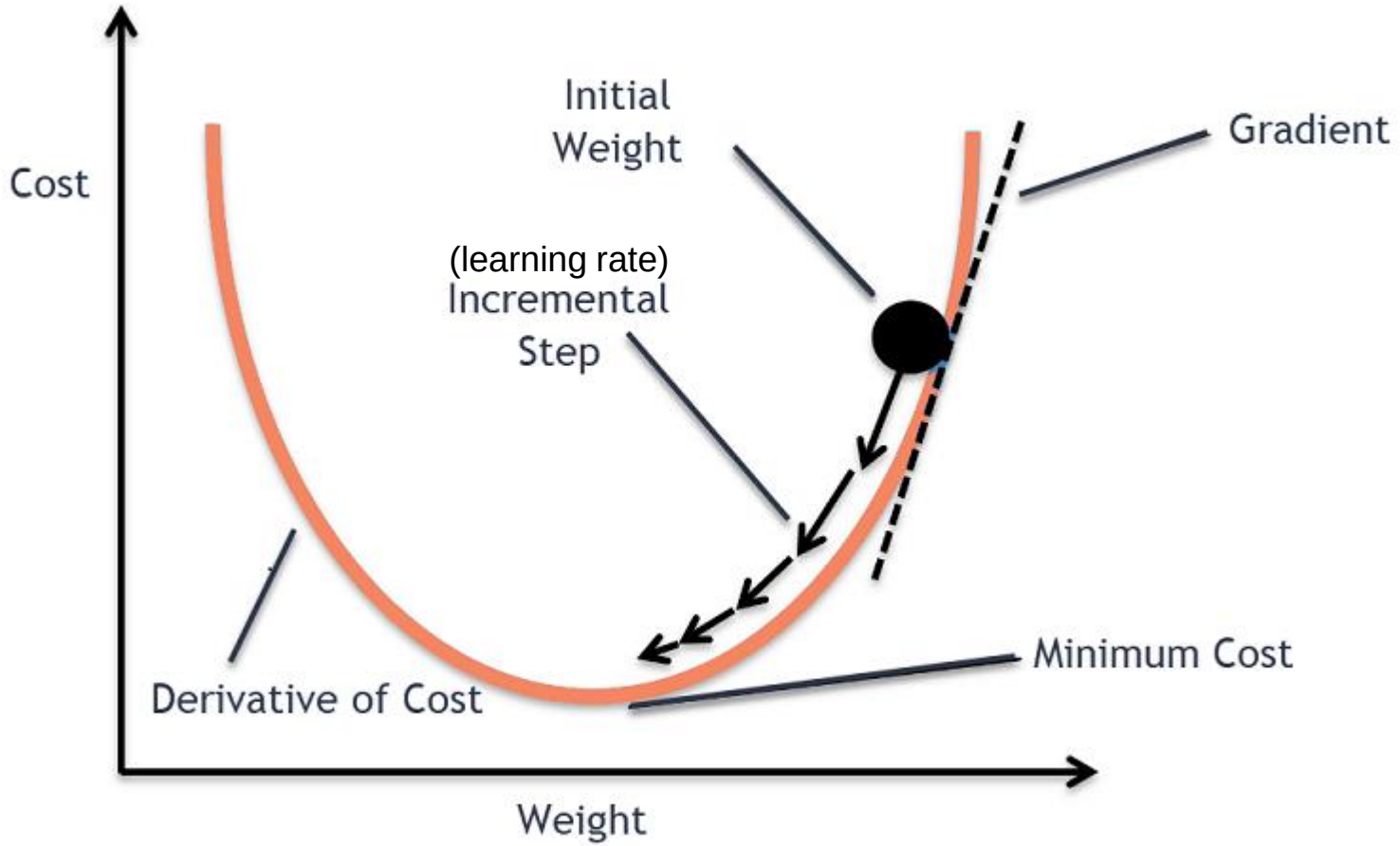
Where = Opposite of derivative/slope of the error

$$\theta_j = \theta_j - a \frac{\partial}{\partial \theta_j} J$$

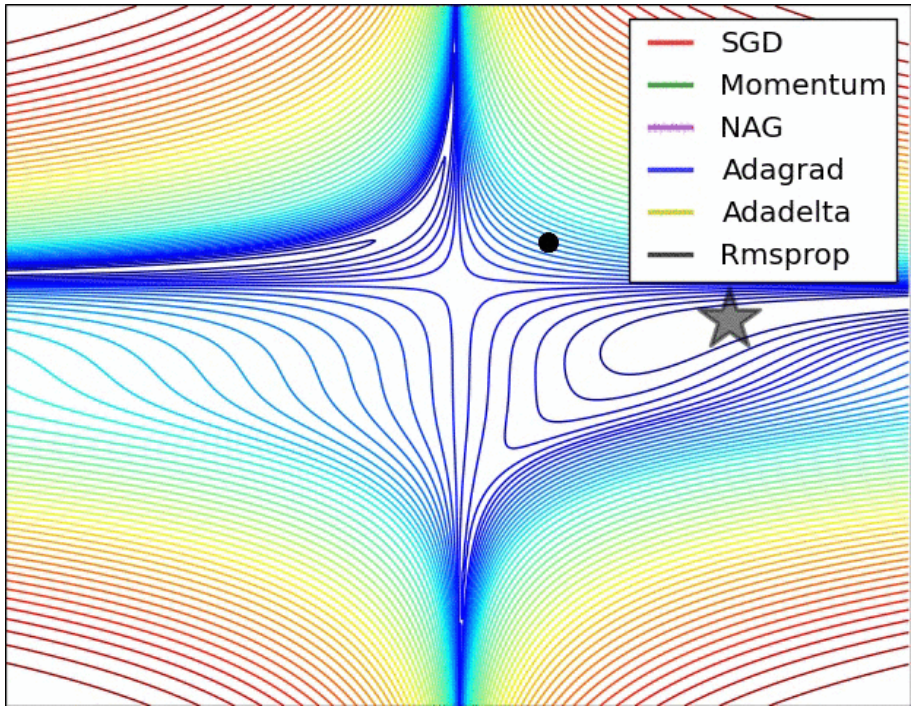
Where, j = cost function

How fast = Learning rate (numerical factor)
https://developers.google.com/machine-learning/crash-course/fitter/graph

Error Back Propagation (Gradient Descent)

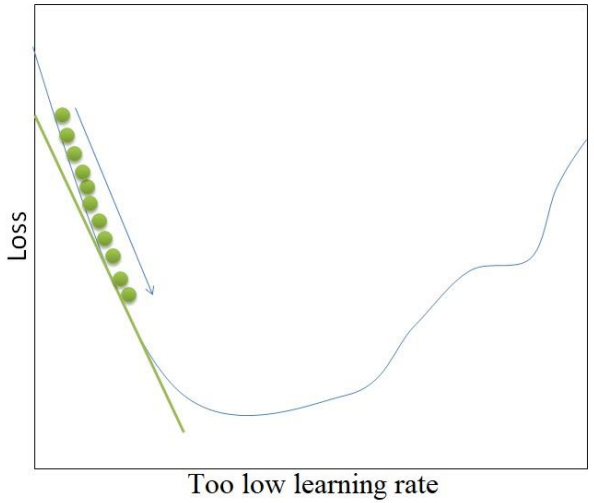


Automatically done by
“optimizers”

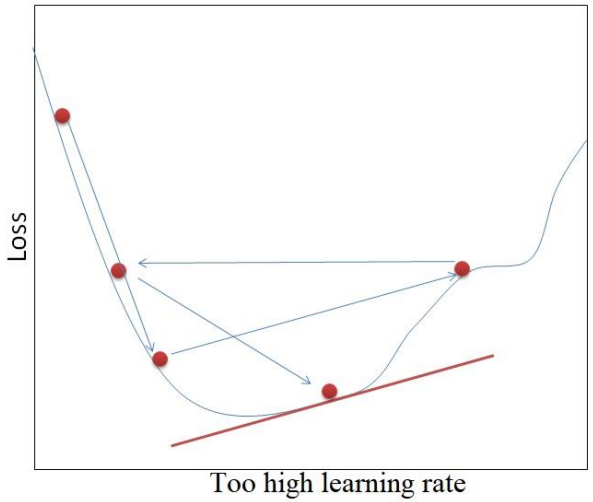


Back-propagation = calculating the derivatives.
Gradient descent = Descending through the gradient to change parameters.

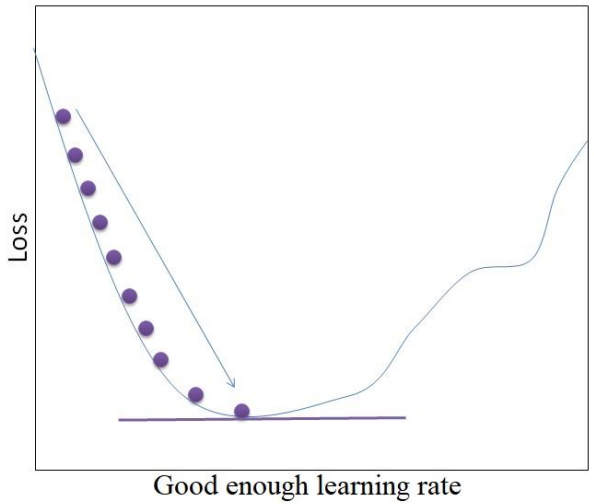
Why Optimizers? (learning rate + momentum)



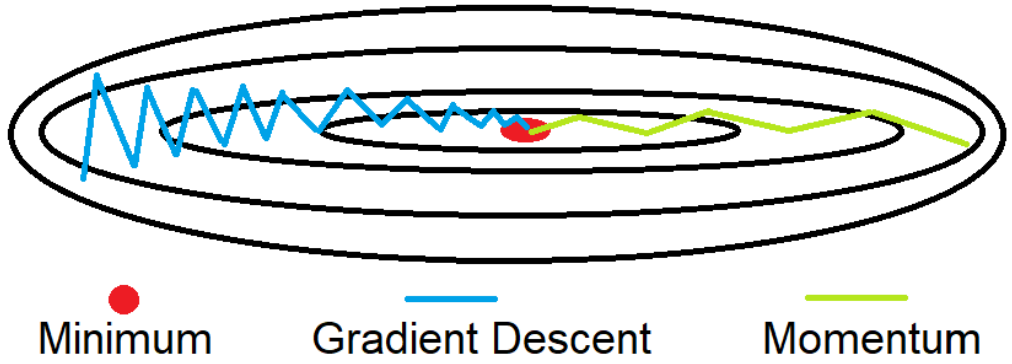
Too low learning rate



Too high learning rate



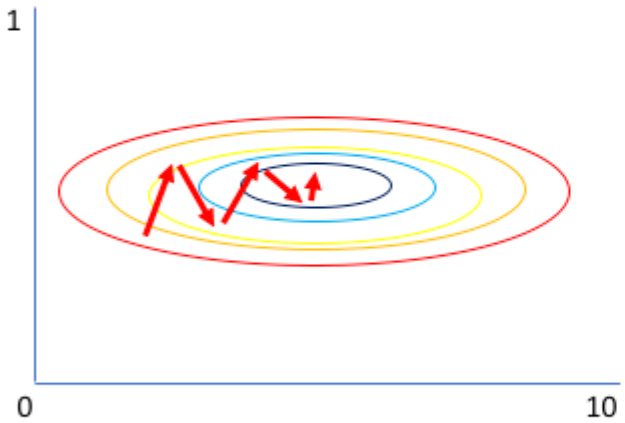
Good enough learning rate



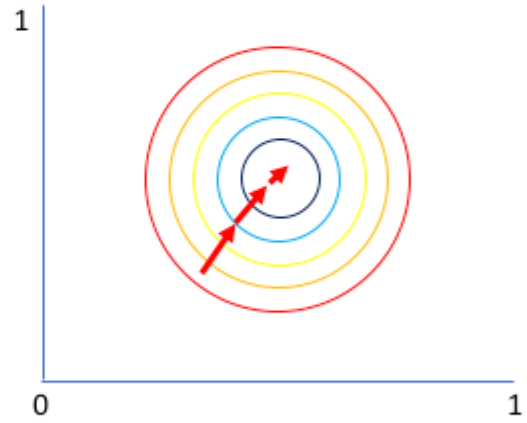
Source: https://www.andreaperlato.com/aipost/gradient-descent-with-momentum/

Normalization of data – Features

Why normalize?



Gradient of larger parameter dominates the update



Both parameters can be updated in equal proportions

e.g.,

Number of rooms
[0 … 5]

Square meters
[0 … 300]

Outline

How to understand Neural Network modeling:

- ~~Concept (basic principles) & Building (what are the components?)~~

Answer: Yes! We know that now! We have a randomly parametrized model….. But ok, then what ? It should just spit out random results now right ?

- Train (How does it learn?)

Answer: …

- Evaluate (Did it really learn anything?)

Outline

How to understand Neural Network modeling:

- ~~Concept (basic principles) & Building (what are the components?)~~

Answer: Yes! We know that now! We have a randomly parametrized model….. But ok, then what ? It should just spit out random results now right ?

- Train (How does it learn?)

Answer: Now we calculate the derivative of the error and we change the parameters of the network. Sure… But how do we test that thing ? I mean… It learned something that we explicitly trained. But everyone can do that… is it smart enough to deal with brand new information ?

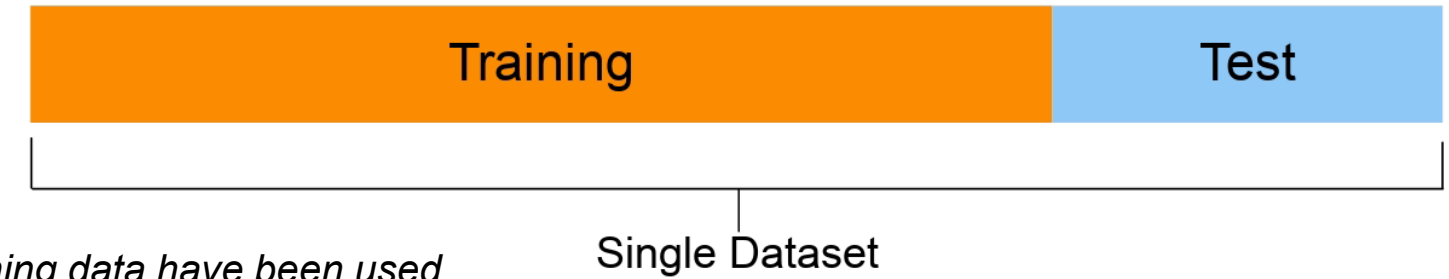
- Evaluate (Did it really learn anything?)

Train – Test – Validation sets

Question

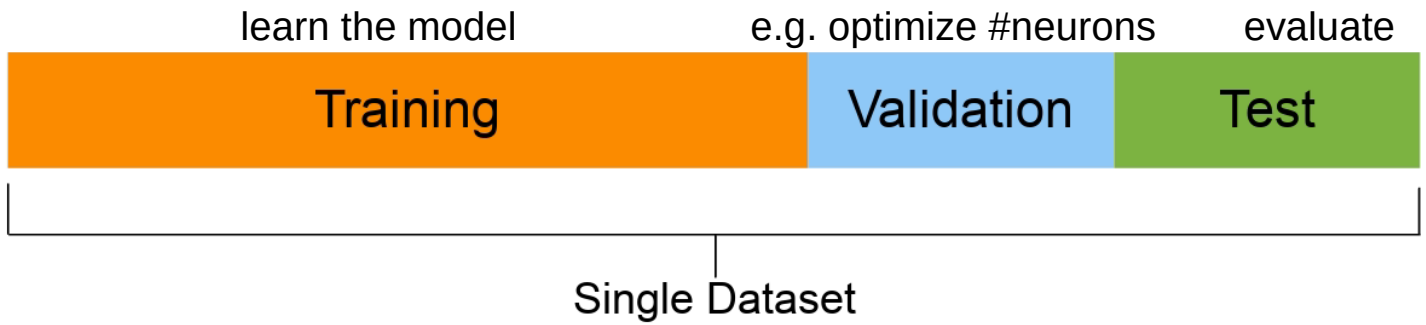
How do we evaluate the model now ??

A

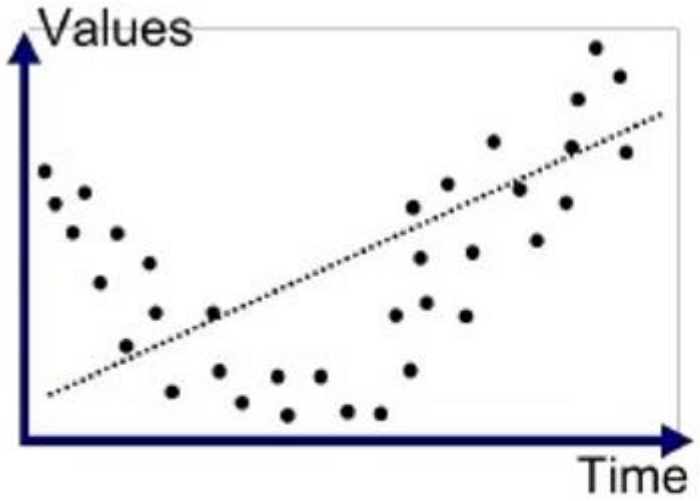


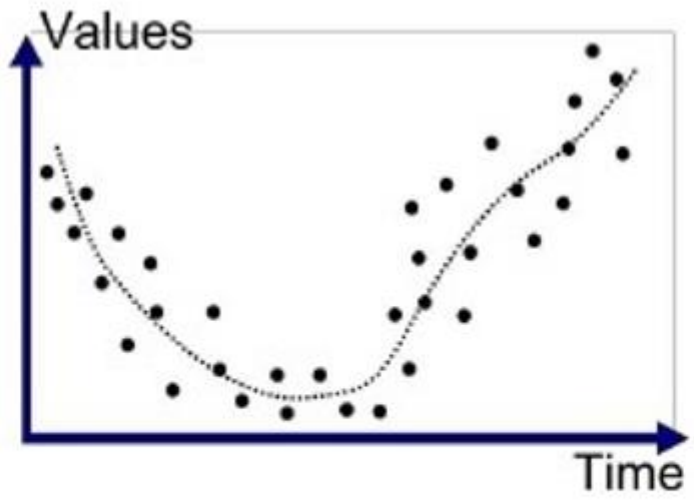
1 “epoch” = 1 time training data have been used.

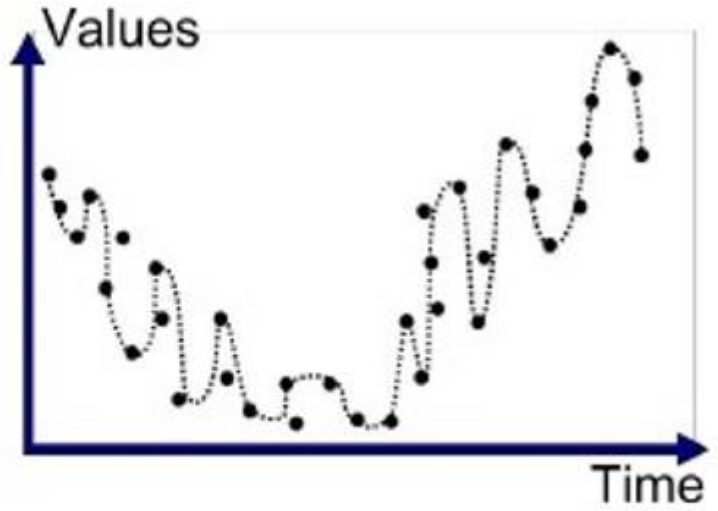
B



Overfitting



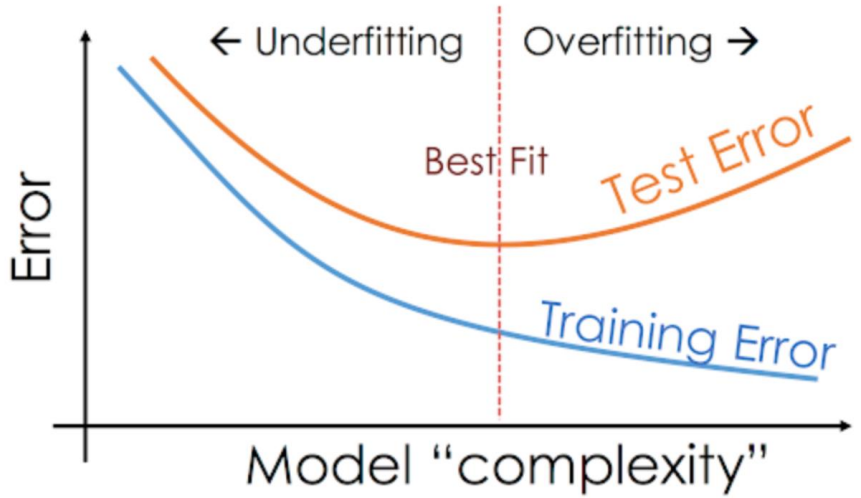




Underfitted

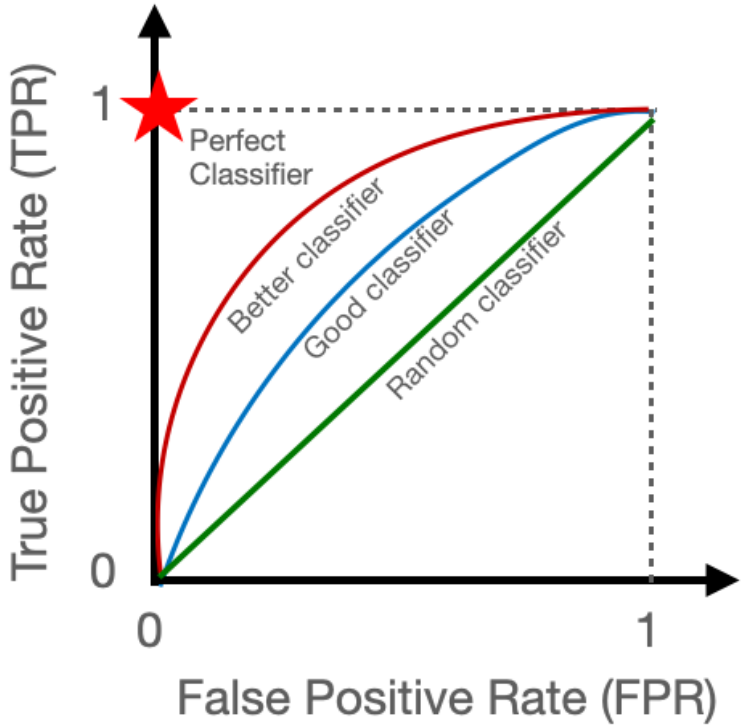
Good Fit/Robust

Overfitted



Evaluation Metrics (Classification)

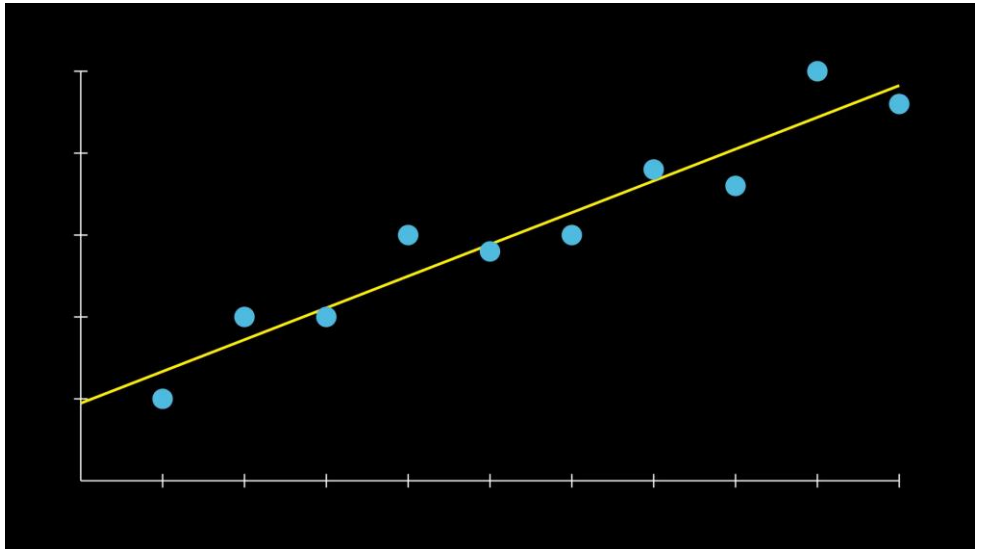
	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP



Metric	Formula
True positive rate, recall	$\frac{TP}{TP+FN}$
False positive rate	$\frac{FP}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
F-measure	$\frac{2 \cdot precision \cdot recall}{precision + recall}$

Evaluation Metrics (Regression)

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$



Mean squared error	$\mathrm{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\mathrm{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\mathrm{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Source: https://gist.github.com/thomasnield

Outline

How to understand Neural Network modeling:

- ~~Concept (basic principles) & Building (what are the components?)~~

Answer: Yes! We know that now! We add a lot of stuff, to a randomly parametrized model….. But ok, then what ? It should just spit out random result now right ?

- ~~Train (How does it learn?)~~

Answer: Ok Ok, we got it. Now we calculate the derivative and by comparing the initially random output to the one we wanted, we change the parameters of the network. Sure… But ok, how do we test that thing ? I mean… It learned something that was taught. But everyone can do that… is it smart enough to deal with brand new information ?

- Evaluate (Did it really learn anything?)

Answer: …

Outline

How to understand Neural Network modeling:

- ~~Concept (basic principles) & Building (what are the components?)~~

Answer: Yes! We know that now! We add a lot of stuff, to a randomly parametrized model….. But ok, then what ? It should just spit out random result now right ?

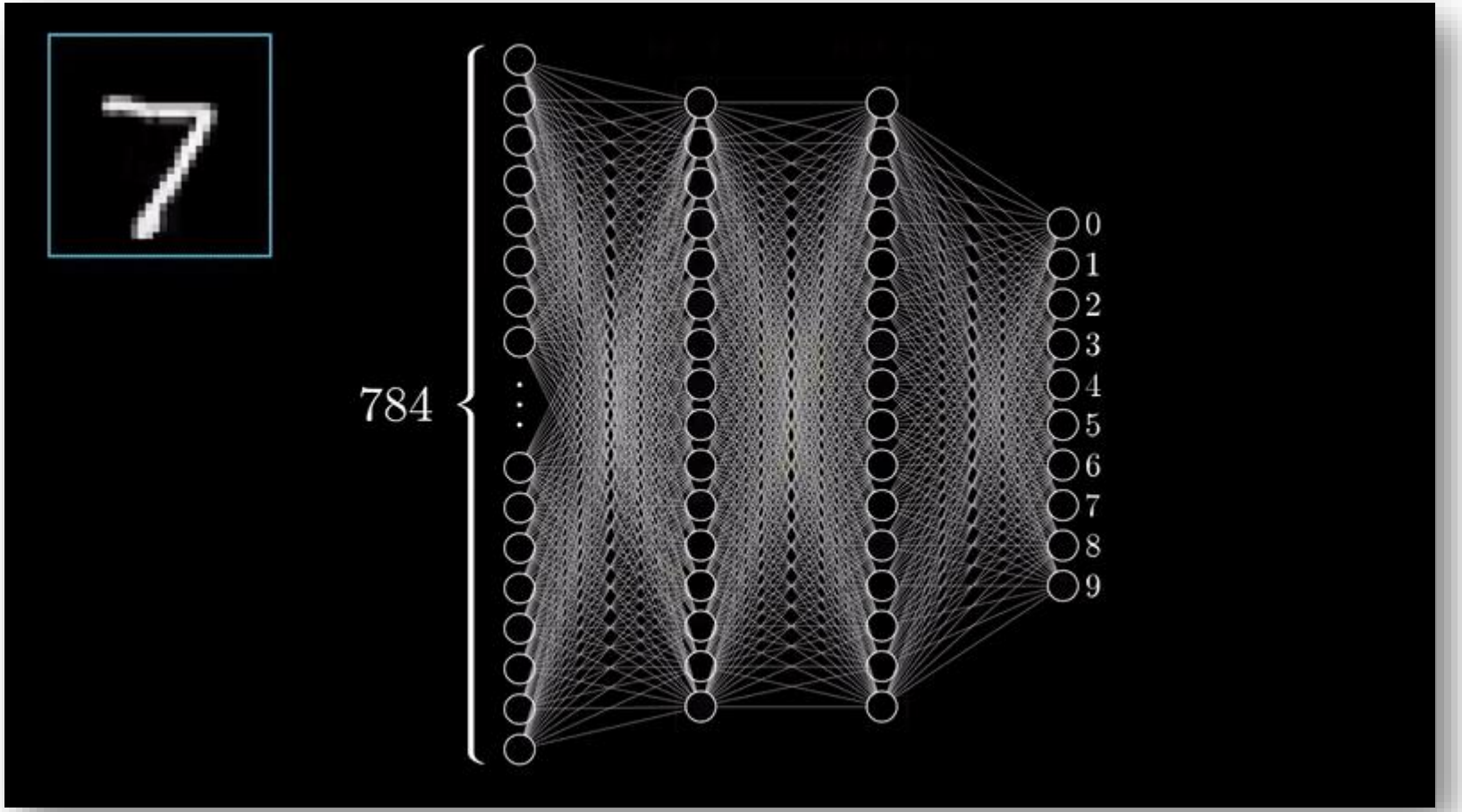
- ~~Train (How does it learn?)~~

Answer: Ok Ok, we got it. Now we calculate the derivative and by comparing the initially random output to the one we wanted, we change the parameters of the network. Sure… But ok, how do we test that thing ? I mean… It learned something that was taught. But everyone can do that… is it smart enough to deal with brand new information ?

- Evaluate (Did it really learn anything?)

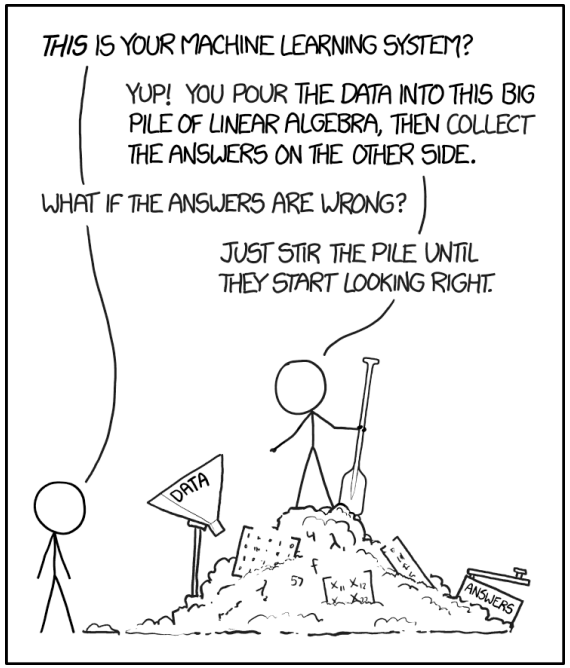
Answer: Ok, so we put data, we change the parameters based on these and then we test with different data to see if the model is smart enough to pick the correct answer. Cool! Then we use different metrics to quantify what happened.

A Trained Neural Network



*Video Courtesy**: 3Blue1Brown** (Check him on YouTube!)

Play time



http://playground.tensorflow.org

We can try to “stir the pile” with:

-Extra input
-Extra veurons
-Extra layers

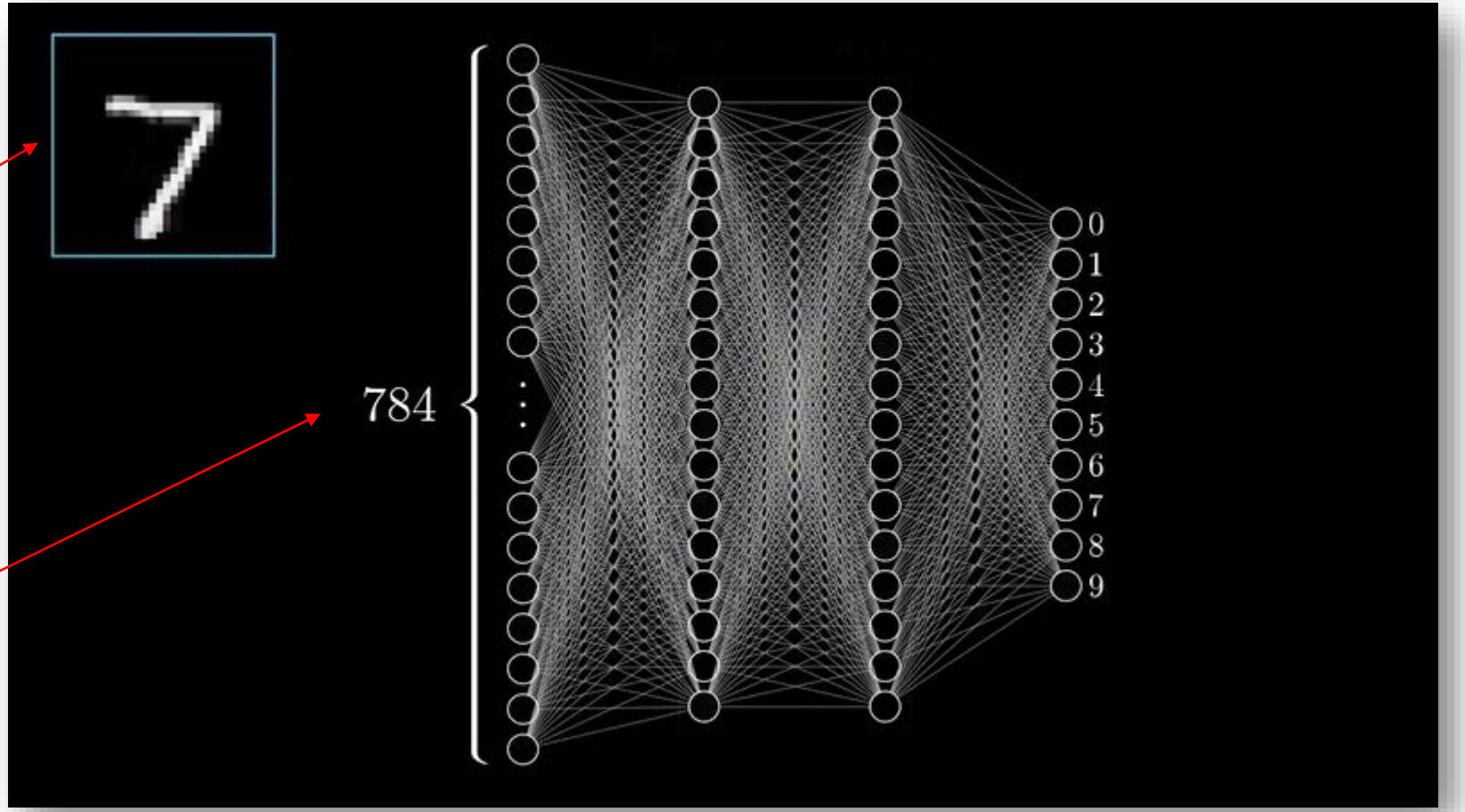
But wait, there is more !

See anything weird ?

28 x 28 image

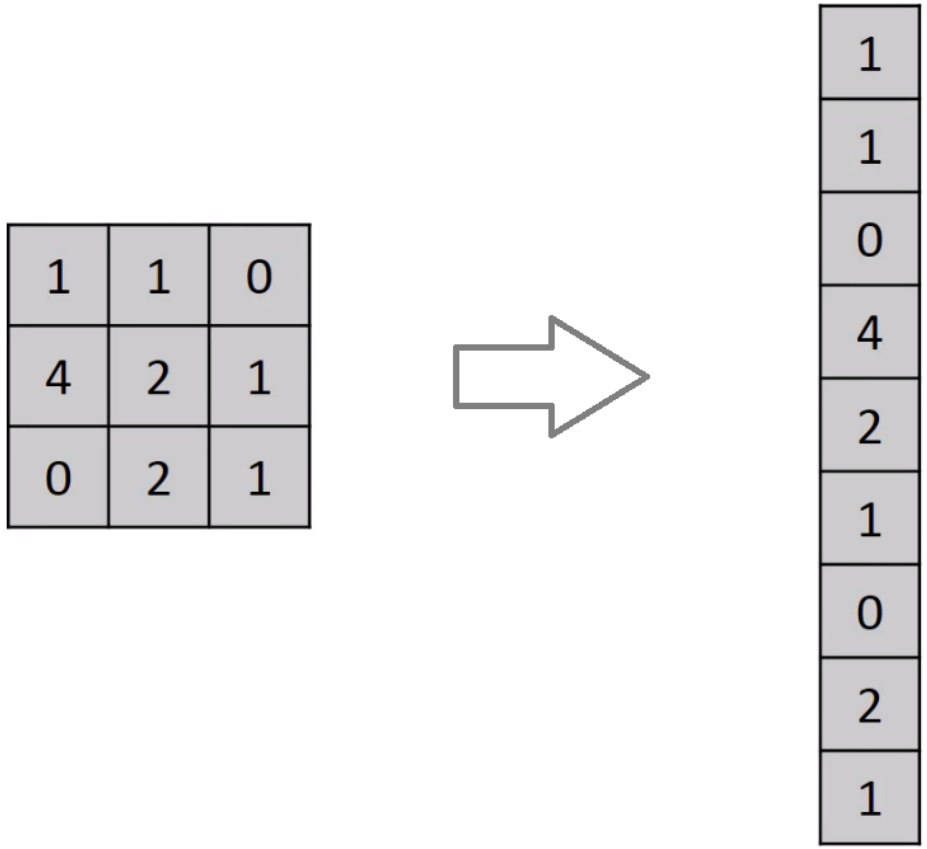
Hint 2

Hint

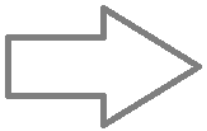


*Video Courtesy**: 3Blue1Brown** (Check him on YouTube!)

Neural Networks with Images



Neural Networks with Images – Dog example

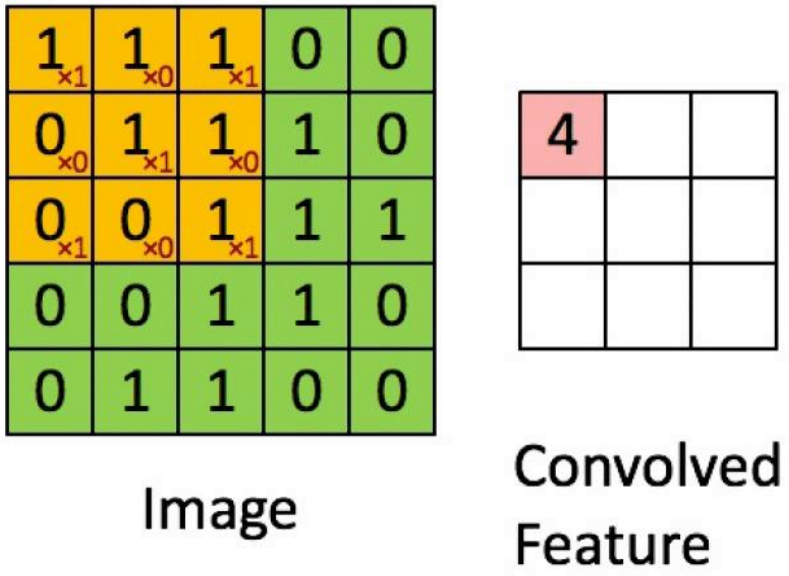


“slightly” exaggerated

Convolutional Neural Networks

Convolutional Neural Network (CNN) Layers

Convolution
Extract features & Keep spatial relationship



Pooling/Subsampling
Reduce dimensionality & retain information

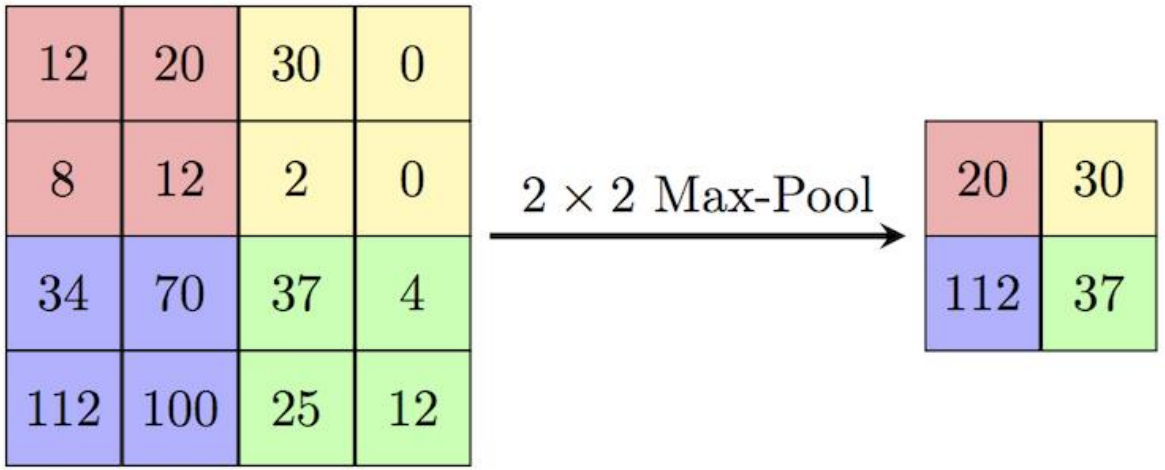
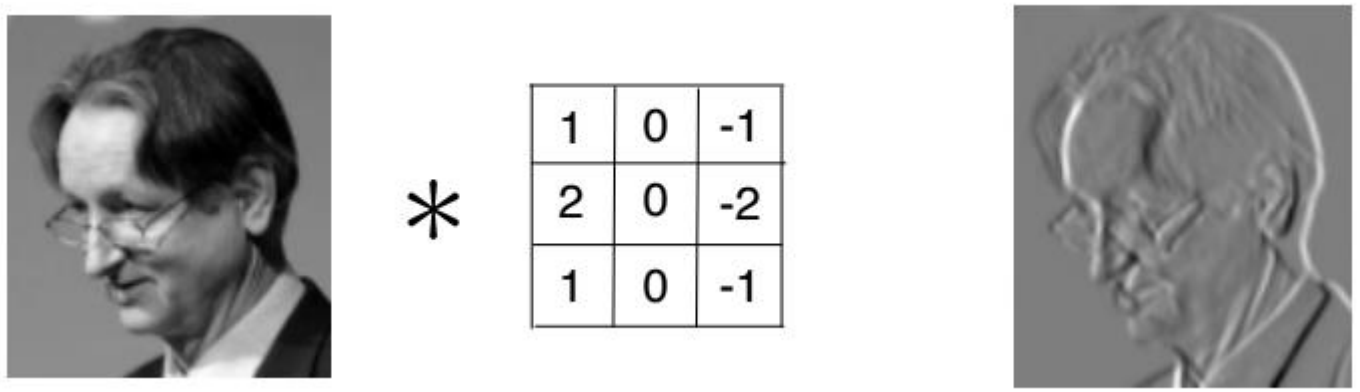


*Figure Courtesy: Erik Reppel

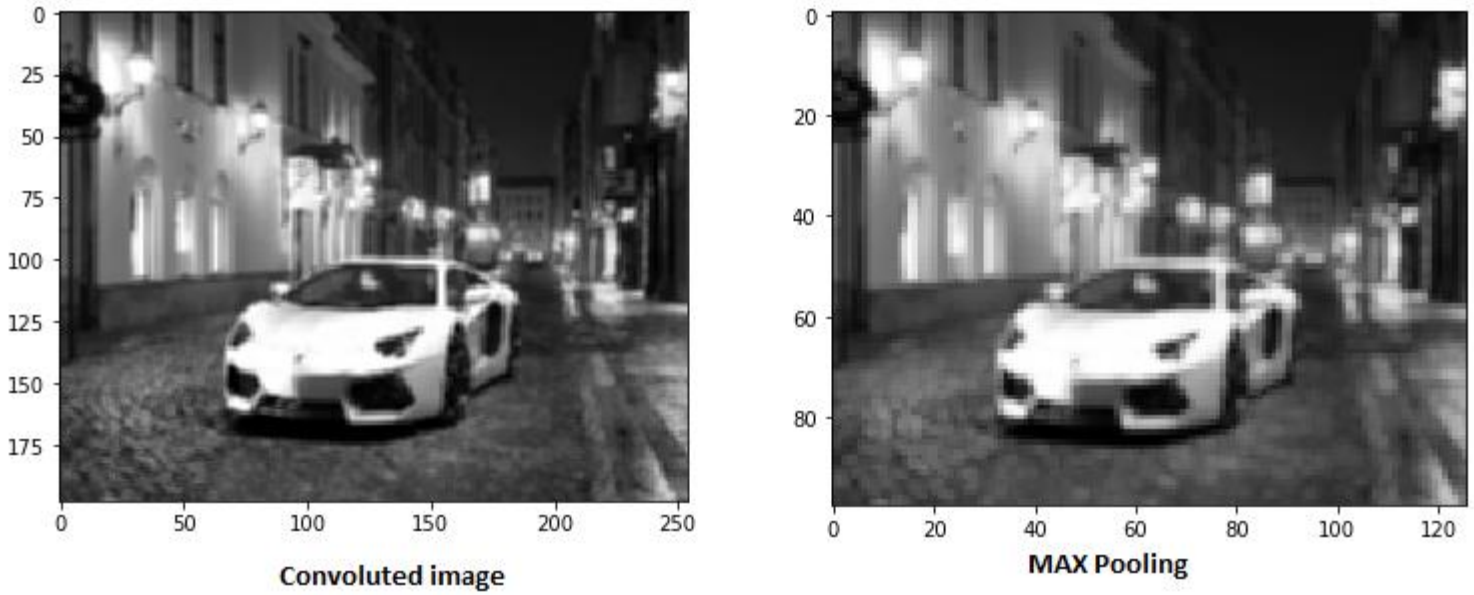
*Figure Courtesy: Cambridge Spark Ltd

Convolution and max pooling layer on image

Convolution



Max Pooling



Source: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/convolution | https://www.analyticsvidhya.com/blog/2017/06/architecture-of-convolutional-neural-networks-simplified-demystified/

Example of CNN

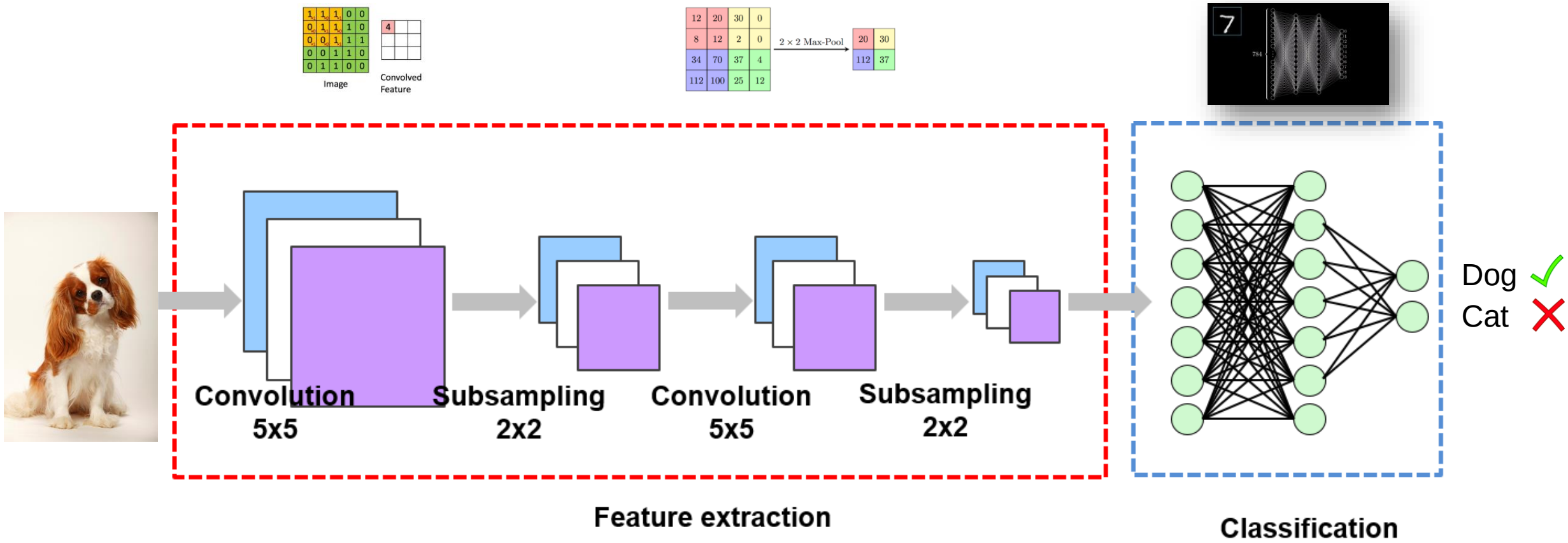
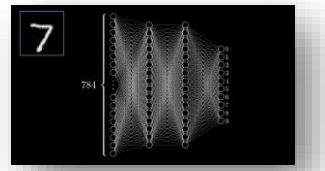


*Figure Courtesy: Suhyun Kim iSystems Design Labs

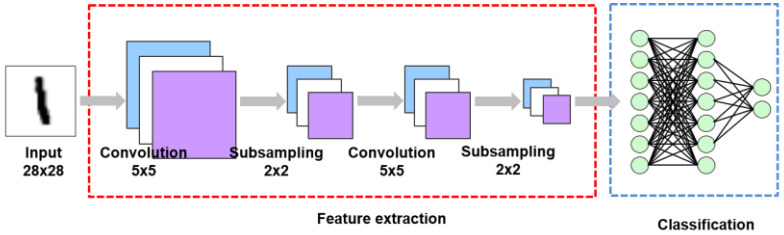
NN vs CNN

Input: MNIST database



Neural Network Result:

97.8%



Convolution Neural Network Result:

99.07%

Hands on examples of Supervised Learning

Hands on examples with MNIST

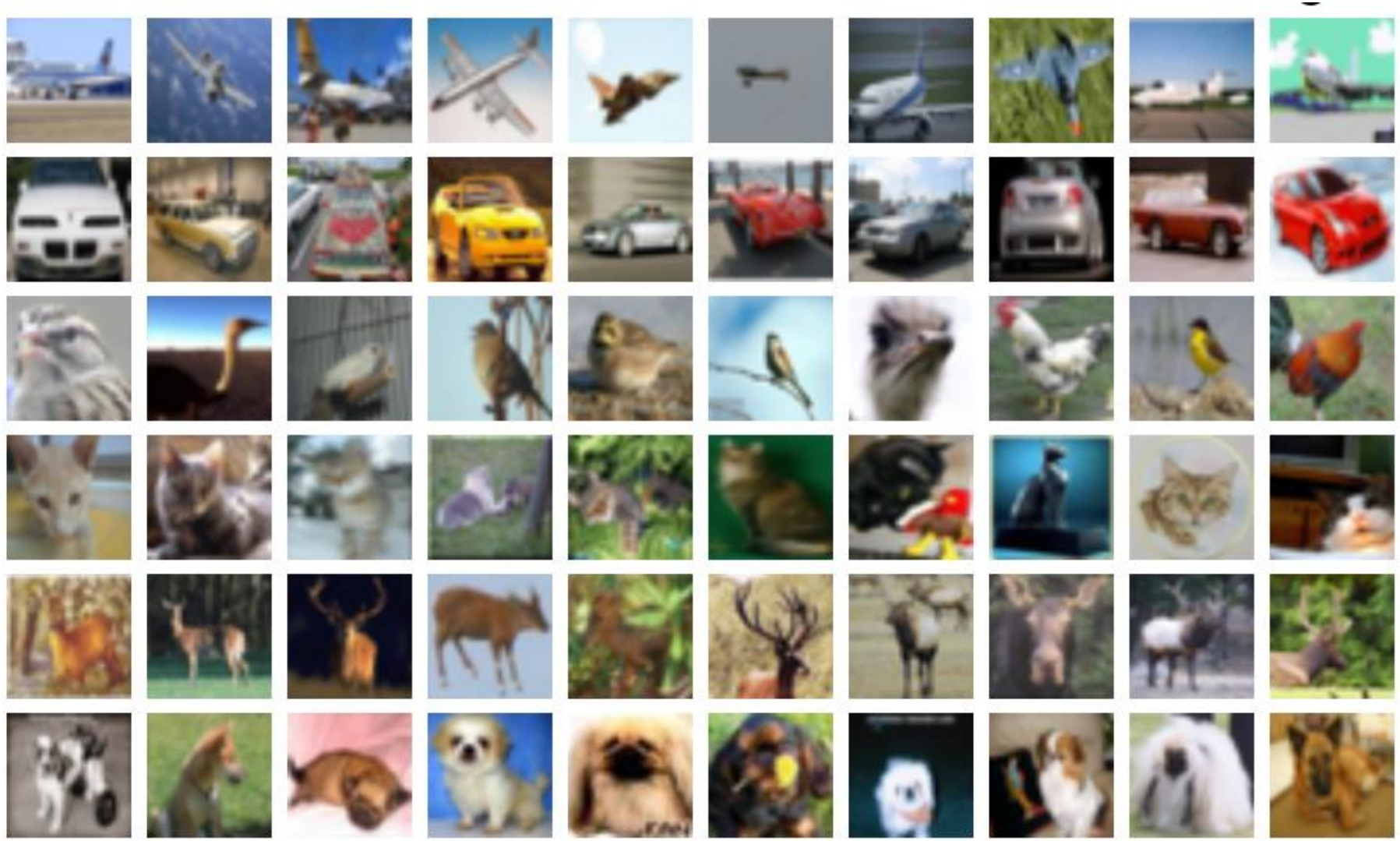
- Neural Network (MNIST)
- Convolution Neural Network (MNIST)

Follow this link & install yourself : https://github.com/SavvasRaptis/machine-learning-examples

Or directly use: https://mybinder.org/v2/gh/SavvasRaptis/machine-learning-examples/HEAD

HANDS-ON LEARNING

If MNIST is easy for you, check CIFAR-10



Can get harder!

https://www.kaggle.com/datasets/returnofsputnik/chihuahua-or-muffin

Supervised Learning in Space Physics

CME prediction example

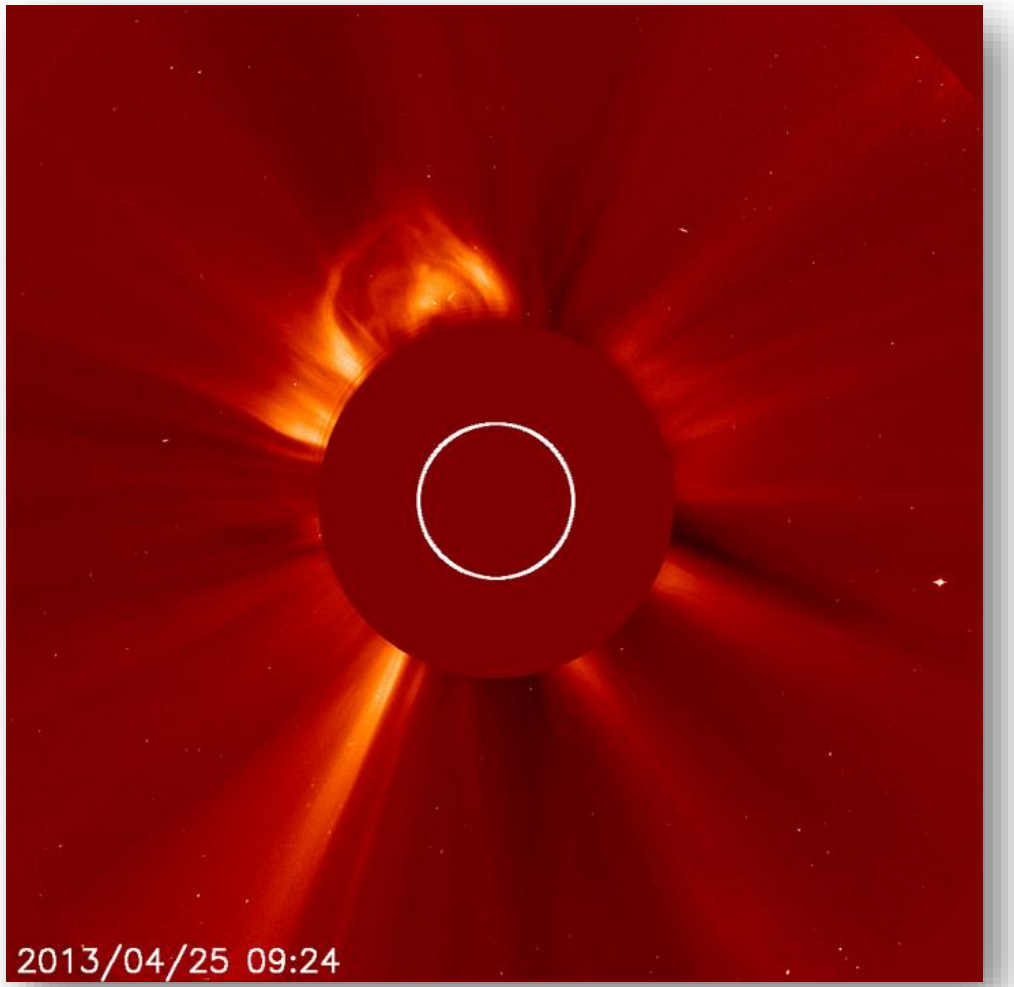
(Classification problem)

Coronal Mass Ejections (CMEs)

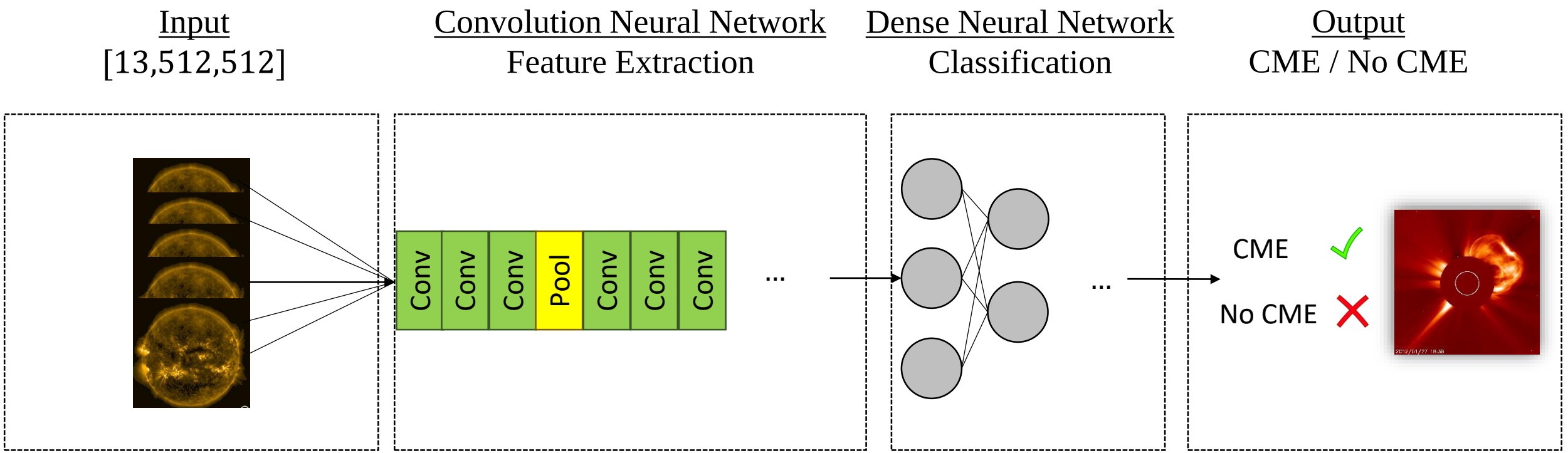
CMEs
Energetic particles from the Sun

⇩

Problems in
Satellites/Communications/Grids/infrastructures etc.



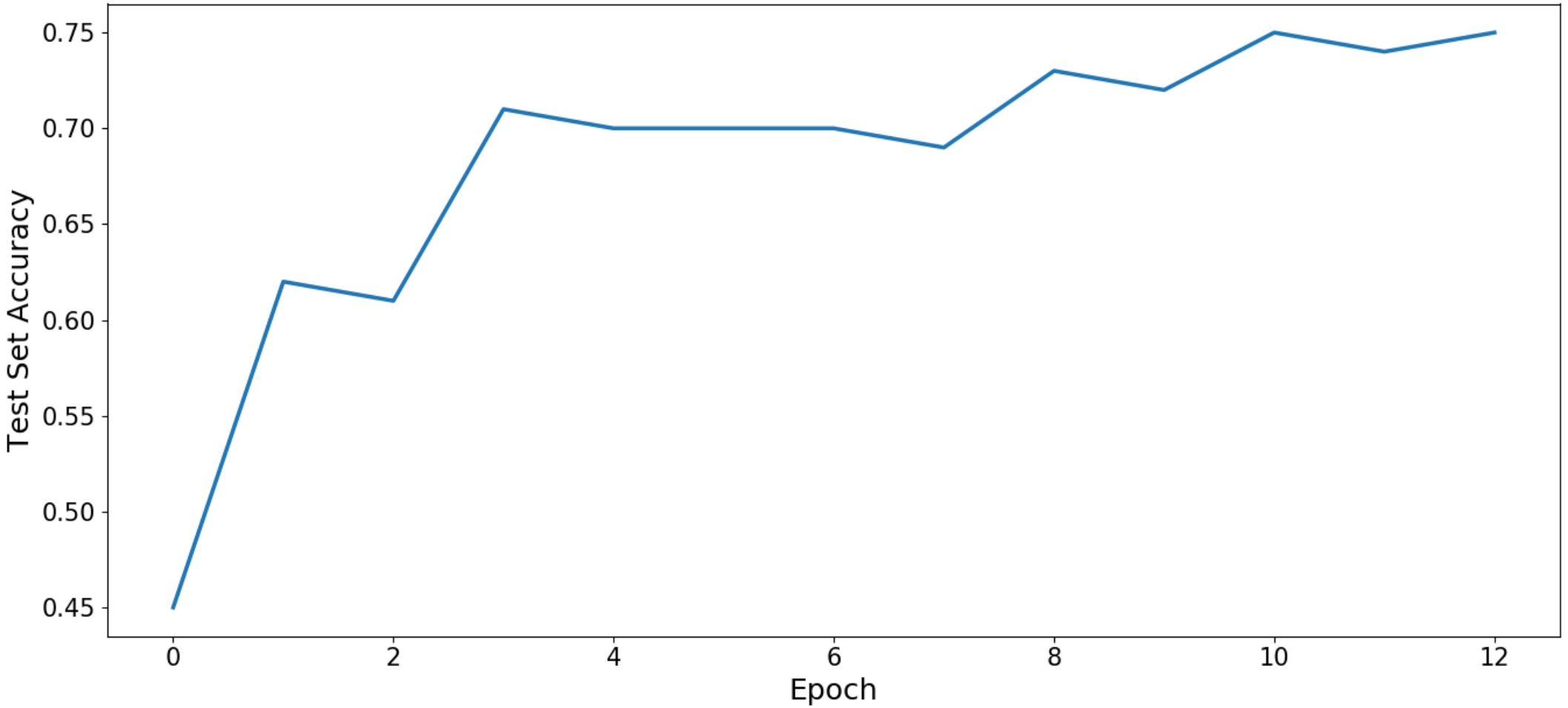
Project description



Input = 13 SDO images, 2 [h] history before the event.

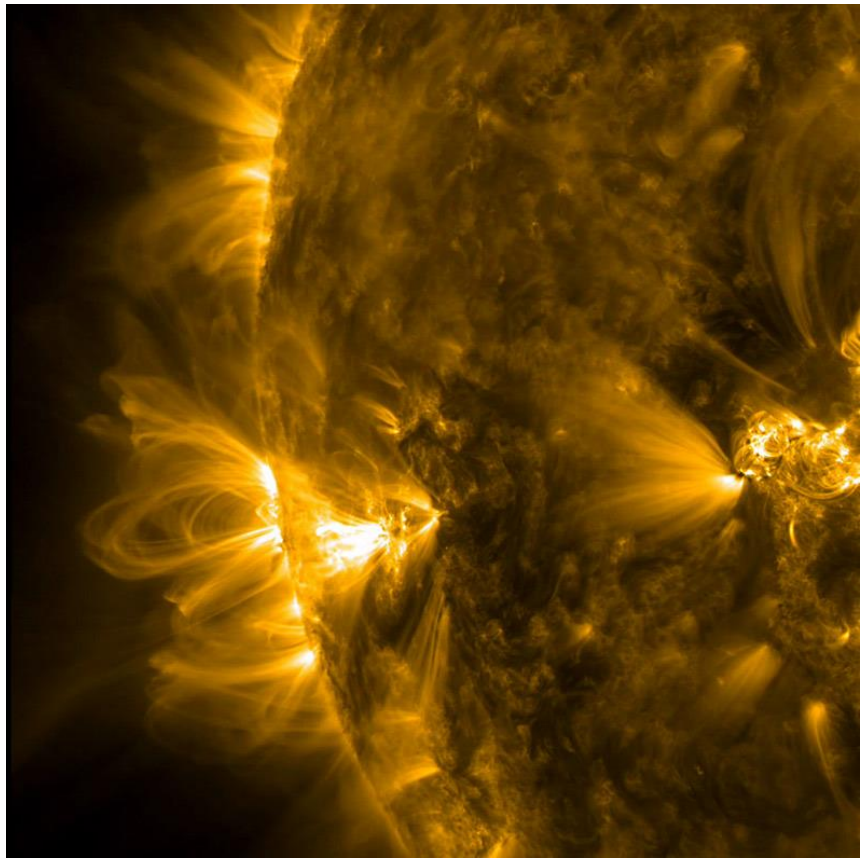
Output = 1/0

Result of CNN



Coronal Loop reconstruction example (Regression problem)

Project description



Basic Properties

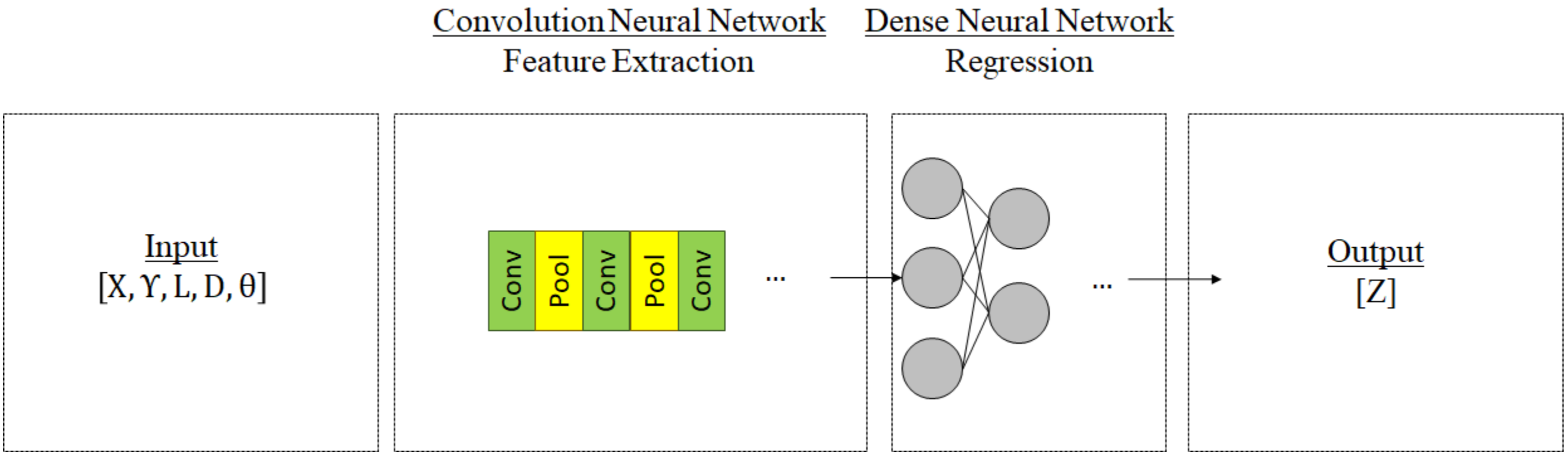
- More easily seen in EUV & X-ray
- More common in solar max
- May be relevant to the coronal heating problem

Information

- We see projections, therefore 2D can be observationally derived but 3D needs to be computed.
- 3D = vital for magnetic field extrapolation
- There are many methods, but have limitations. [Aschwanden (2011), Chifu et al. (2017)]

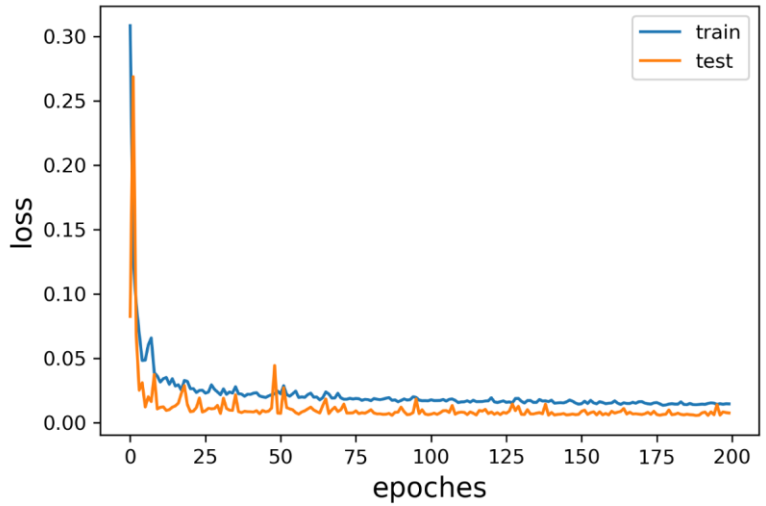
See original publication: [Chifu & Gafeira 2021 ApJL]

Neural Network Architecture

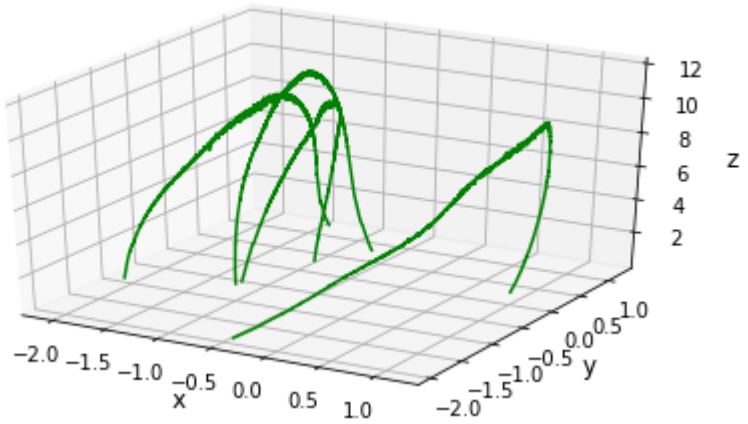


X,Y, Z : Coordinates (1500 points each)
L : Length of the coronal Loop
D : Distance between footpoints
Θ : Angle defined by the two footpoints with the top of the loops

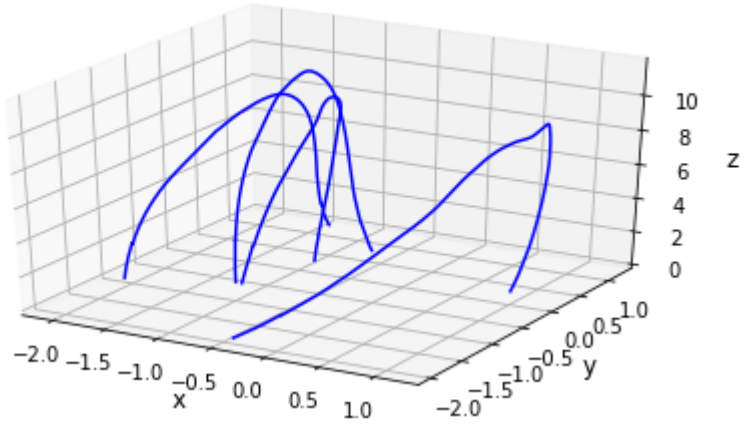
Results of the reconstruction



Predicted Coronal Loops



Actual Coronal Loops

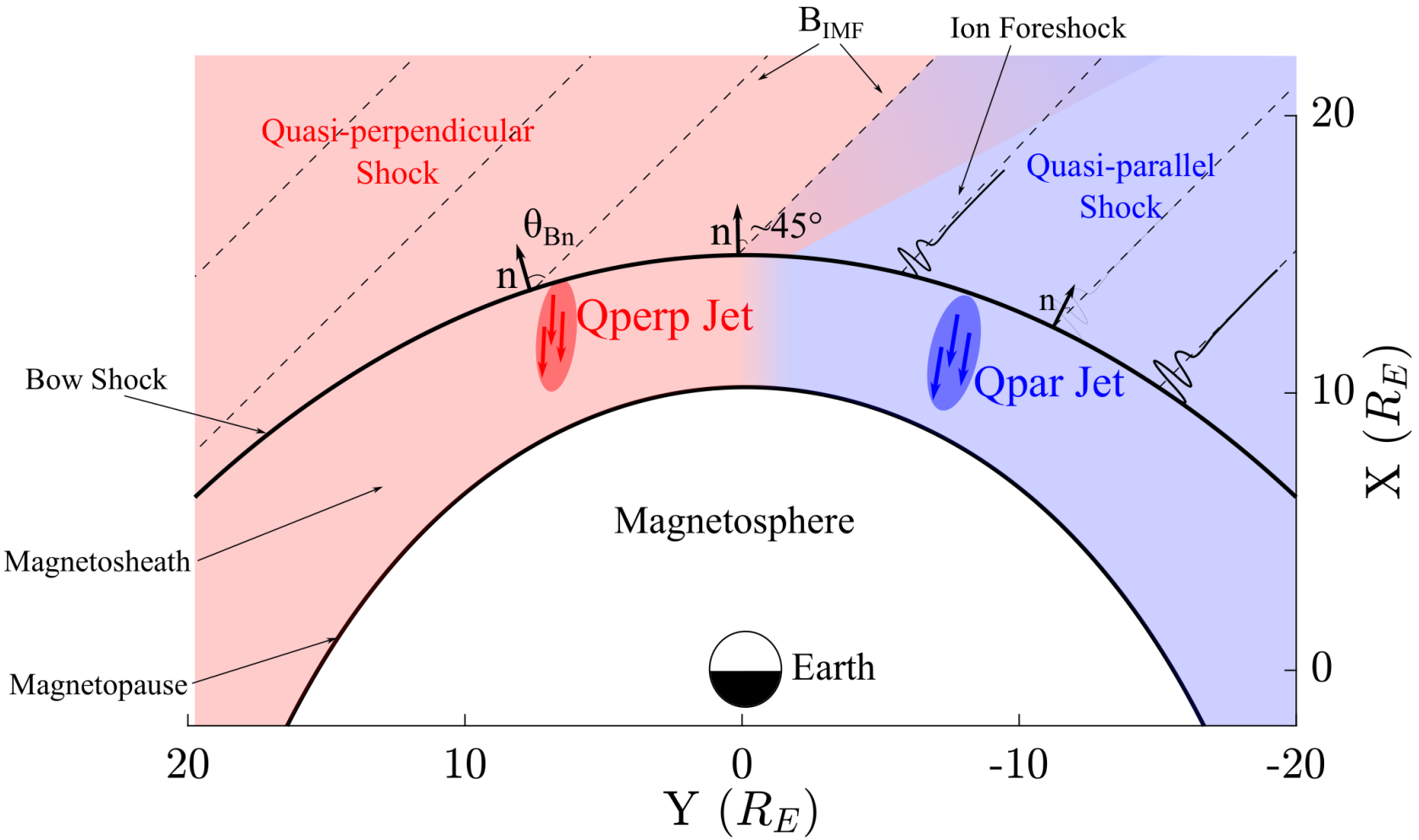


See original results (Chifu & Gafeira 2021)

Magnetosheath jet classification

(Classification problem)

Magnetosheath jet



Definition

Magnetosheath jets are **transient localized enhancements** of **dynamic pressure** (density and/or velocity increase)

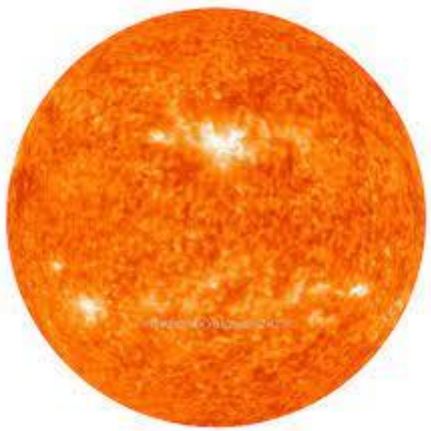
Classification

"Θ_{Bn} is the angle between the IMF and the shock's normal vector"

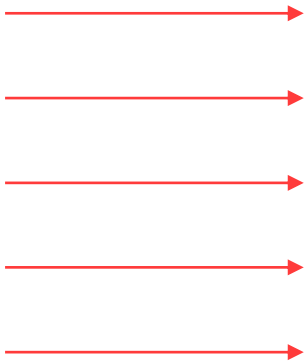
Qpar = $\Theta_{Bn} \lesssim 45°$
Qperp = $\Theta_{Bn} \gtrsim 45°$

"~10 times more often in Qpar MSH"

Classification task



The solar wind

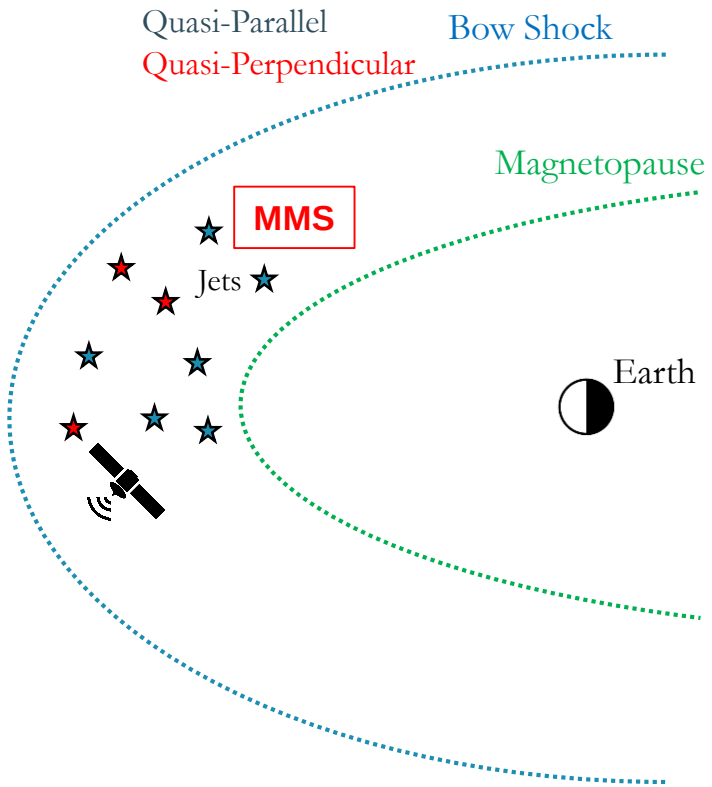


WIND

L_1

ACE

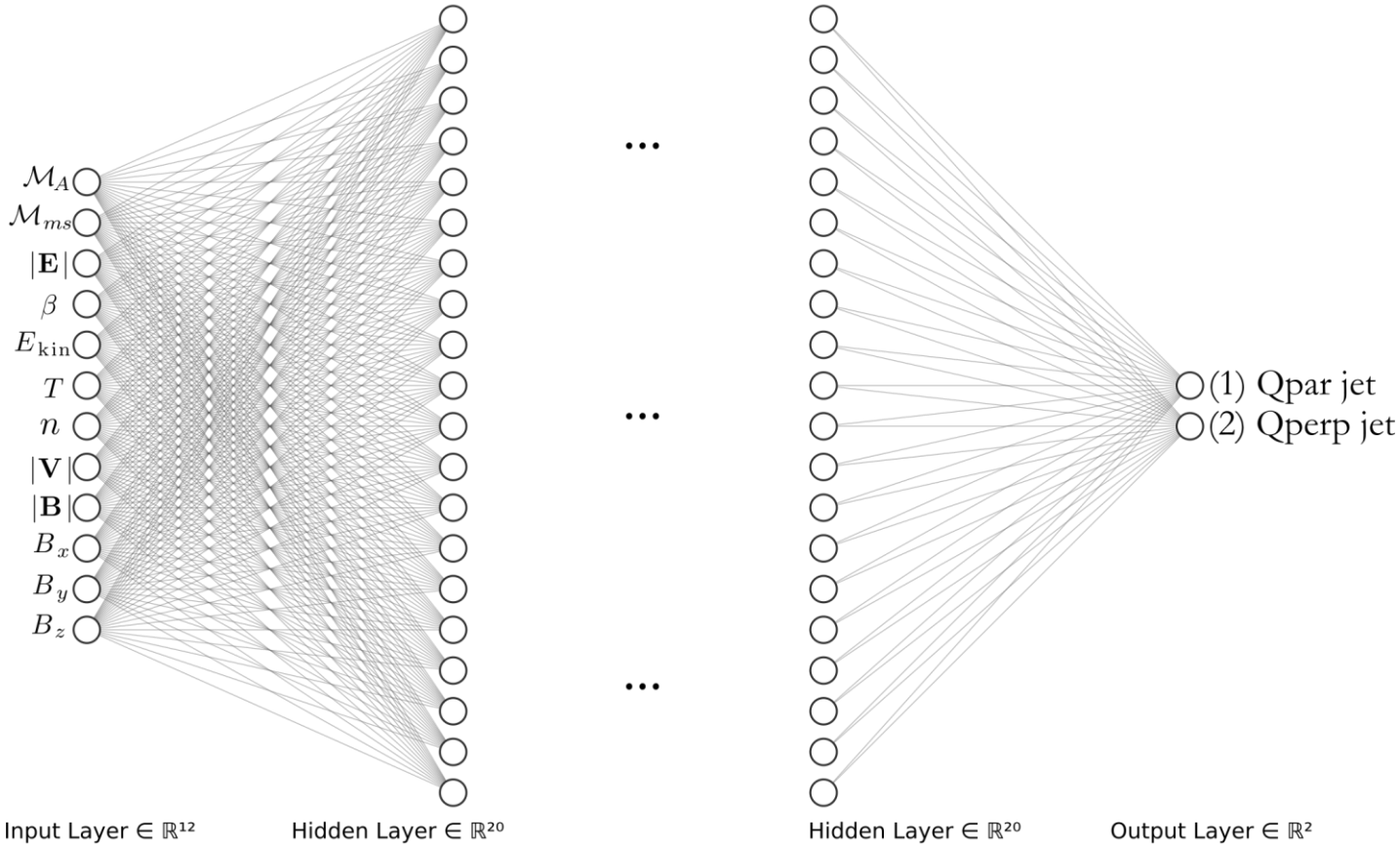
Classify with ML



Classifying jets with Neural Networks

(a) **Method**

Upstream
SW
L1 OMNI



Downstream
MSH
MMS

(b) **Results**

Class	Neural network (B) (%)	Neural network (No - B) (%)	θ_{cone} (%)	Coplanarity (%)	Modeling (%)
Qpar	98	95	61	81	74
Qperp	88	87	94	79	86

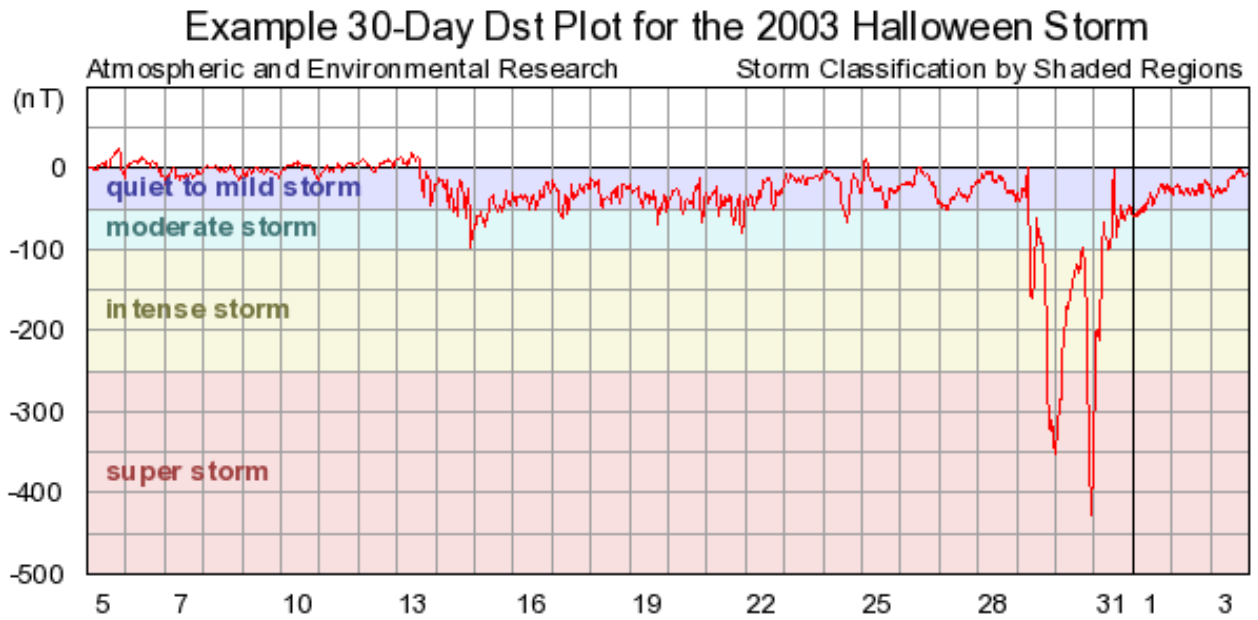
Raptis, Aminalragia-Giamini et al. (2020) | Front. Astron. Space Sci

Forecasting of DST index
(Regression problem)

Prediction of the Dst index

“average disturbance field at the Earth's equator from four low-latitude magnetic observatories”

The value of Dst along with its pattern (sharpness, duration, etc.) provide a profile for the **geomagnetic storm**



Recap questions

Why we call this a regression problem ??

What would a classification problem be here ?

https://github.com/SavvasRaptis/machine-learning-examples

Try for example:

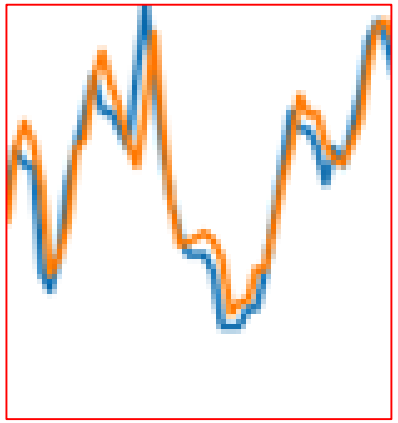
- t_fcast = 1
- t_fcast = 3

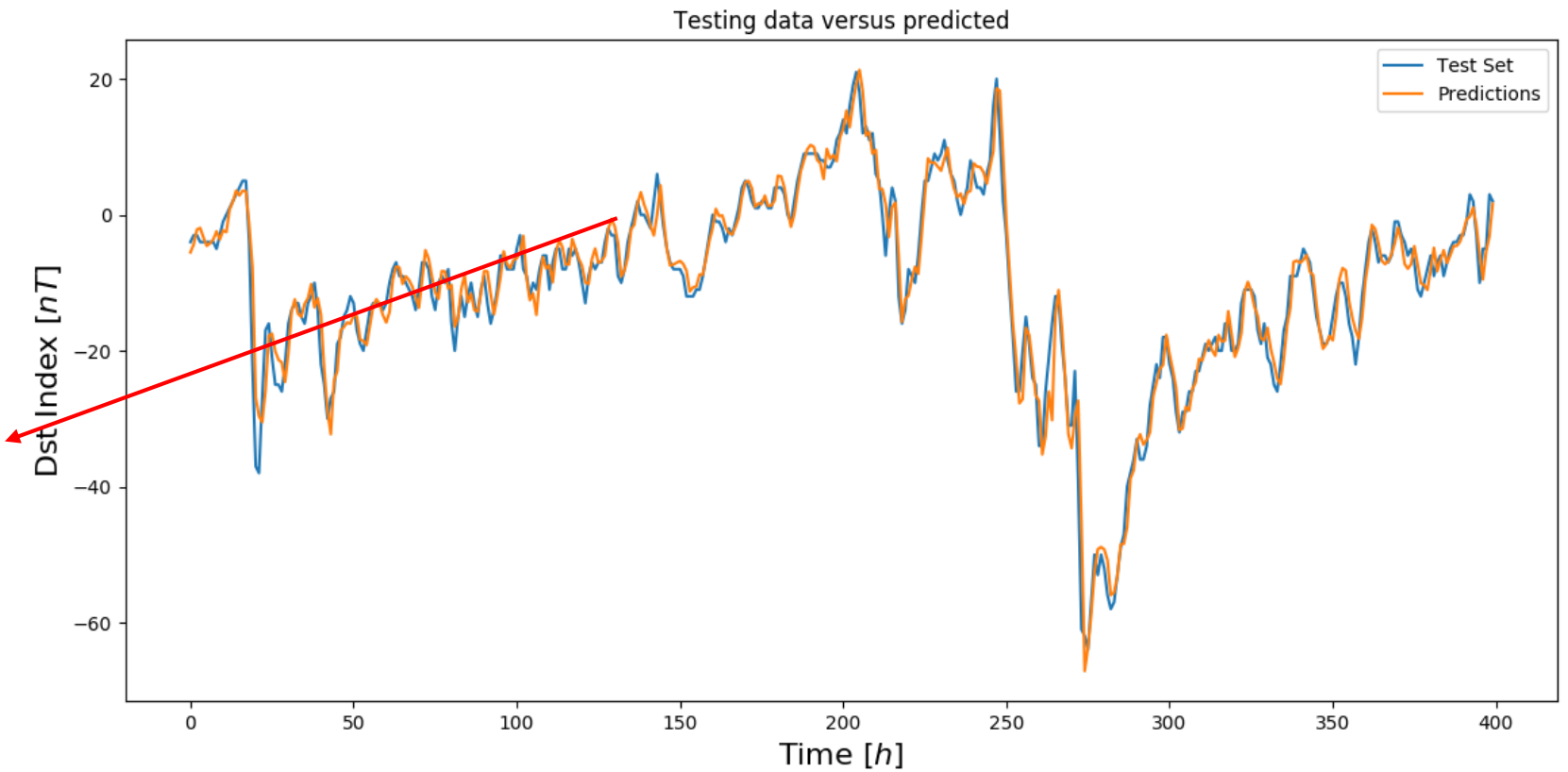
Prediction of the Dst index

Input: Quantities from satellite ACE (L1)

4 Layers of (20, 10, 5, 1) Neurons
500 Epochs

Output: DST index (1 Hour later)

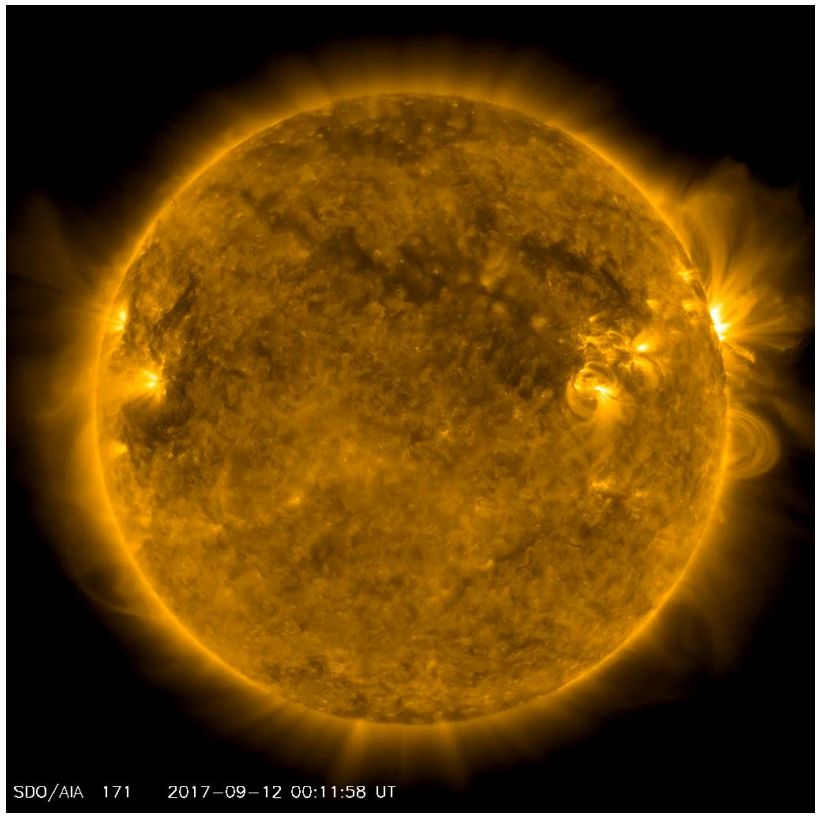


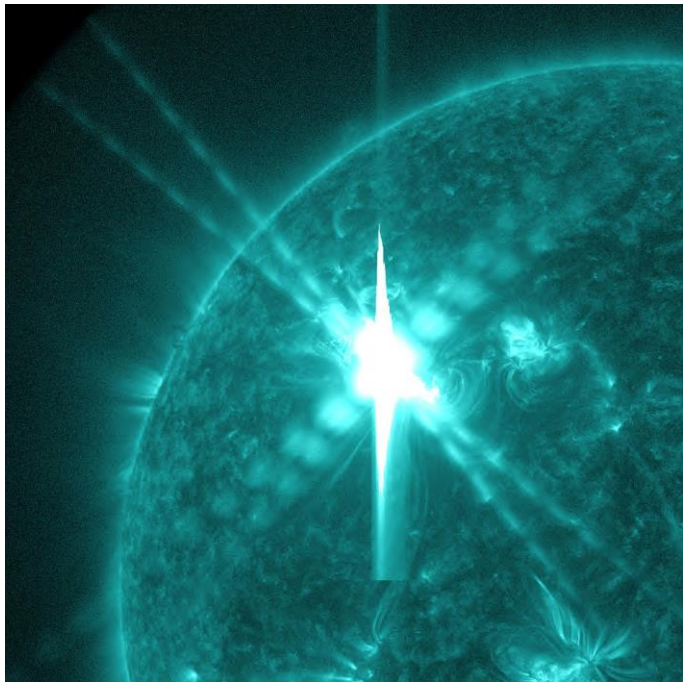


Anything fishy ?? Is this a good prediction ??

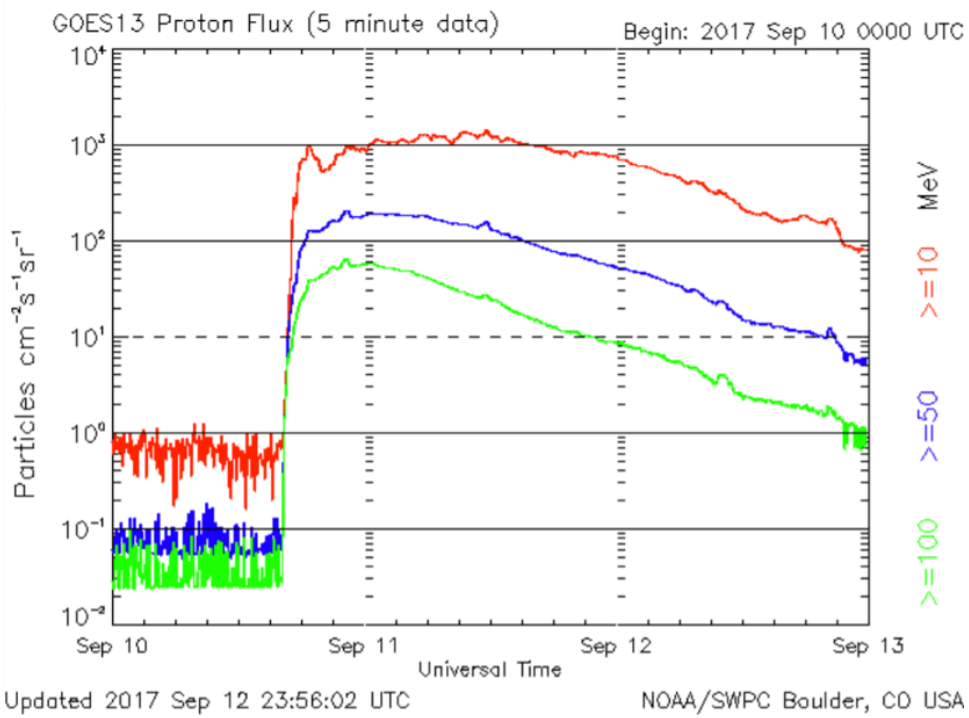
Solar Energetic Particle Prediction (Classification problem)

Solar Energetic Particles (SEPs) & Flares





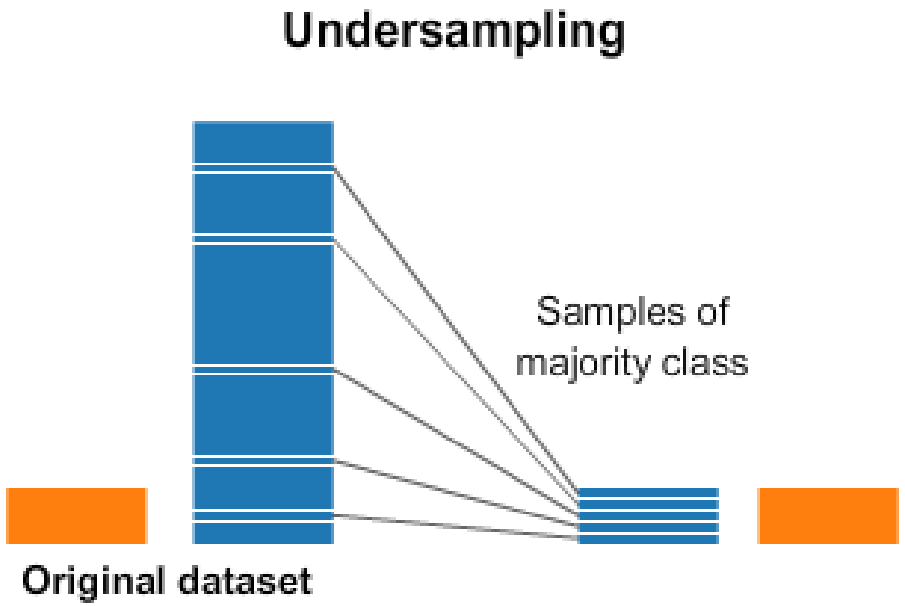
Flares = X-ray emission

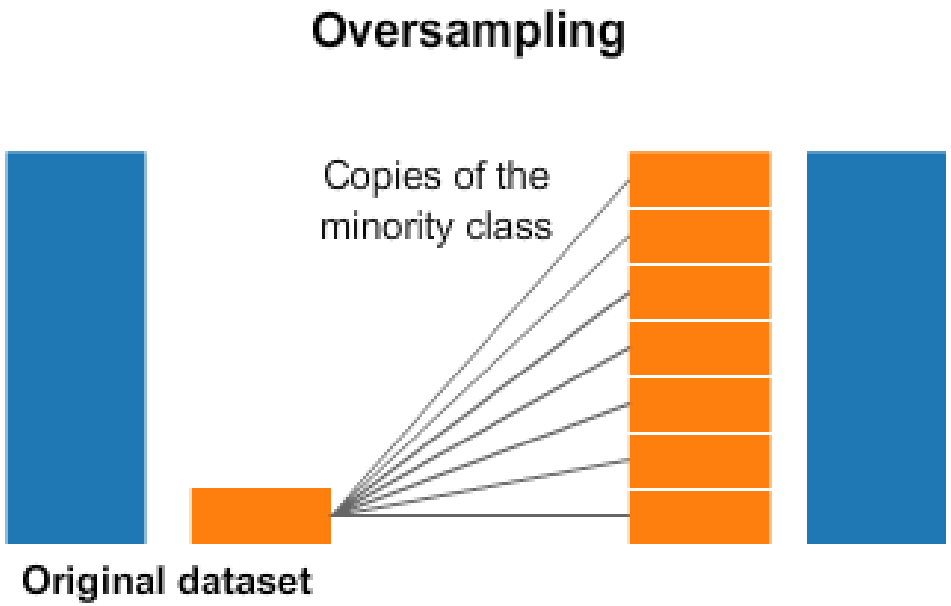


High energy proton flux
SEP

Classes in real datasets

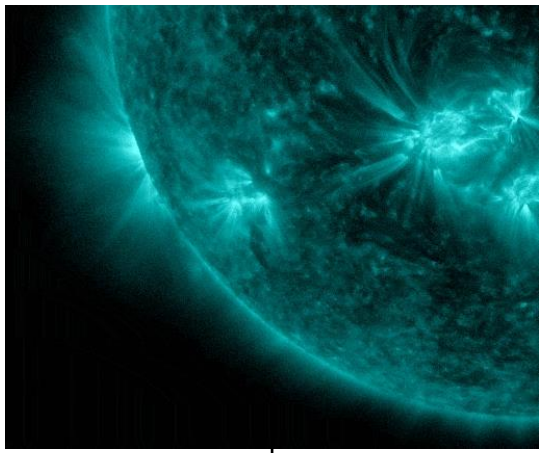
Question: In the examples we show before… how “balanced” was the dataset?





Application on forescasting SEPs

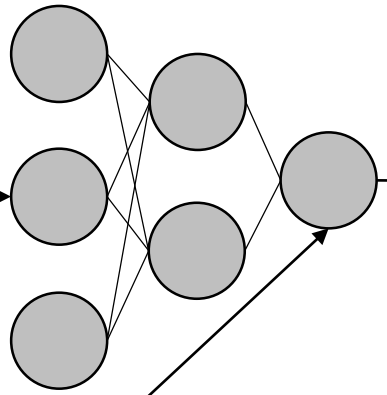
Problem description



24 features

Occurrence of SEP based on X-rays of flares

Modeling



Validation & Results

	Predicted Yes	Predicted No
Actual Yes	TN	FP
Actual No	FN	TP

Comparisons to other models recently published
Review of Solar Energetic Particle Models (Advances in Space Research)

Aminalragia-Giamini, Raptis et al. (2021) | J. Space Weather Space Clim

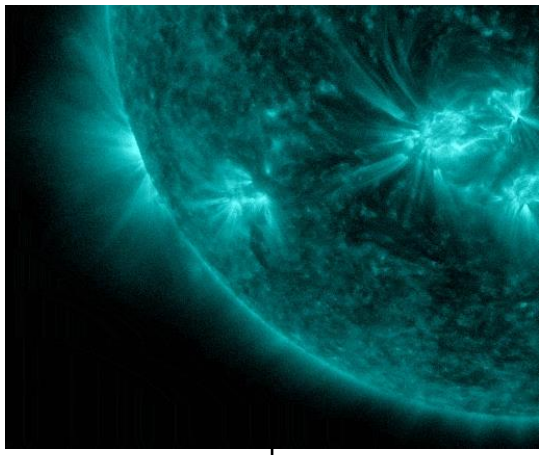
https://github.com/SavvasRaptis/machine-learning-examples

Try for example:

- 50 epochs
- 300 epochs

Application on forescasting SEPs

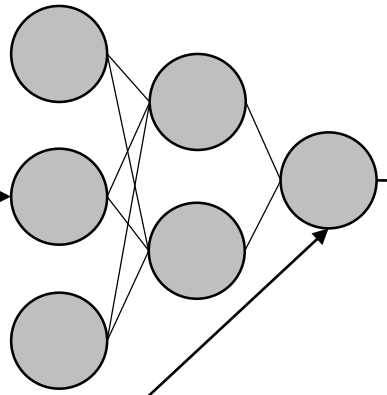
Problem description



24 features

Occurrence of SEP based on X-rays of flares

Modeling



Validation & Results

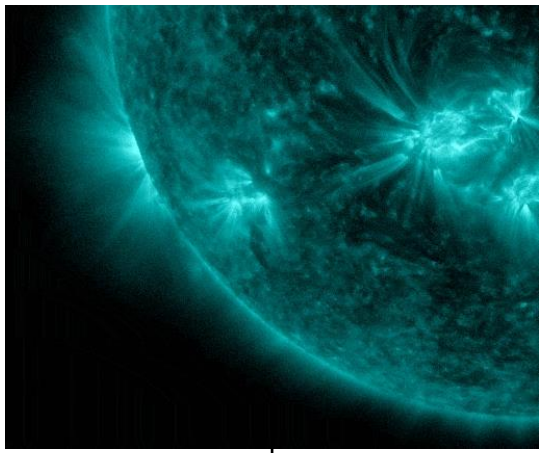
	Predicted Yes	Predicted No
Actual Yes	TN	FP
Actual No	FN	TP

Comparisons to other models recently published
Review of Solar Energetic Particle Models (Advances in Space Research)

Aminalragia-Giamini, Raptis et al. (2021) | J. Space Weather Space Clim

Application on forescasting SEPs

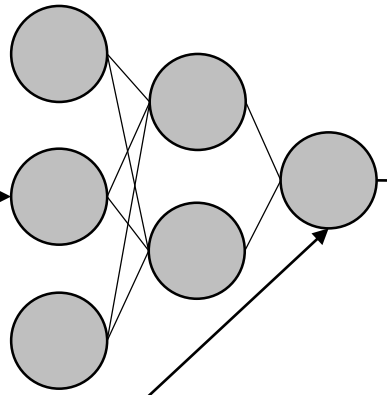
Problem description



24 features

Occurrence of SEP based on X-rays of flares

Modeling



Validation & Results

	SEP predicted always YES	SEP predicted always NO
SEP occurred YES	191/220 [86.81 %]	19/220 [8.63%]
SEP occurred NO	[7.77 %]	[92.23%]

Comparisons to other models recently published
Review of Solar Energetic Particle Models (Advances in Space Research)

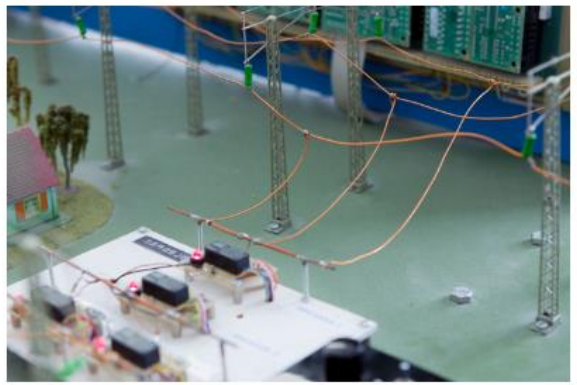
Aminalragia-Giamini, Raptis et al. (2021) | J. Space Weather Space Clim

Learn more on machine learning & space physics

EF2245 Space Physics II 7.5 credits

Before course selection

Administer About course



The course is a continuation of Space Physics I, and aims to give the student a more quantitative knowledge of space plasma phenomena and research methods.

DD2424 Deep Learning in Data Science 7.5 credits

Before course selection

Administer About course

Final notes

- We only saw a small piece of the puzzle today. Applications in physics include:

- *Unsupervised* learning (clustering of magnetospheric environments)

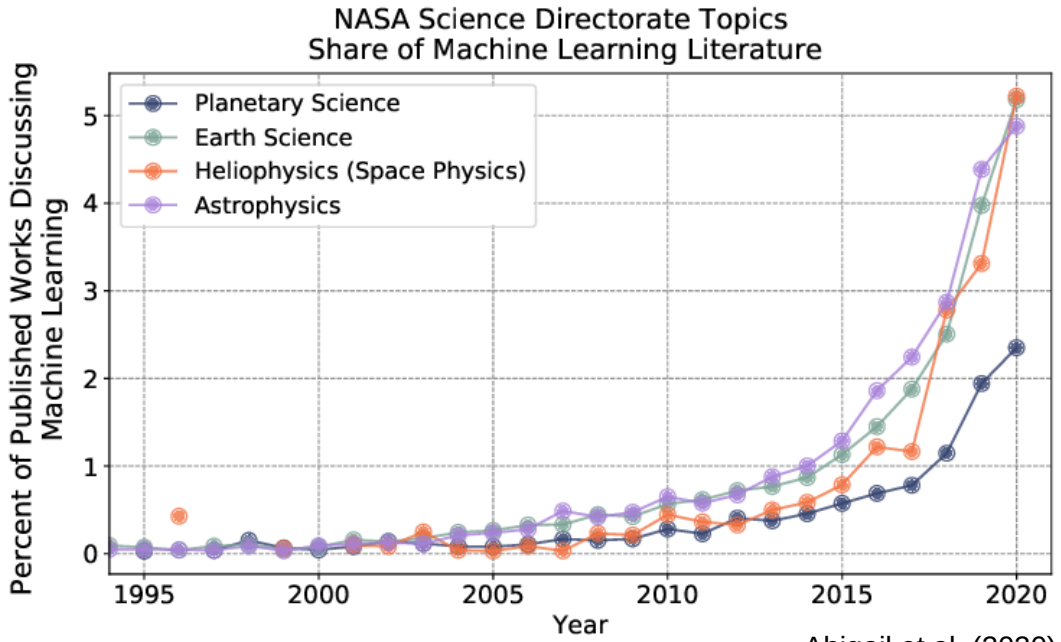
- *Reinforcement* learning (optimize when and where to make observations with telescopes)

- Physics – Informed Neural Networks (PINNS). (e.g., diffusion in radiation belts)

I will be more than glad to share information about these topics if you want to learn more & provide material.

Summary & Take-Home Message

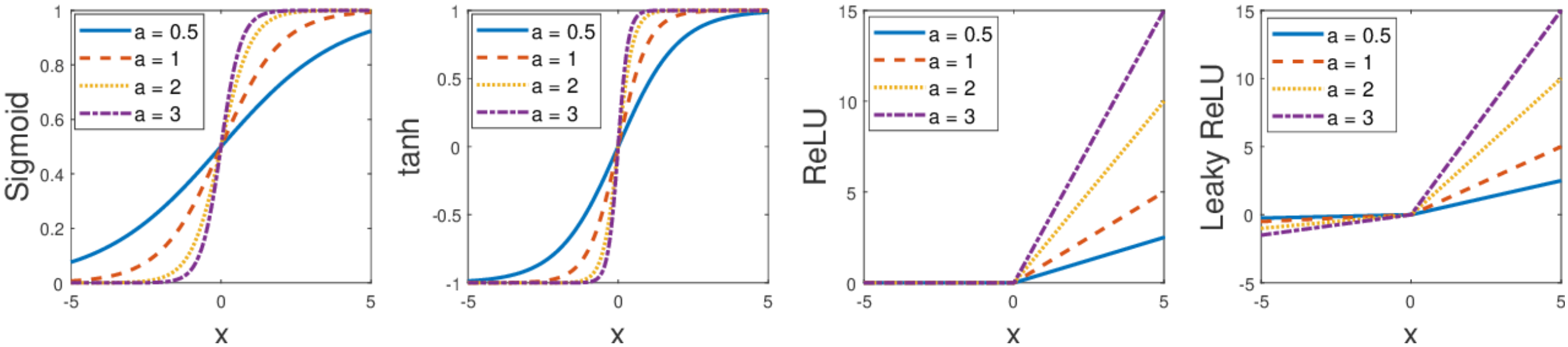
- Machine learning techniques are already used a lot in research, both in academia and in industry.
- Things move very quickly, PINNs that I mentioned in the previous slide are just here for the last ~5 years and there are hundreds of articles on them already.



Abigail et al. (2020)

EXTRAS

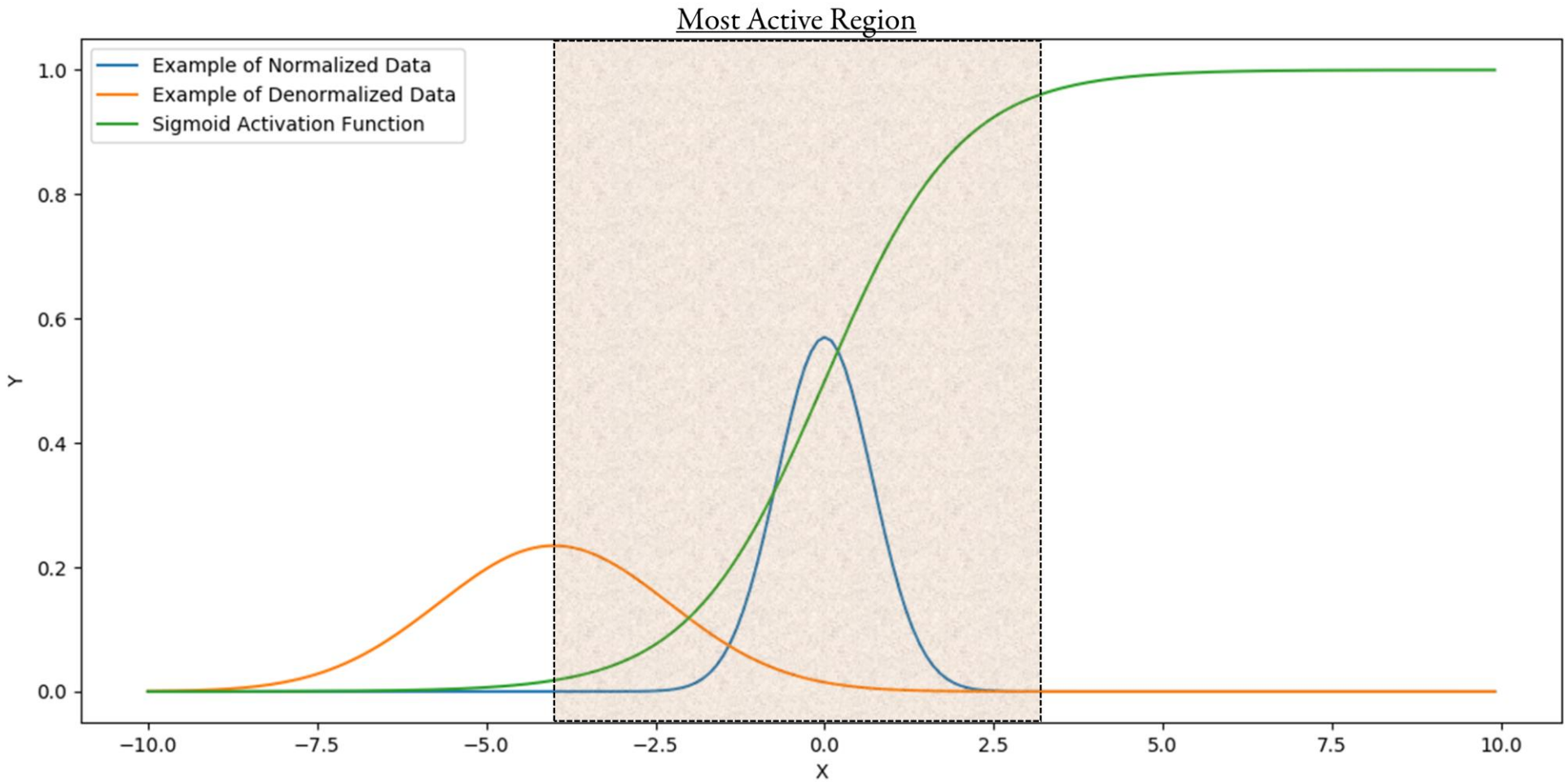
Adaptive activation functions



TLDR: optimize the activation function ∀ neurons during training = no hardcoded a

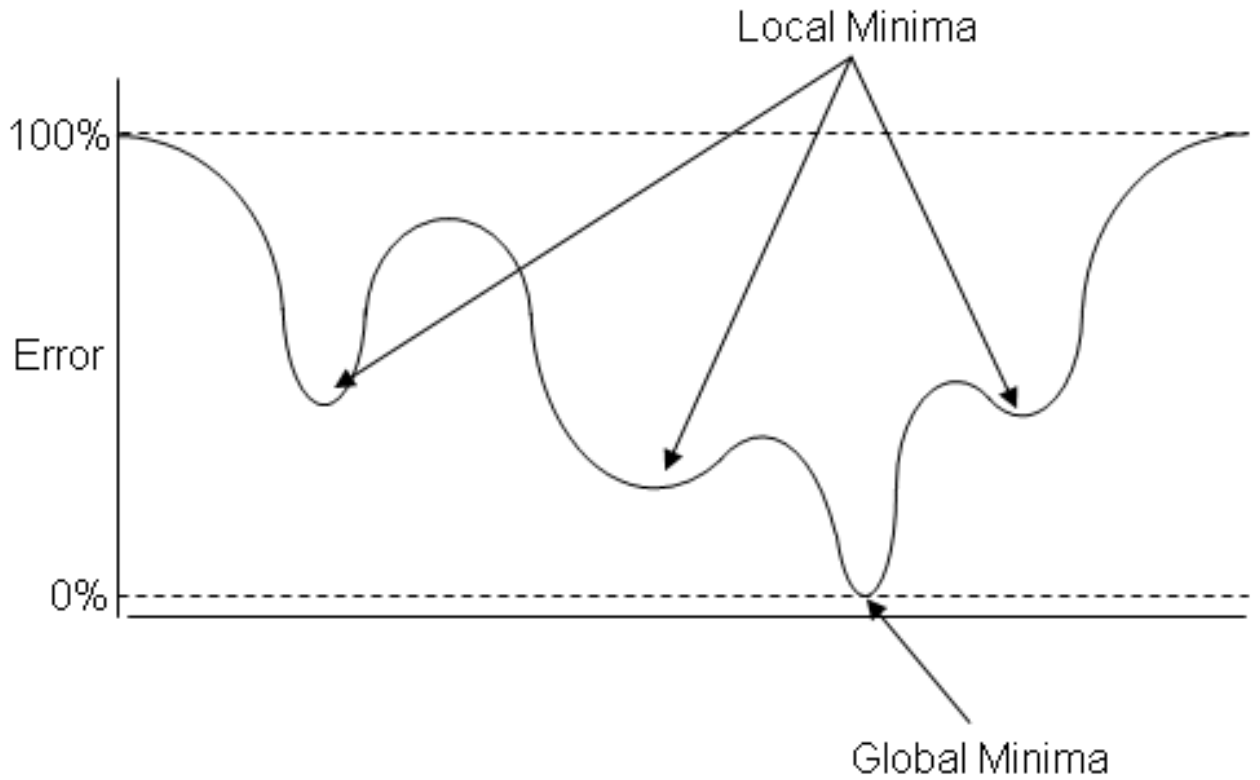
Jagtap, A. et. al (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks.

Why normalization is vital?



Raptis (2018) | KU Leuven

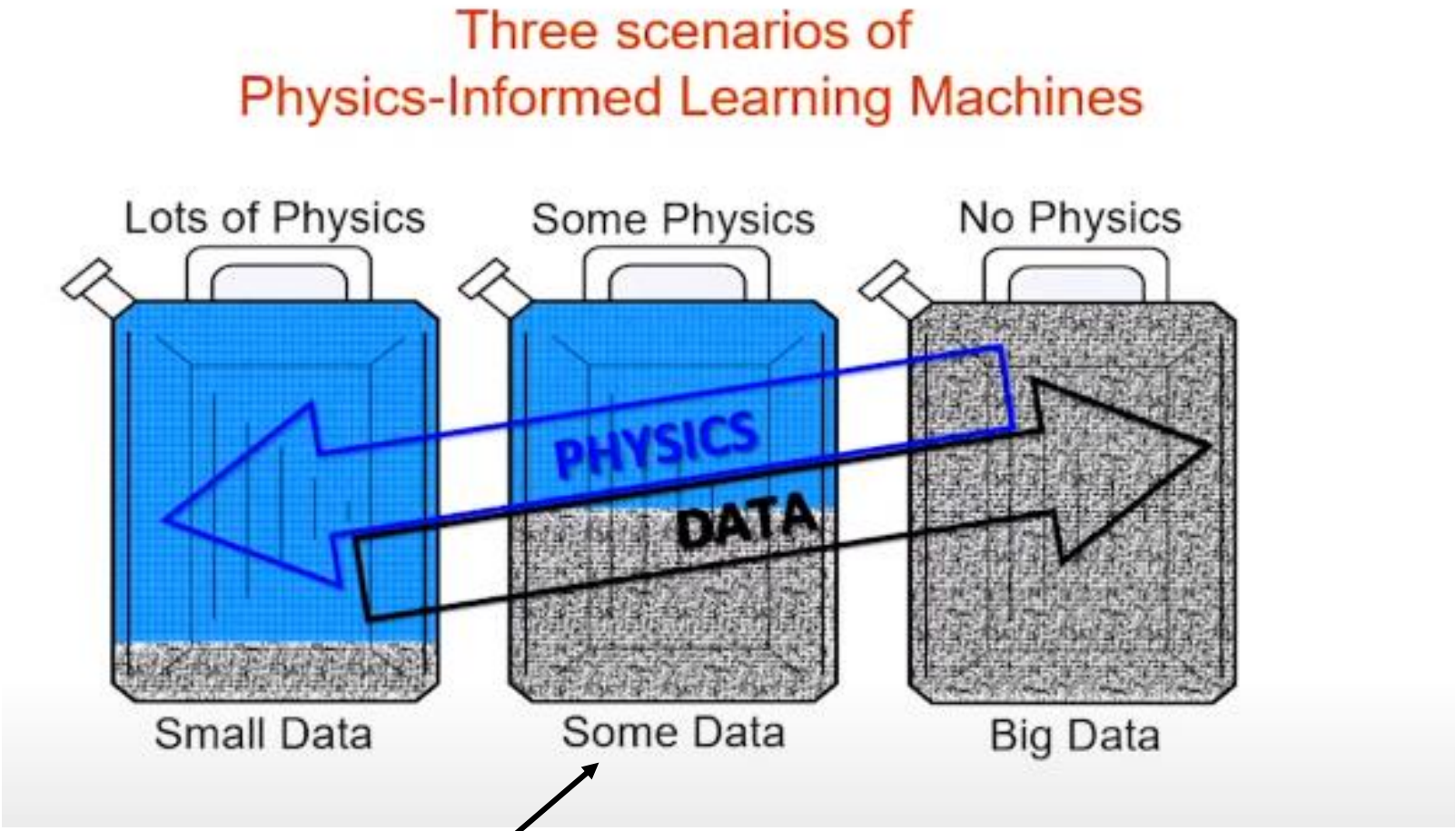
“Proven” to solve a vital problem in optimization



TLDR: Huge problem in traditional machine learning a few years ago = can be solved now

Kenji Kawaguchi et. al (2019)

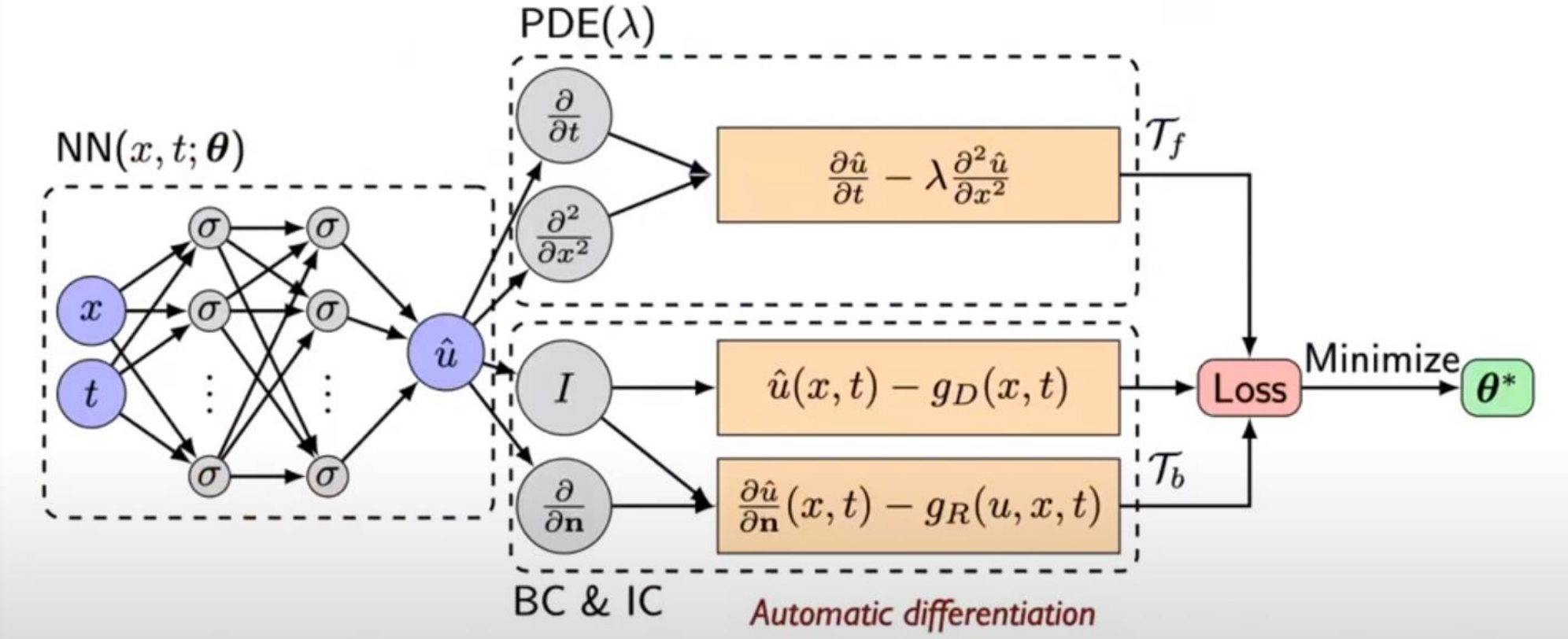
Physics Informed Machine Learning



Usual case in certain fields (fluid mechanics, space, plasma, medicine etc.)

What exactly is a PINN then ?

$$f\left(\mathbf{x}; \frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}; \frac{\partial^2 u}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 u}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right) = 0, \quad \mathbf{x} \in \Omega \qquad \mathcal{B}(u, \mathbf{x}) = 0 \quad \text{on} \quad \partial\Omega,$$



$$loss = data\ fit + PDE\ residual + ICs\ fit + BCs\ fit$$

Idea = If output is a differentiable quantity with respect to an input = can compute PDEs = combined loss functions

2 type of problems

Data-driven solutions to partial differential equations

1) Burger’s equation
2) Schrodinger Equation

Data-driven discovery of nonlinear partial differential equations

3) Navier-Stokes Equation

Data-driven solutions
(forward problem)

Example 1 – Burger's equation

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

Let us define $f(t, x)$ to be given by

$$f := u_t + uu_x - (0.01/\pi)u_{xx},$$

$$MSE = MSE_u + MSE_f,$$

where

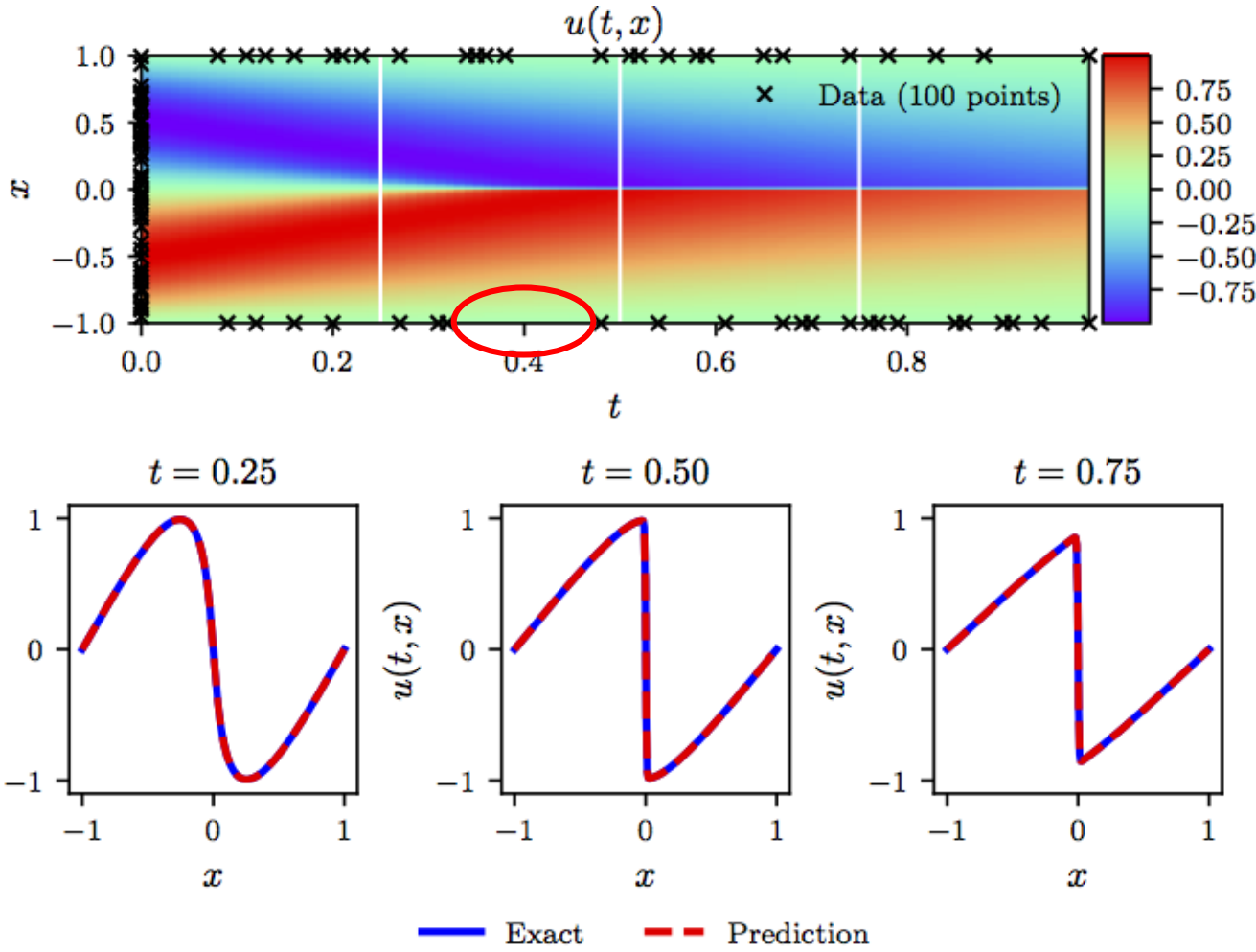
$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2,$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Here we basically enforce initial and boundary conditions.

Penalty of points not satisfying f / residual loss



Example 2 – Schrodinger equation

$$ih_t + 0.5h_{xx} + |h|^2 h = 0, \quad x \in [-5, 5], \quad t \in [0, \pi/2],$$

$$h(0, x) = 2\,\mathrm{sech}(x),$$

$$h(t, -5) = h(t, 5),$$

$$h_x(t, -5) = h_x(t, 5),$$

$$f := ih_t + 0.5h_{xx} + |h|^2 h,$$

$$MSE = MSE_0 + MSE_b + MSE_f,$$

where

$$MSE_0 = \frac{1}{N_0}\sum_{i=1}^{N_0} |h(0, x_0^i) - h_0^i|^2,$$

$$MSE_b = \frac{1}{N_b}\sum_{i=1}^{N_b} \left(|h^i(t_b^i, -5) - h^i(t_b^i, 5)|^2 + |h_x^i(t_b^i, -5) - h_x^i(t_b^i, 5)|^2 \right)$$

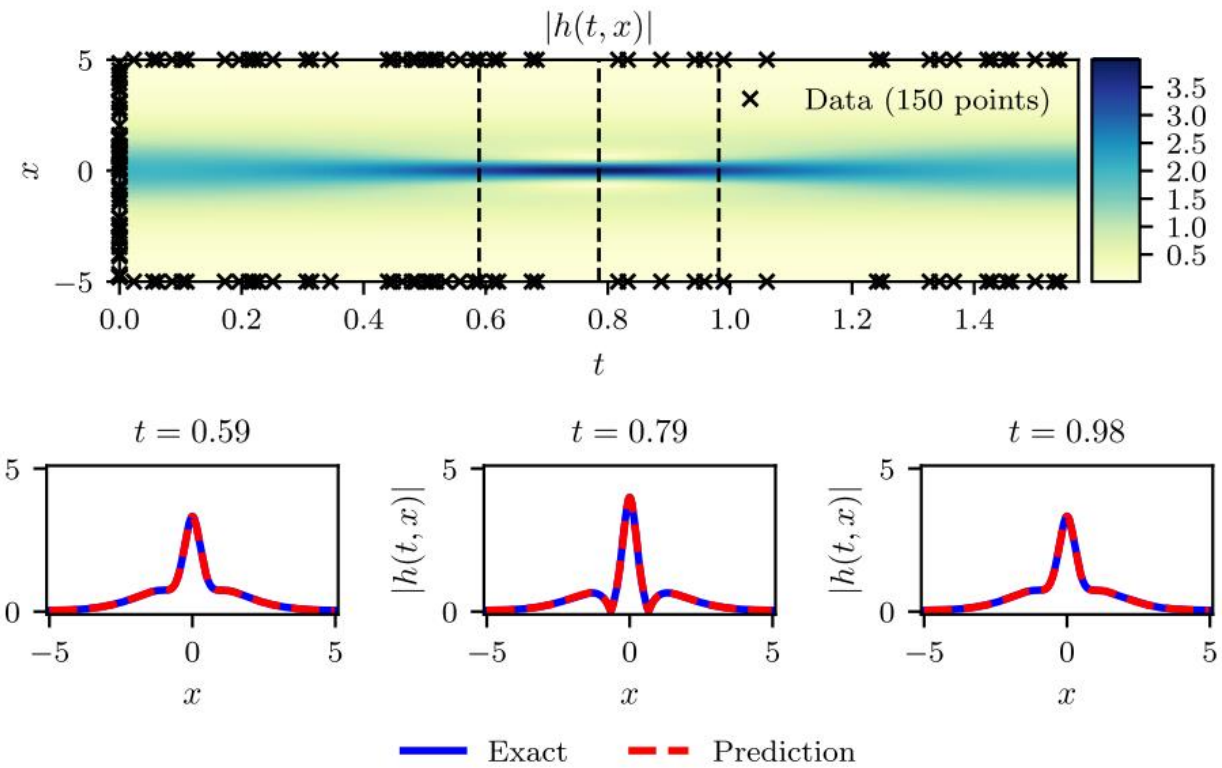
and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2.$$

Same thing as we saw before (initial)

Basically enforcing the above conditions (boundary)

Penalty of points not satisfying f (residuals)



Data-driven discoveries
(inverse problem)

Example 3 – Navier–Stokes equation (1)

$$u_t + \lambda_1(uu_x + vu_y) = -p_x + \lambda_2(u_{xx} + u_{yy}),$$
$$v_t + \lambda_1(uv_x + vv_y) = -p_y + \lambda_2(v_{xx} + v_{yy}),$$

Also, continuity equation/divergence-free result:

$$u_x + v_y = 0.$$

So basically we define the following system (f,g):

$$f := u_t + \lambda_1(uu_x + vu_y) + p_x - \lambda_2(u_{xx} + u_{yy}),$$
$$g := v_t + \lambda_1(uv_x + vv_y) + p_y - \lambda_2(v_{xx} + v_{yy}),$$

Basically imposing boundary conditions

The system of differential equations

Goal: Estimate $\lambda_{1,2}$ and $p(t, x, y)$

First step: Generate some data for steady state solution

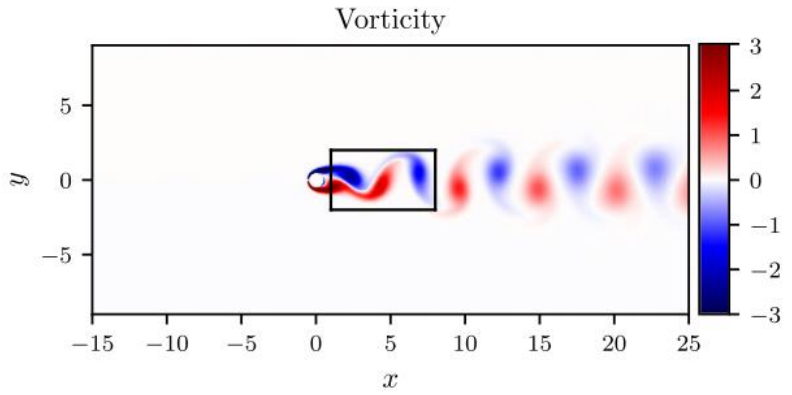
Second step: minimize MSE to satisfy boundary conditions for $t \neq 0$ and the general PDE.

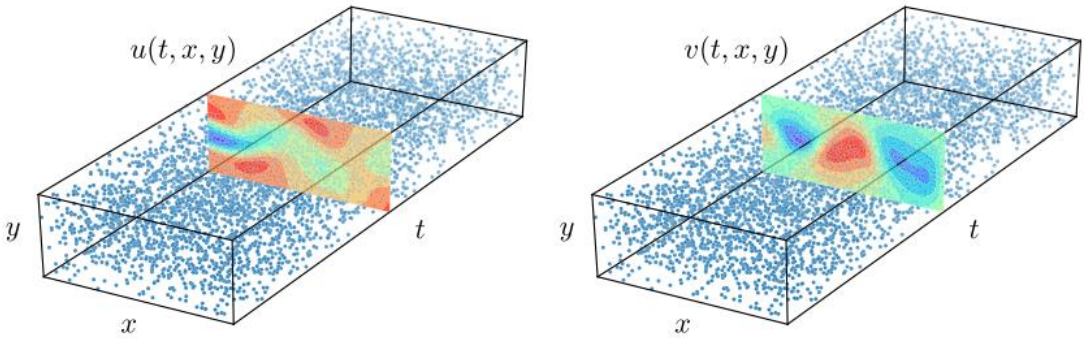
$$MSE := \frac{1}{N}\sum_{i=1}^{N}\left(|u(t^i, x^i, y^i) - u^i|^2 + |v(t^i, x^i, y^i) - v^i|^2\right) + \frac{1}{N}\sum_{i=1}^{N}\left(|f(t^i, x^i, y^i)|^2 + |g(t^i, x^i, y^i)|^2\right).$$

Example 3 – Navier–Stokes equation (2)

5000 scattered data points

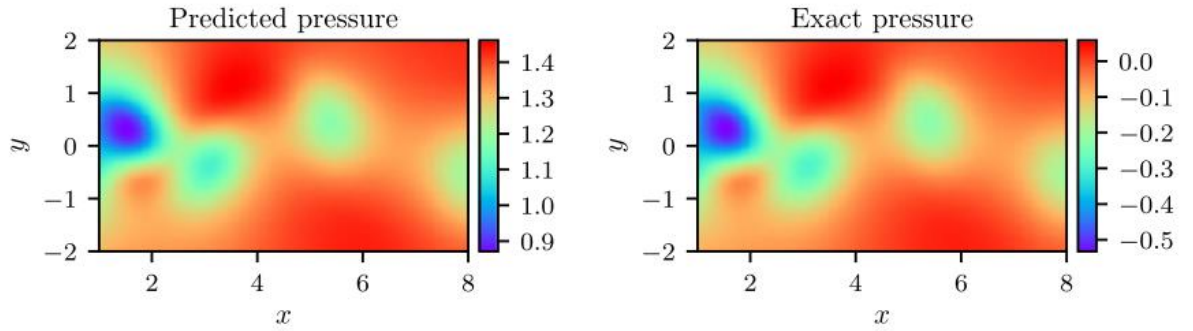
Training Data





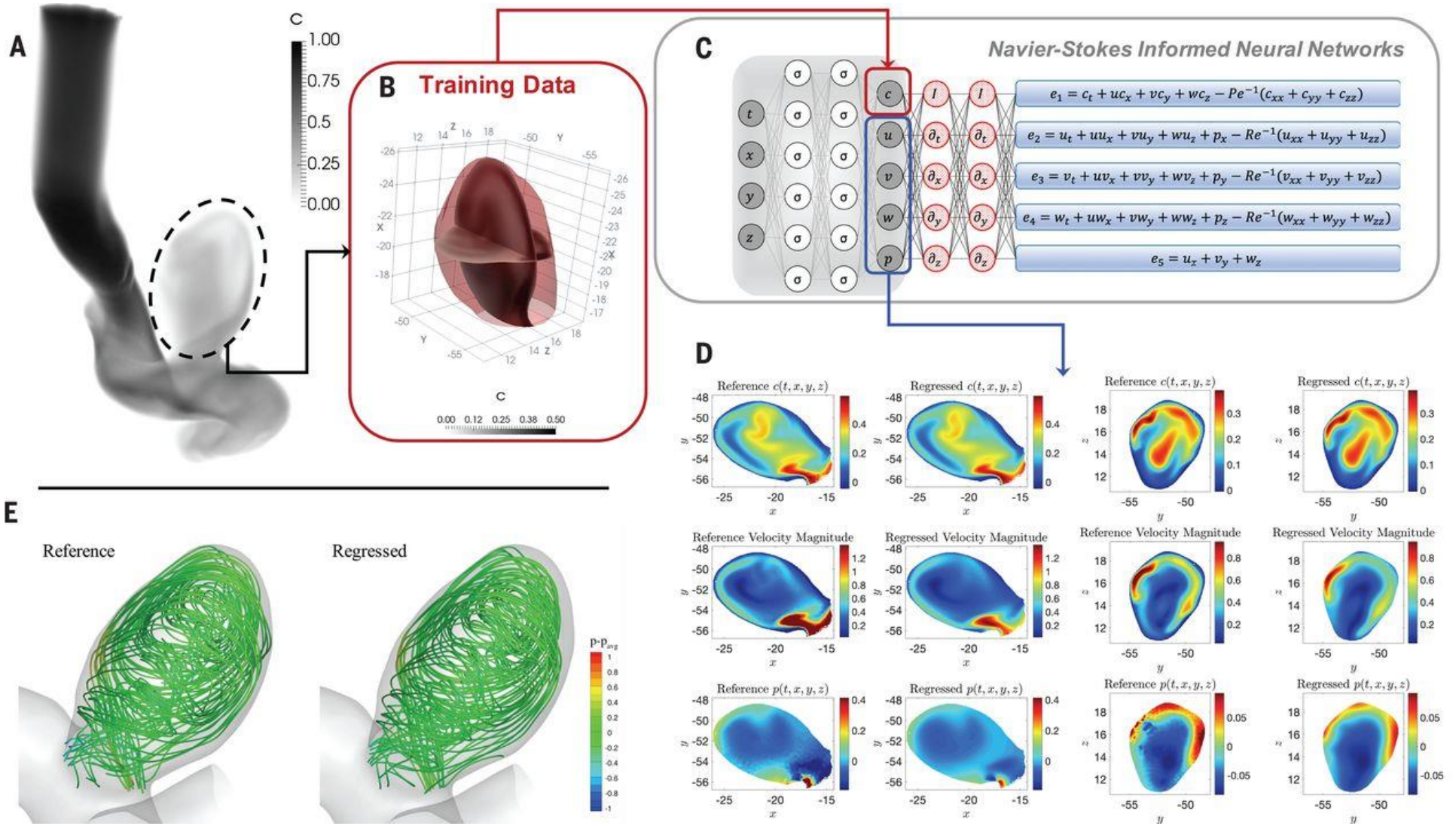
Stream wise and transverse velocity

2 Results



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.999(uu_x + vu_y) = -p_x + 0.01047(u_{xx} + u_{yy})$ $v_t + 0.999(uv_x + vv_y) = -p_y + 0.01047(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.998(uu_x + vu_y) = -p_x + 0.01057(u_{xx} + u_{yy})$ $v_t + 0.998(uv_x + vv_y) = -p_y + 0.01057(v_{xx} + v_{yy})$

Example 4 – Hidden Fluid Mechanics



Maziar Raissi et. al (2020) Science (https://science.sciencemag.org/content/367/6481/1026)

https://maziarraissi.github.io/research/09_hidden_fluid_mechanics/

Bayesian Physics Informed Neural Networks (B-PINNs)

Statistics > Machine Learning

[Submitted on 13 Mar 2020]

B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems with Noisy Data

Liu Yang, Xuhui Meng, George Em Karniadakis

We propose a Bayesian physics-informed neural network (B-PINN) to solve both forward and inverse nonlinear problems described by partial differential equations (PDEs) and noisy data. In this Bayesian framework, the Bayesian neural network (BNN) combined with a PINN for PDEs serves as the prior while the Hamiltonian Monte Carlo (HMC) or the variational inference (VI) could serve as an estimator of the posterior. B-PINNs make use of both physical laws and scattered noisy measurements to provide predictions and quantify the aleatoric uncertainty arising from the noisy data in the Bayesian framework. Compared with PINNs, in addition to uncertainty quantification, B-PINNs obtain more accurate predictions in scenarios with large noise due to their capability of avoiding overfitting. We conduct a systematic comparison between the two different approaches for the B-PINN posterior estimation (i.e., HMC or VI), along with dropout used for quantifying uncertainty in deep neural networks. Our experiments show that HMC is more suitable than VI for the B-PINNs posterior estimation, while dropout employed in PINNs can hardly provide accurate predictions with reasonable uncertainty. Finally, we replace the BNN in the prior with a truncated Karhunen-Loève (KL) expansion combined with HMC or a deep normalizing flow (DNF) model as posterior estimators. The KL is as accurate as BNN and much faster but this framework cannot be easily extended to high-dimensional problems unlike the BNN based framework.

TLDR: Noisy/Real data → Need uncertainty quantification → Bayesian approach is good → combine

Maybe we can see some examples ?

[Submitted on 11 May 2020 (v1), last revised 16 Sep 2020 (this version, v2)]

SciANN: A Keras/Tensorflow wrapper for scientific computations and physics-informed deep learning using artificial neural networks

Ehsan Haghighat, Ruben Juanes

In this paper, we introduce SciANN, a Python package for scientific computing and physics-informed deep learning using artificial neural networks. SciANN uses the widely used deep-learning packages Tensorflow and Keras to build deep neural networks and optimization models, thus inheriting many of Keras's functionalities, such as batch optimization and model reuse for transfer learning. SciANN is designed to abstract neural network construction for scientific computations and solution and discovery of partial differential equations (PDE) using the physics-informed neural networks (PINN) architecture, therefore providing the flexibility to set up complex functional forms. We illustrate, in a series of examples, how the framework can be used for curve fitting on discrete data, and for solution and discovery of PDEs in strong and weak forms. We summarize the features currently available in SciANN, and also outline ongoing and future developments.