Machine Learning & Artificial Intelligence for Physics

Part 1: Principles & Key Tools

Julia Gonski

17 July 2024
African School of Physics
Marrakech Morocco

ATLAS EXPERIMENT

SLAC NATIONAL ACCELERATOR LABORATORY

U.S. DEPARTMENT OF ENERGY
Office of Science

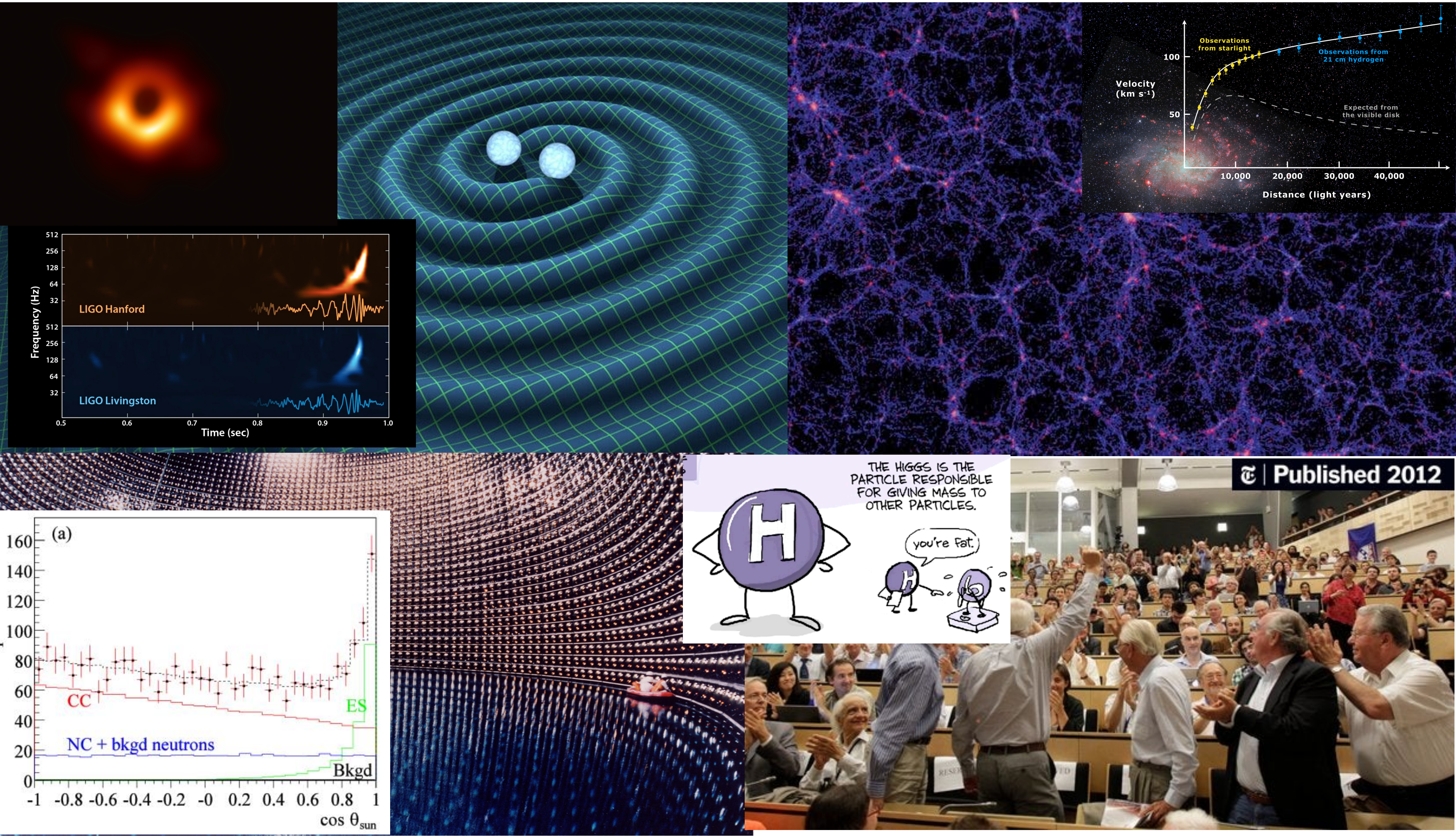
Outline

- Lecture 1: principles & key tools
  - What is AI/ML? How is it useful in physics research?
  - Basics of neural nets: architecture & development

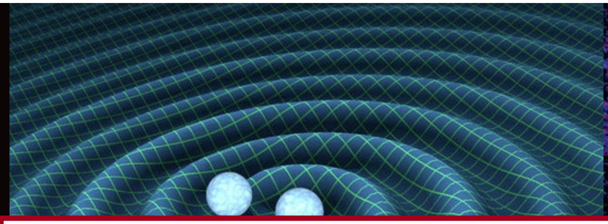
- Lecture 2: applications & advanced models
  - Anomaly detection
  - Geometrical ML
  - Hardware acceleration

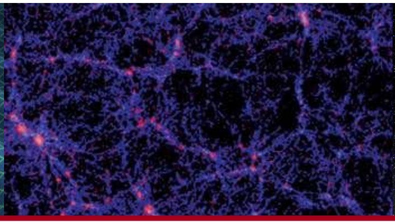
The Why & How of Physics Research

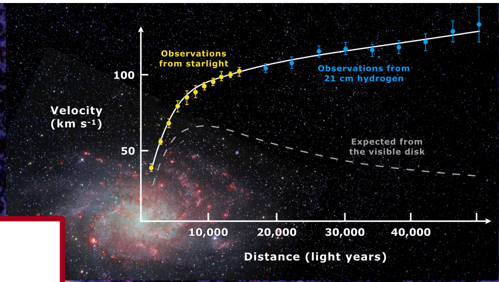
SLAC

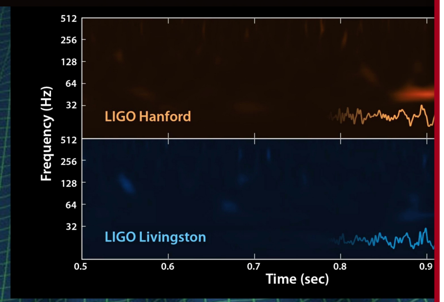


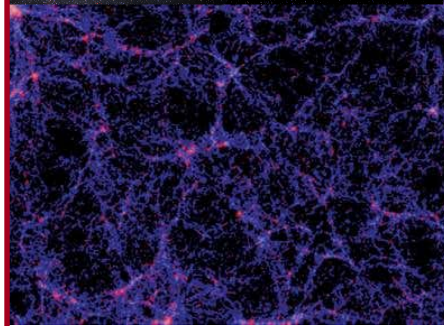
The Why & How of Physics Research











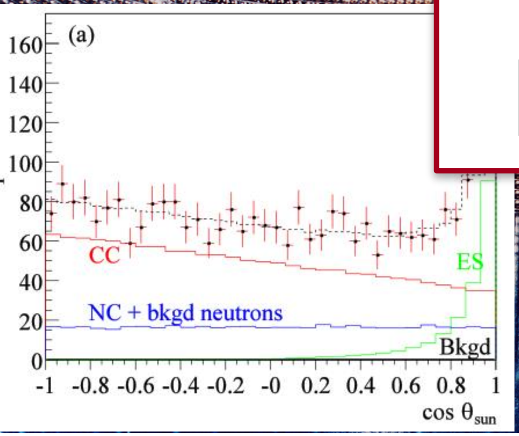
Big Questions

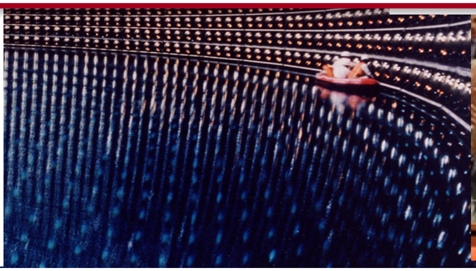
↓

Theoretical Modeling

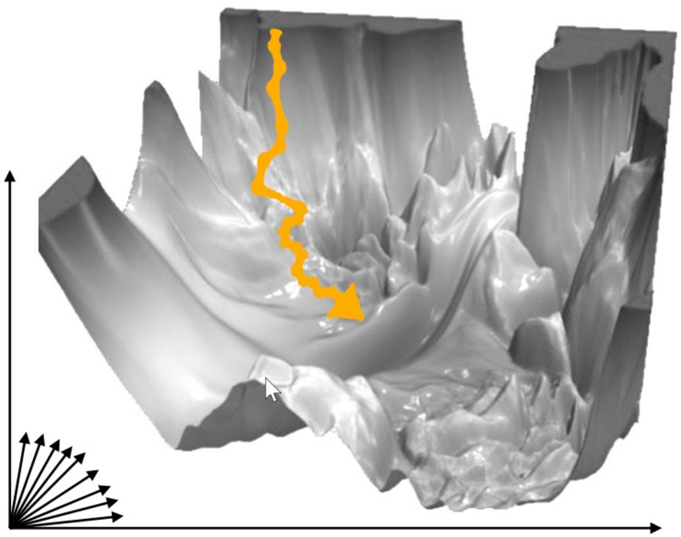
↓

Experimental Tests





The AI Revolution

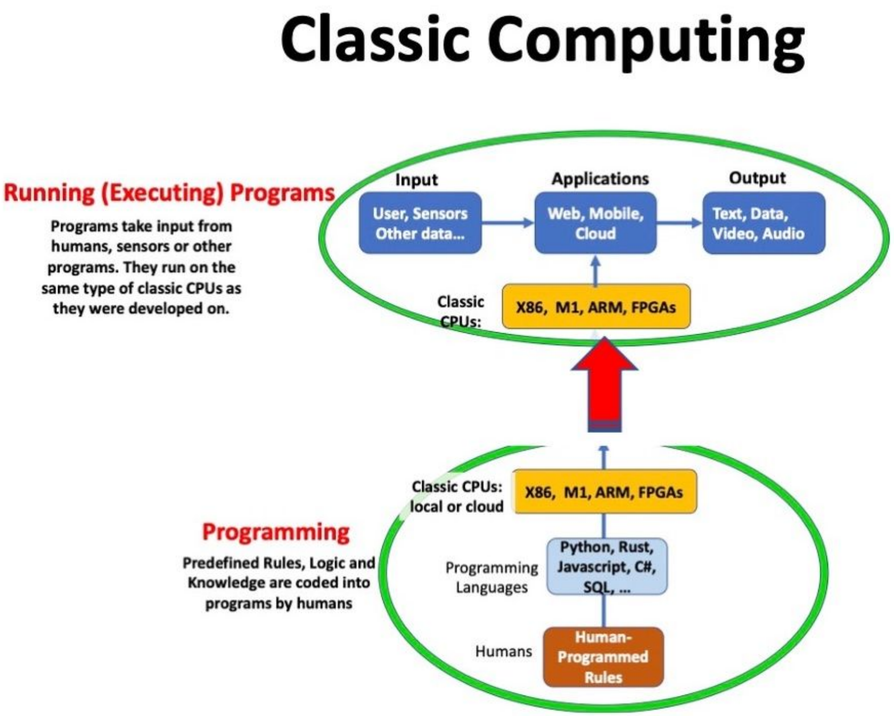


Billion parameter gradient descent

Unreal Engine Kite Demo (Epic Games 2015)

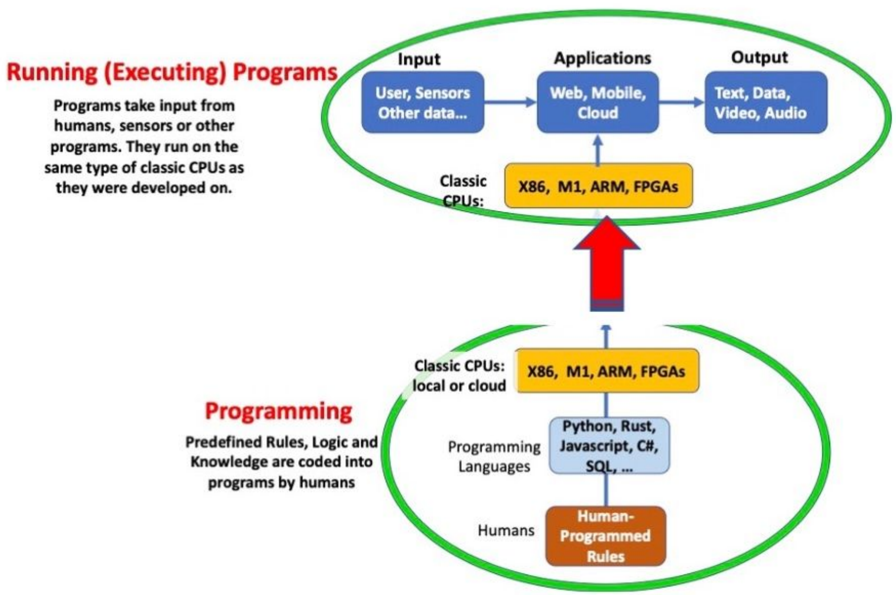
Prompt: a landscape from the Moon with the Earth setting on the horizon, realistic, detailed
Negative prompt: whimsical interpretation of the prompt
Parameters: Steps: 30, Sampler: Euler a, CFG scale: 7.0, Seed: 4252913504, Size: 512x384, Model hash: 82aac931

The “Machine” in Machine Learning

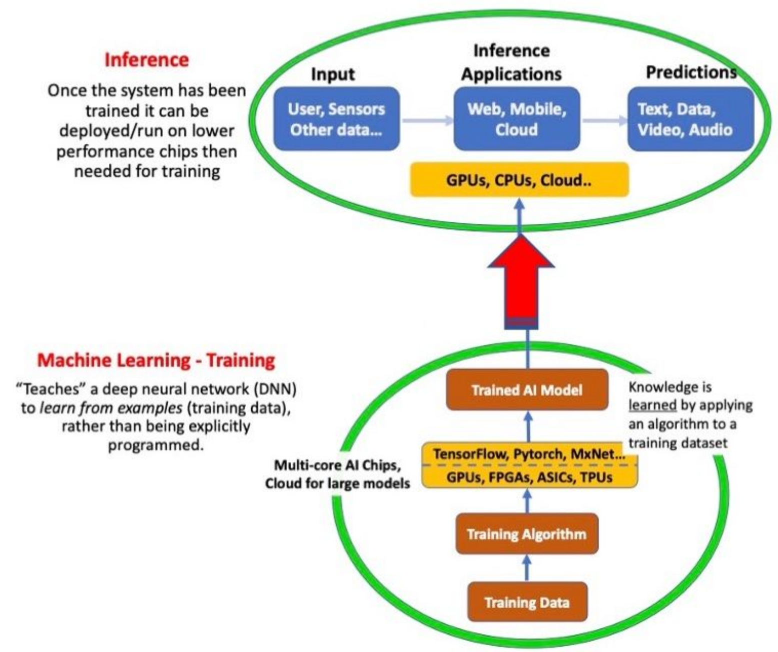


The “Machine” in Machine Learning

Classic Computing



Machine Learning

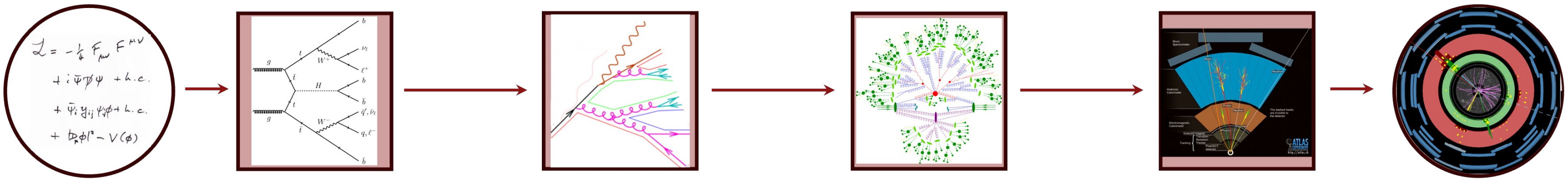


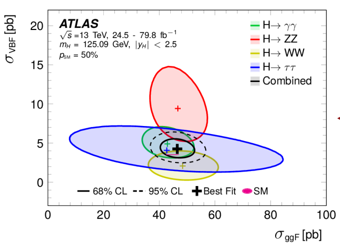
Fundamental Physics ↔ AI/ML

O(10) Physics parameters

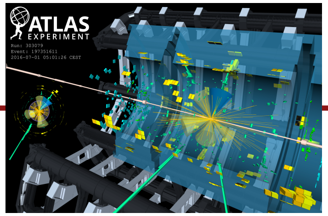
O(10^9) sensor elements

Theory   Hard Scatter   Parton Shower   Hadronization   Material Interactions   Observations

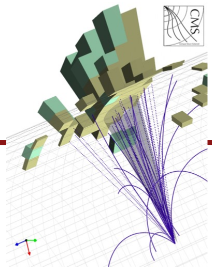




Statistical Inference



Population selection & analysis



Reconstruct objects

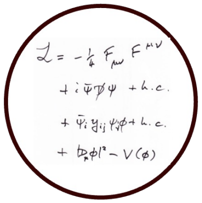
M. Kagan

Fundamental Physics ↔ AI/ML

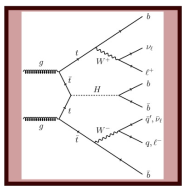
SLAC

Generate plausible, high-dim. data from high-level concepts

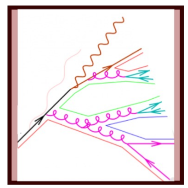
Theory



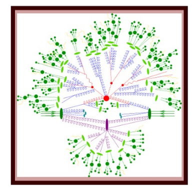
Hard Scatter



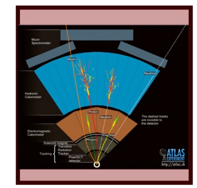
Parton Shower



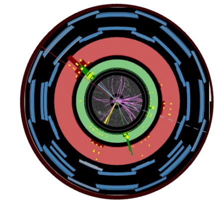
Hadronization

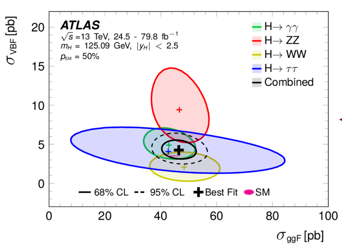


Material Interactions

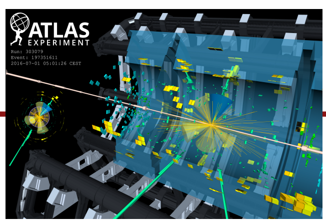


Observations

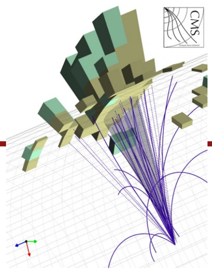




Statistical Inference



Population selection & analysis

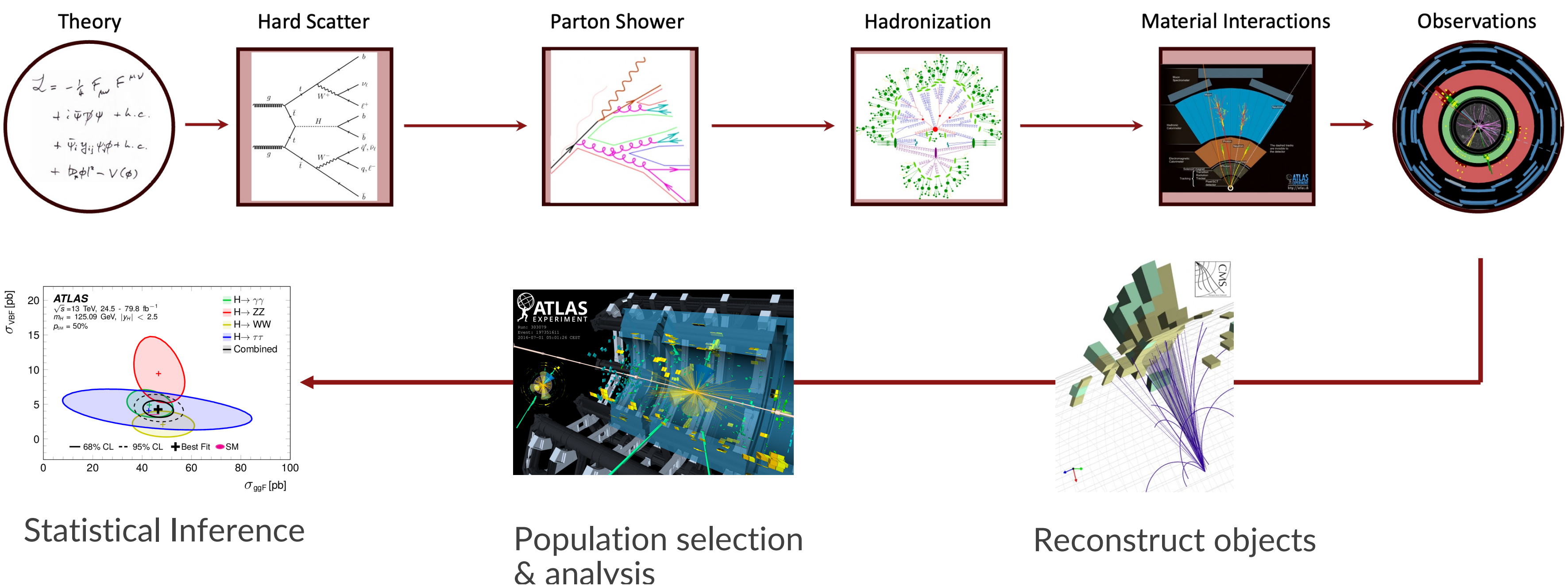


Reconstruct objects

M. Kagan

Fundamental Physics ↔ AI/ML

Generate plausible, high-dim. data from high-level concepts



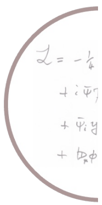
Extract high-level concepts from low-level, high-dim. data

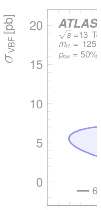
M. Kagan

Fundamental Physics ↔ AI/ML

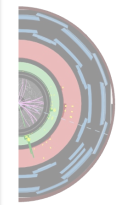
G ts

The ations





Stati



Generate plausible, high-dim data from high-level concepts

Prompt:
street style photo of a woman selling pho at a Vietnamese street market, sunset, shot on fujifilm

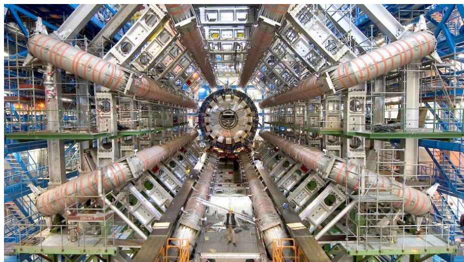
Extract high-level concepts from low-level, high-dim. data

Classification:
A photo of guacamole, a type of food

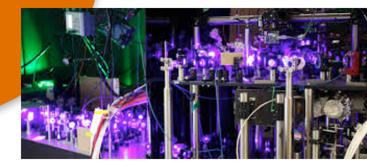
Extract high-level concepts from low-level, high-dim. data

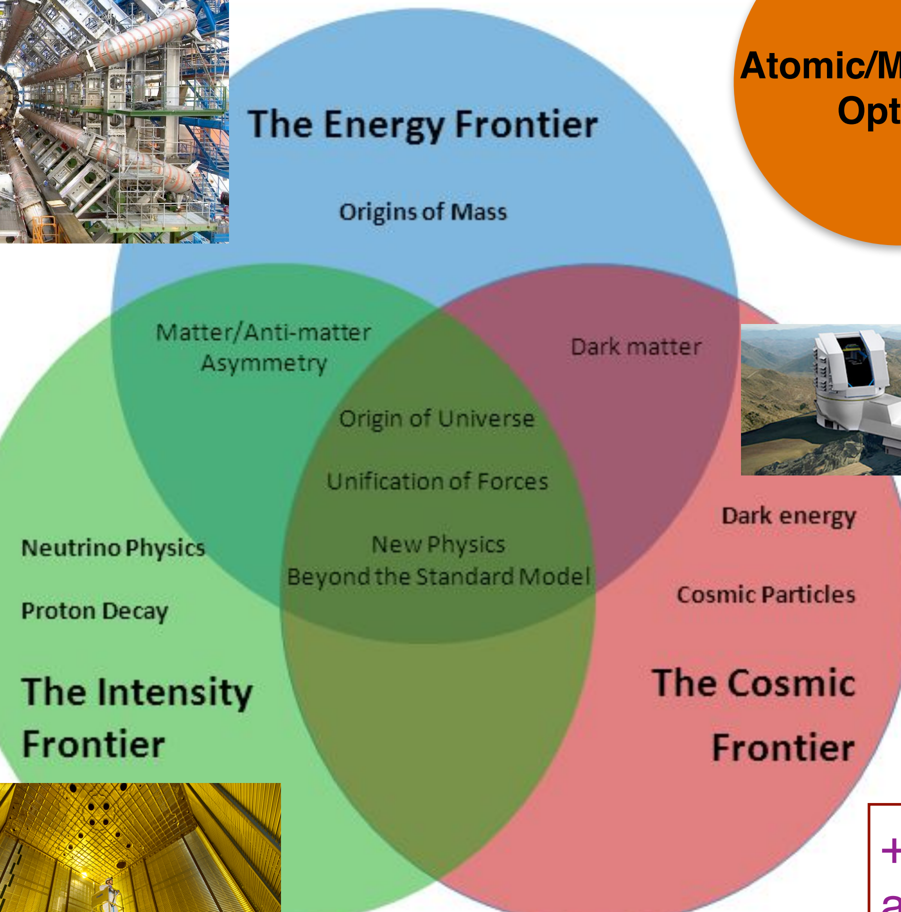
M. Kagan

Frontiers of Physics

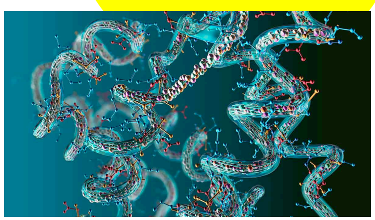


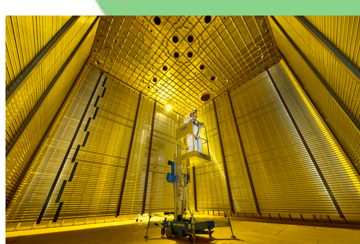
Atomic/Molecular/ Optical





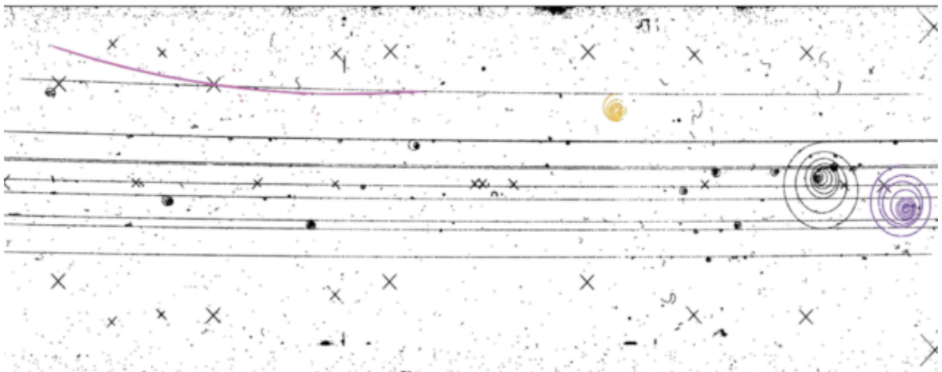
Biophysics

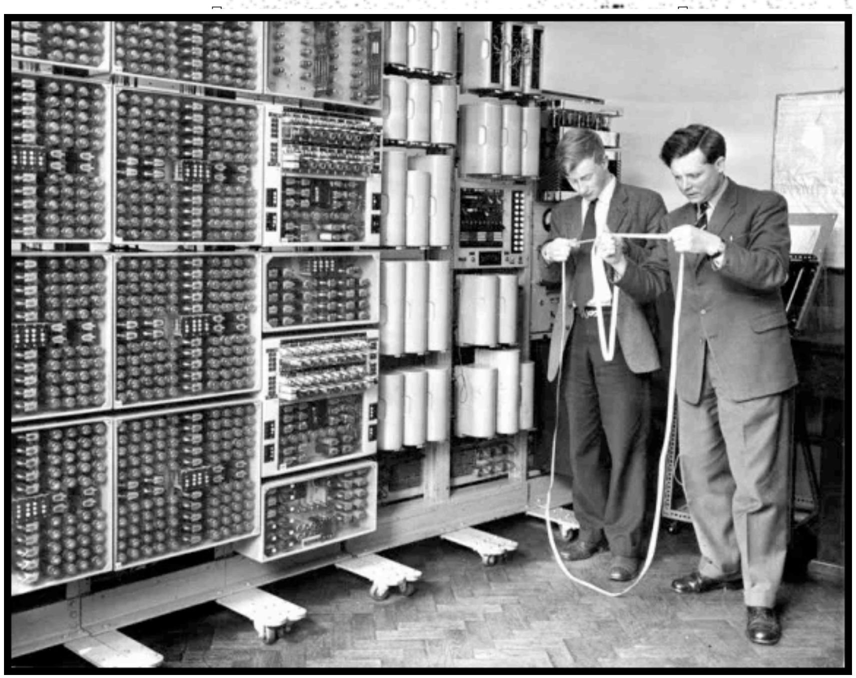


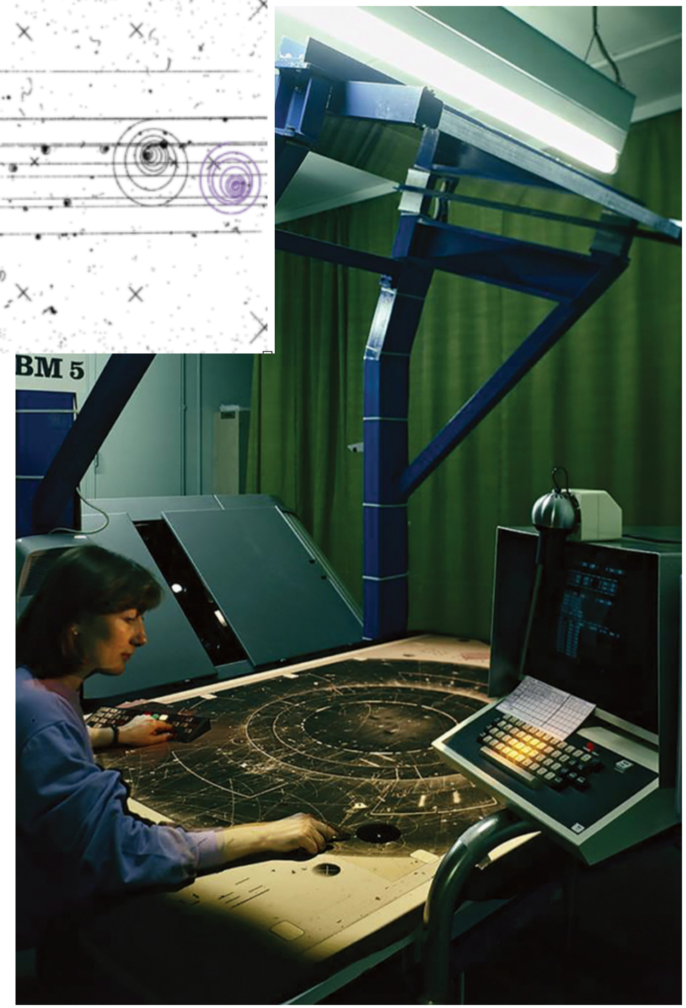


+ theory, computing, advanced technology, & more!

How it All Began…



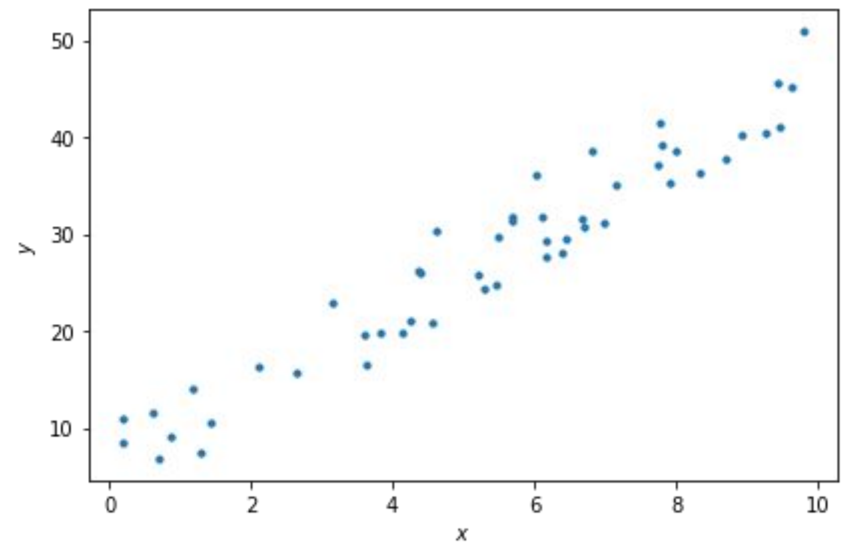


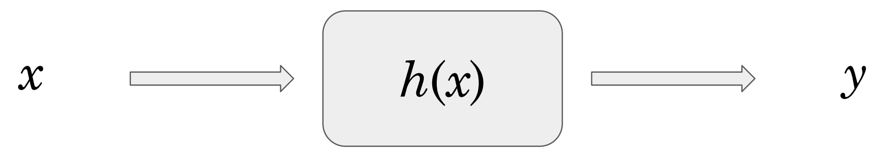


ML Building Blocks: Linear Regression

- Fitting a straight line to a linear function

- **Cost function** = returns a global error between the predicted values from a mapping function *h(x)* (predictions) and all the target values (observations) of the training data set





$$h_\theta(x) = \theta_0 + \theta_1\ x$$

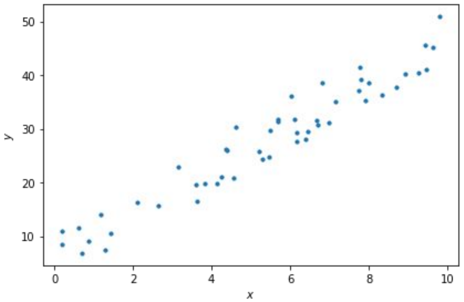
1D ⇒ 2 parameters:

$$\boldsymbol{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix}$$

Cost Functions

- <u>Ex. linear regression</u>: cost function = mean-squared error

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

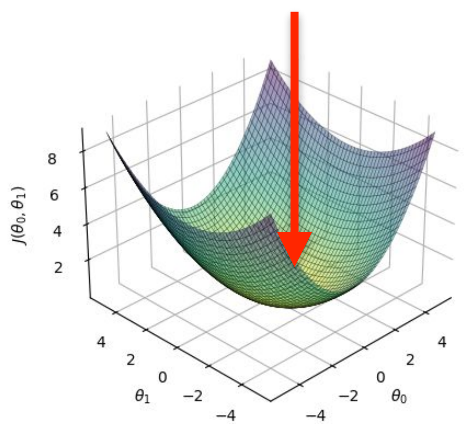


"Fit the data"

Find $\boldsymbol{\theta}$ parameters

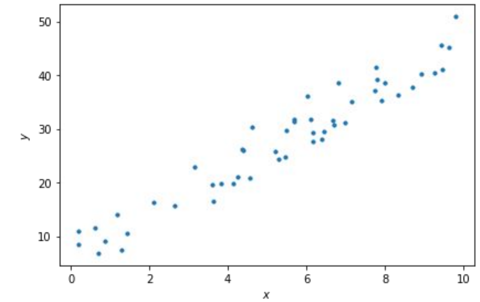
minimizing the cost J

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$



Cost Functions

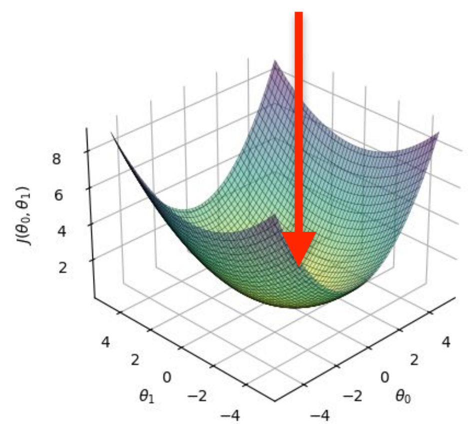
- Ex. linear regression: cost function = mean-squared error



$$J\left(\theta_0, \theta_1\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

"Fit the data" ⇨ Find $\boldsymbol{\theta}$ parameters minimizing the cost J

$$\min_{\theta_0, \theta_1} J\left(\theta_0, \theta_1\right)$$



➡ 1-dimensional **gradient descent!**

Gradient Descent

- **Gradient descent**: iterative procedure to update the parameters in the direction of 'descending gradient', i.e. dictated by the slope of the cost function's partial derivatives for each parameter
  - Until: N iterations or partial derivatives →0
  - Convergence to values for the parameters minimizing the cost function

(θ_0, θ_1)

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0' = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\theta_1' = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

Hyperparameter α = learning rate

n input features

m data samples

	x_1	x_2	x_3	x_4
0	53.844621	-2.236084	2158.276717	-0.035128
1	45.108137	1.204087	1935.968922	0.038599
2	43.764648	-1.509527	2080.412520	-0.080645
3	44.739359	-2.604297	2048.357208	-0.040369
4	38.390222	-2.325613	1984.086668	0.243353
5	49.152215	-0.600885	1972.311885	-0.034590

Shallow Learning: The Neural Net

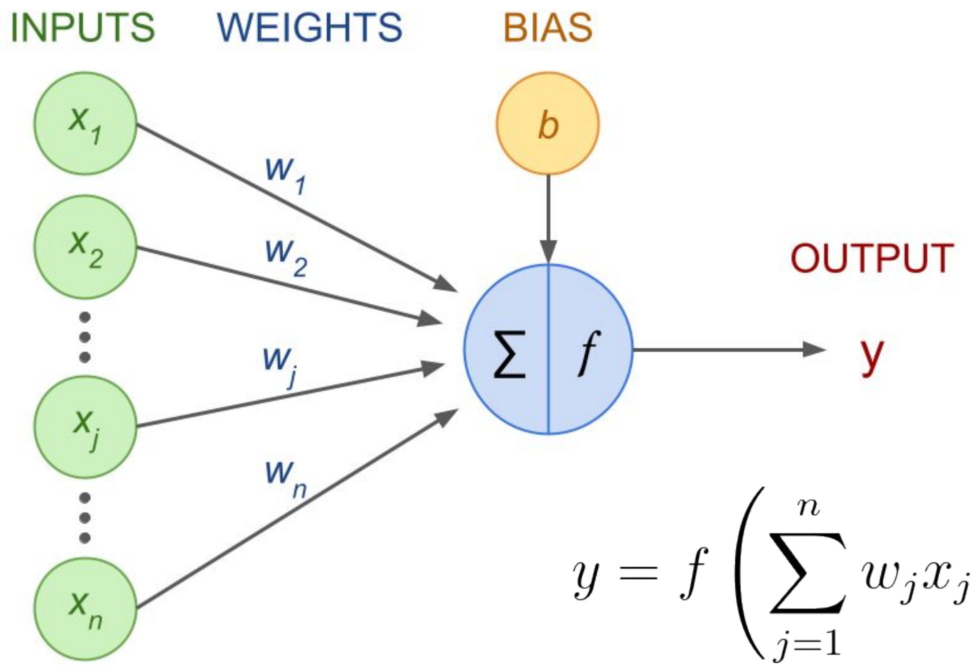
x_j : input nodes

w_j: weights

b : bias term

Σ: weighted sum

f: activation function



$$y = f\left(\sum_{j=1}^{n} w_j x_j + b\right)$$

Claire David African School of Physics 2022

Shallow Learning: The Neural Net

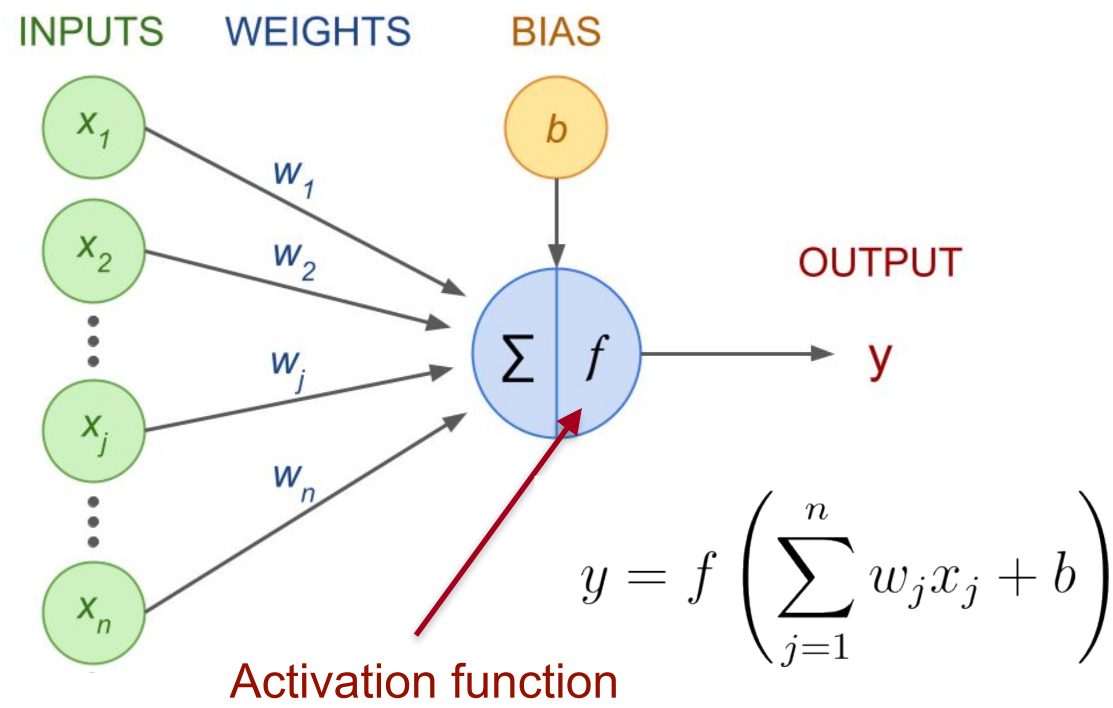
x_j : input nodes

w_j: weights

b : bias term

Σ: weighted sum

f: activation function



Claire David African School of Physics 2022

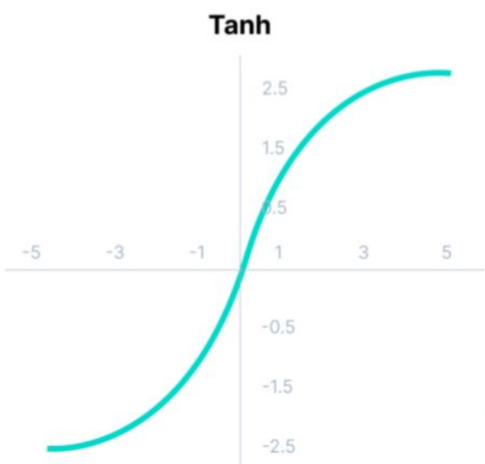
The Activation Function

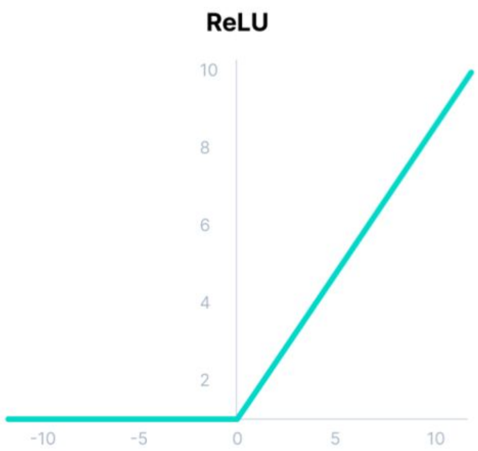
The **Activation Function** is a mathematical operation deciding whether the neuron's input to the network is important or not.

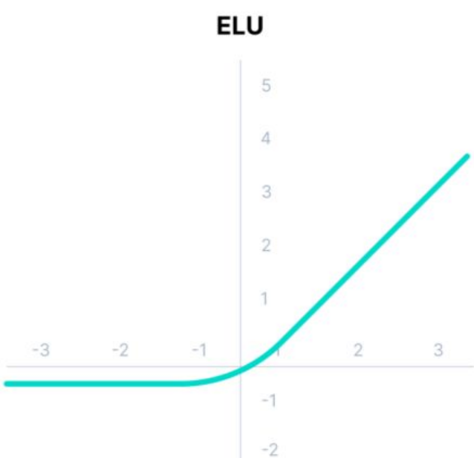
Returning a non-zero values means the neuron is "activated", or "fired."

The purpose of the activation function is to **introduce non-linearity** into the output of a neural network.

- **Differentiable**, ideally continuously differentiable
- **Non-linear:** the only way for the network to learn







Shallow Learning: The Neural Net

x_j : input nodes

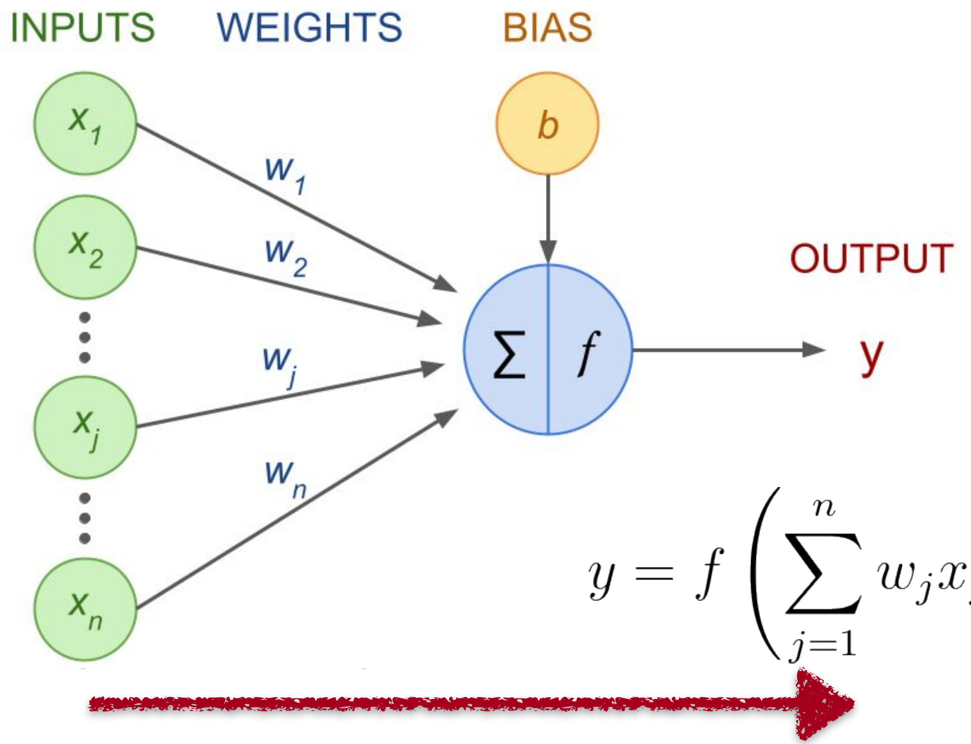
w_j: weights

b : bias term

Σ: weighted sum

f: activation function

Claire David African School of Physics 2022



$$y = f\left(\sum_{j=1}^{n} w_j x_j + b\right)$$

- **Feed-forward propagation:** process of computing all activation units of a NN
  - At last layer (output), leads to predictions (which are compared to observations in training)

Shallow Learning: The Neural Net

x_j : input nodes

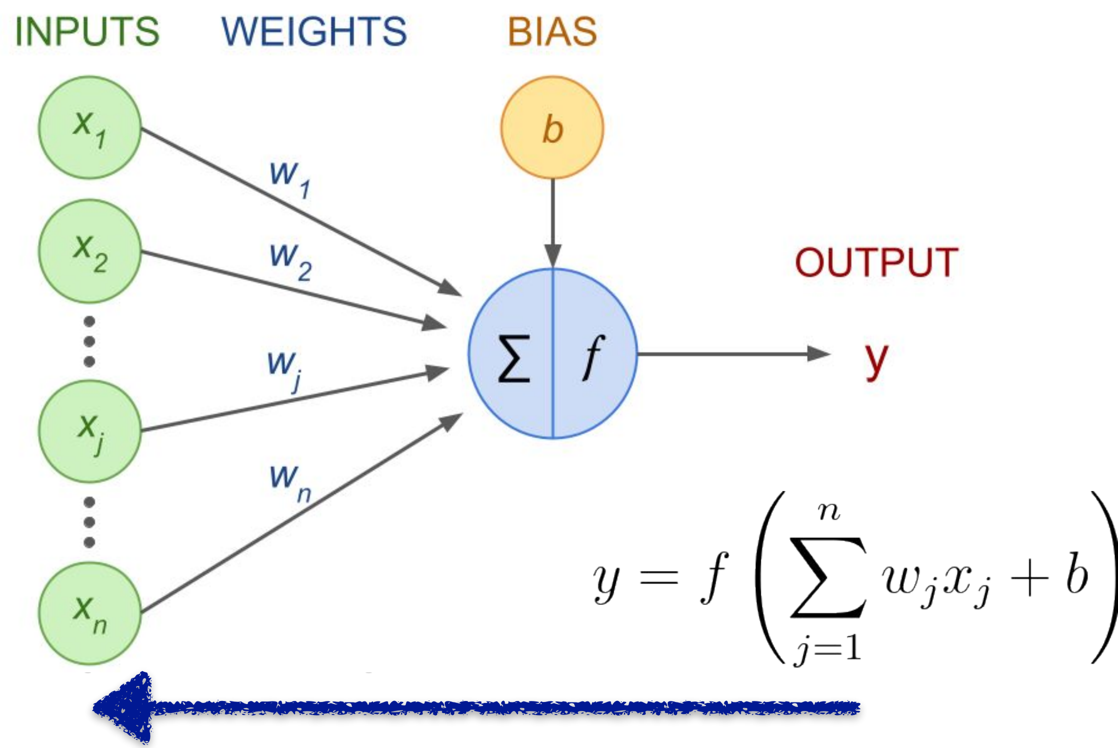
w_j: weights

b : bias term

Σ: weighted sum

f: activation function

Claire David African School of Physics 2022



$$y = f\left(\sum_{j=1}^{n} w_j x_j + b\right)$$

- **Backpropagation:** uses the chain rule to compute how much each activation unit contributed to the overall error & adjust weights/biases to reduce overall error

Deep Learning: The Power of Scale

- More dataset complexity means you need a larger model, which means you need more input data, which means you need more computational power!

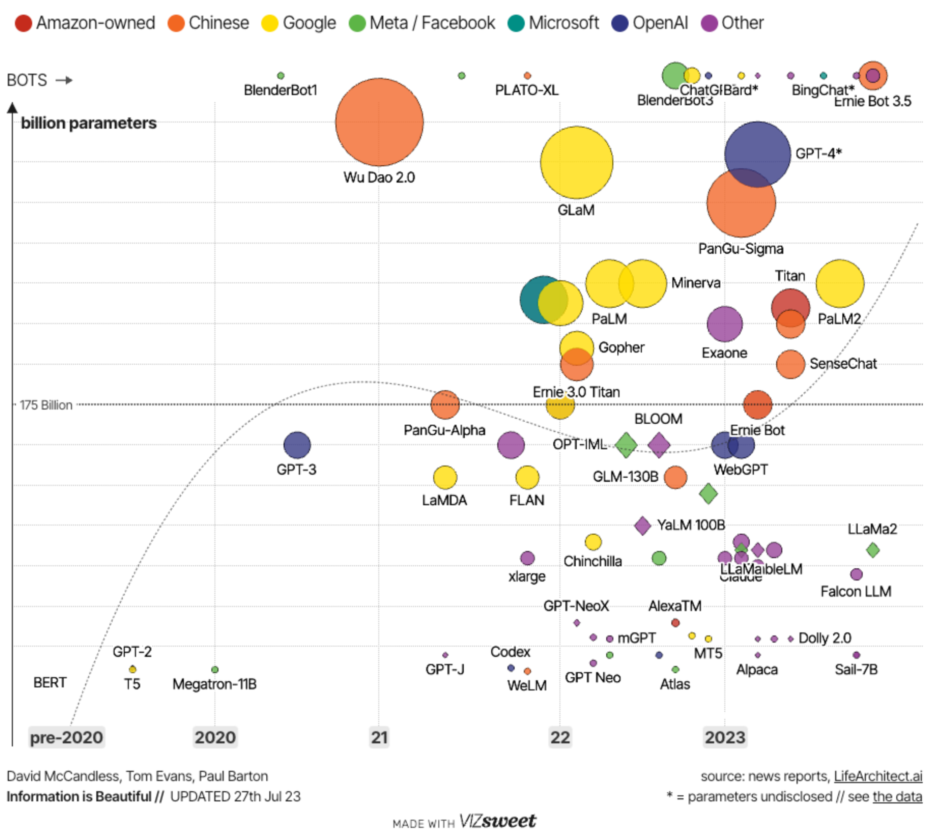
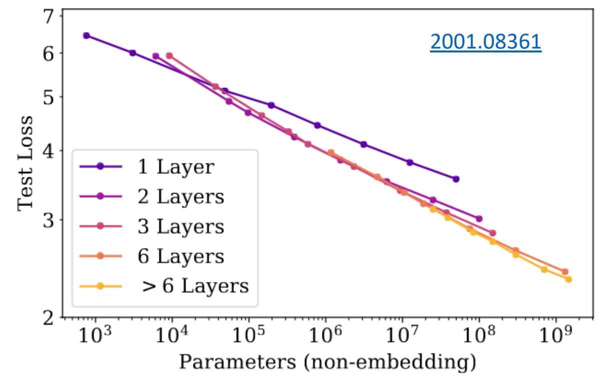
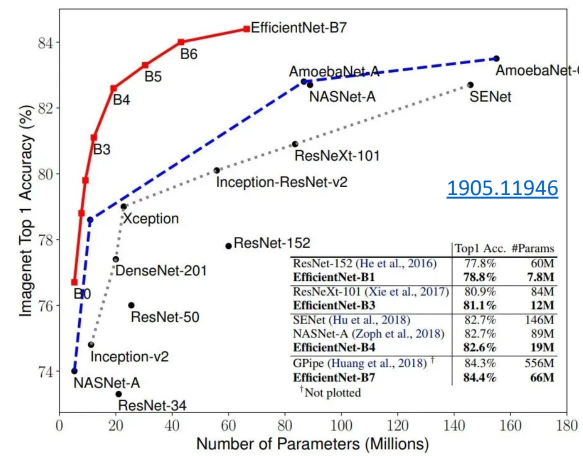


Image credit: D. McCandless, T. Evans, P. Barton





	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

† Not plotted

Putting it in Action

1. Select your training data

2. Choose an input modeling

3. Build & train your model

4. Gauge performance (ROC/AUC)

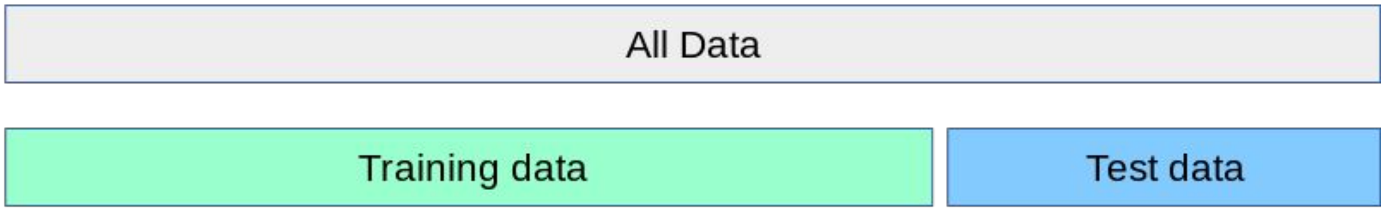
5. Optimize (hyperparameters)

1. Selecting the Data

Training set: dedicated to the fitting procedure (minimizing the cost function).

Validation set: assess the performance of the model & tune the model’s hyperparameters.

Test set: final assessment done on the model → error rate is called **generalization error.**

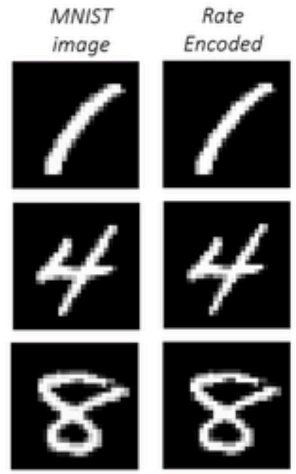


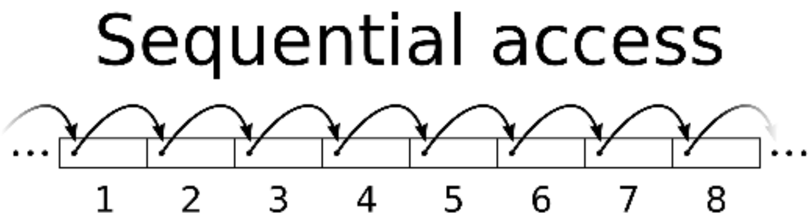
C. David

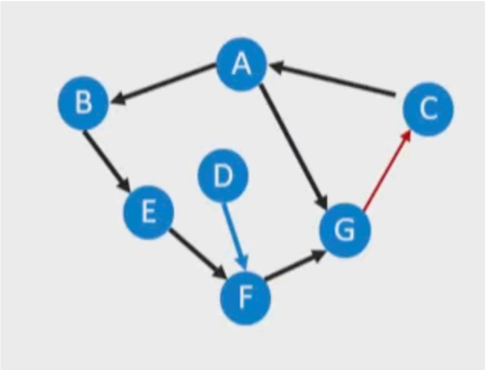
2. Input Modeling

- What does your data *naturally* look like?
  - An image? → convolutional layers
  - A sequence? → recurrent architecture
  - A graph? → geometrical ML (graphs, transformers)

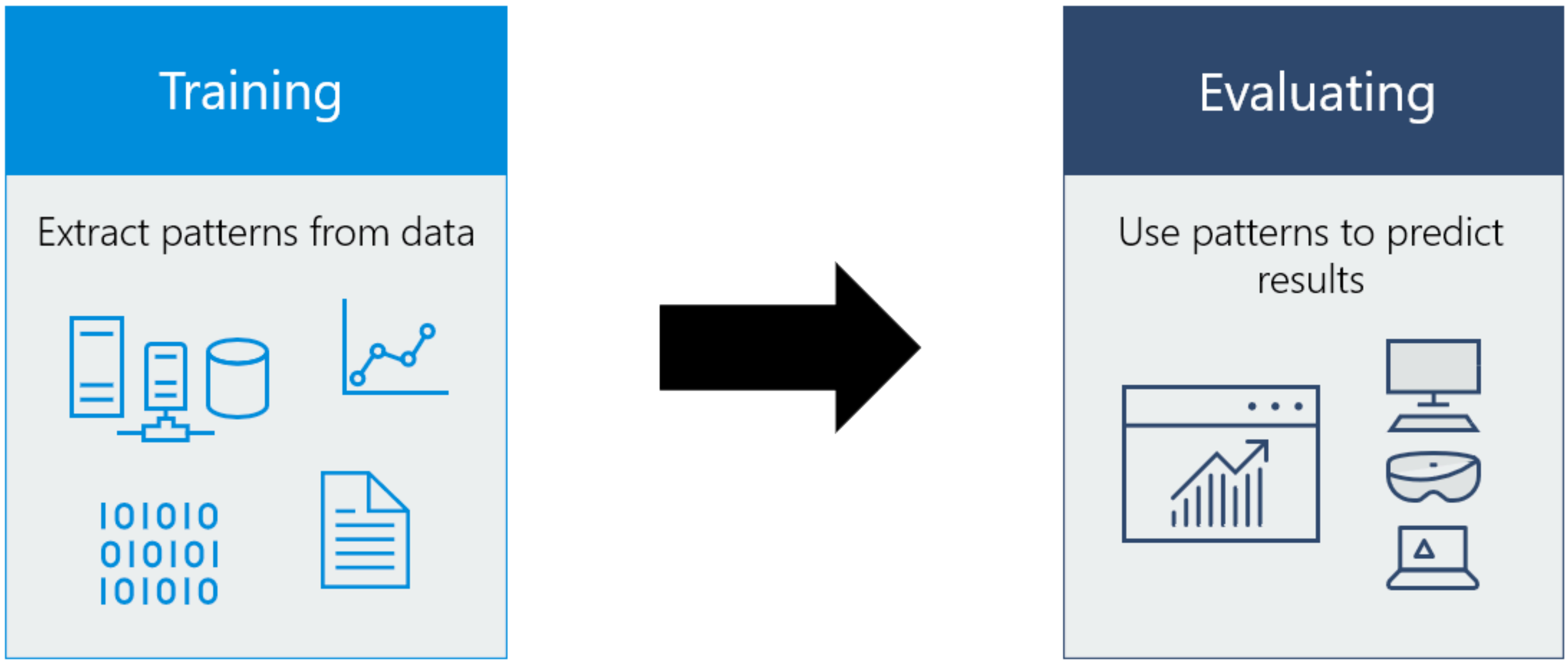
- What information can I give my ML tool to help it best learn from the data?



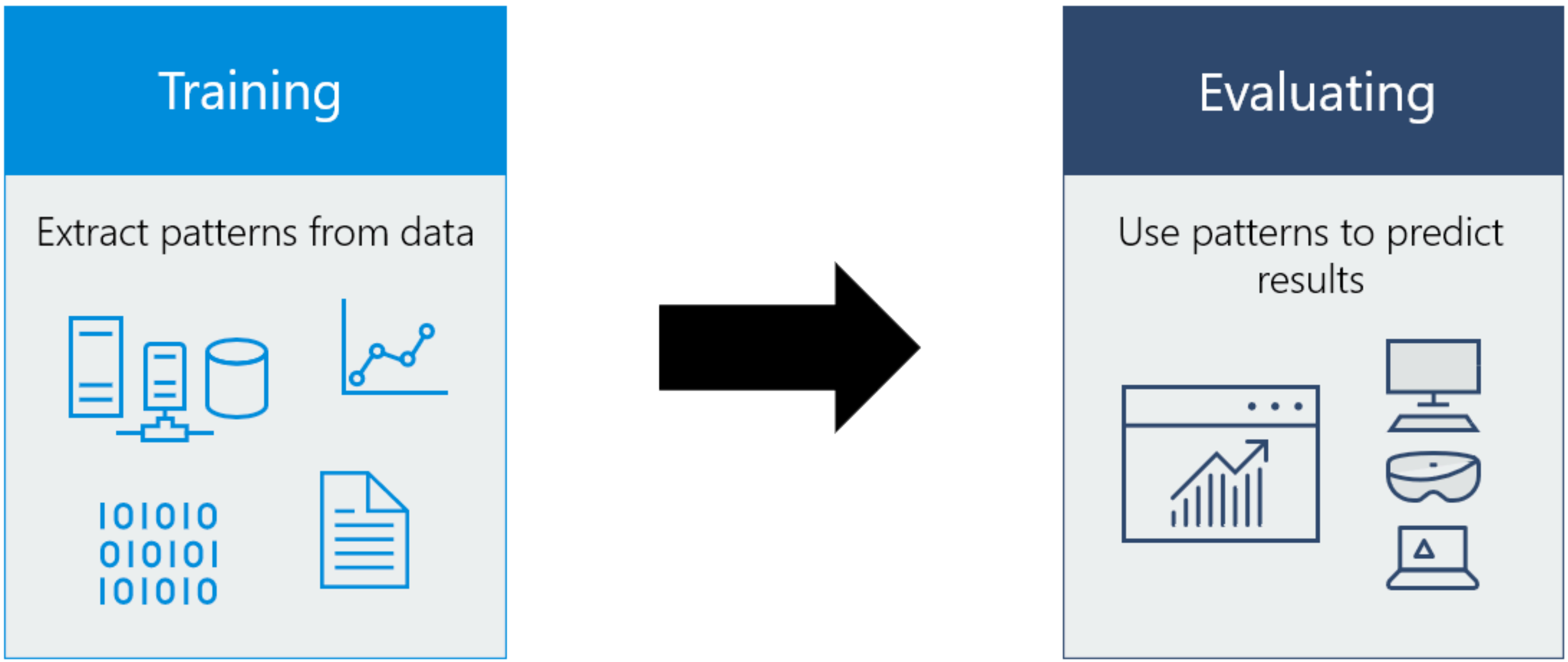




3. Building, Training, and Evaluating the Model



3. Building, Training, and Evaluating the Model



Epoch: iteration of training calculations

Example:
- **Classification:** predict the category of the input
- **Regression:** predict value of key quantity

3. Building, Training, and Evaluating the Model

- This is all very easy to implement!
- Pre-existing functionality through ML python packages such as keras or PyTorch

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Input Data

Sequential Model

Train

Evaluate

4. Quantifying Performance

The Confusion Matrix

we correctly predict signal

we predict background but it was signal It's a miss!

		PREDICTED CLASS	
		1	0
ACTUAL CLASS	1	TRUE POSITIVE	FALSE NEGATIVE Type II error
	0	FALSE POSITIVE Type I error	TRUE NEGATIVE

we predict signal but it's background

we correctly predict background

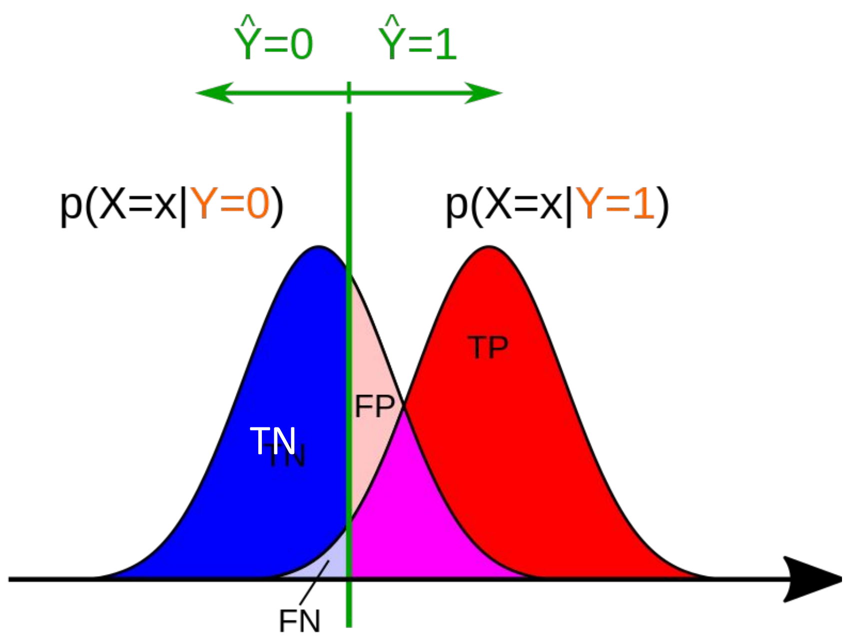
4. Quantifying Performance

The **Receiver Operating Characteristic (ROC)** curve is a graphical display that plot the True Positive Rate (TPR) against the False Positive Rate (FPR) for each value of the decision threshold going over the classifier’s output score range.

Reducing threshold:

TPR ↗

FPR ↘



Increasing threshold:

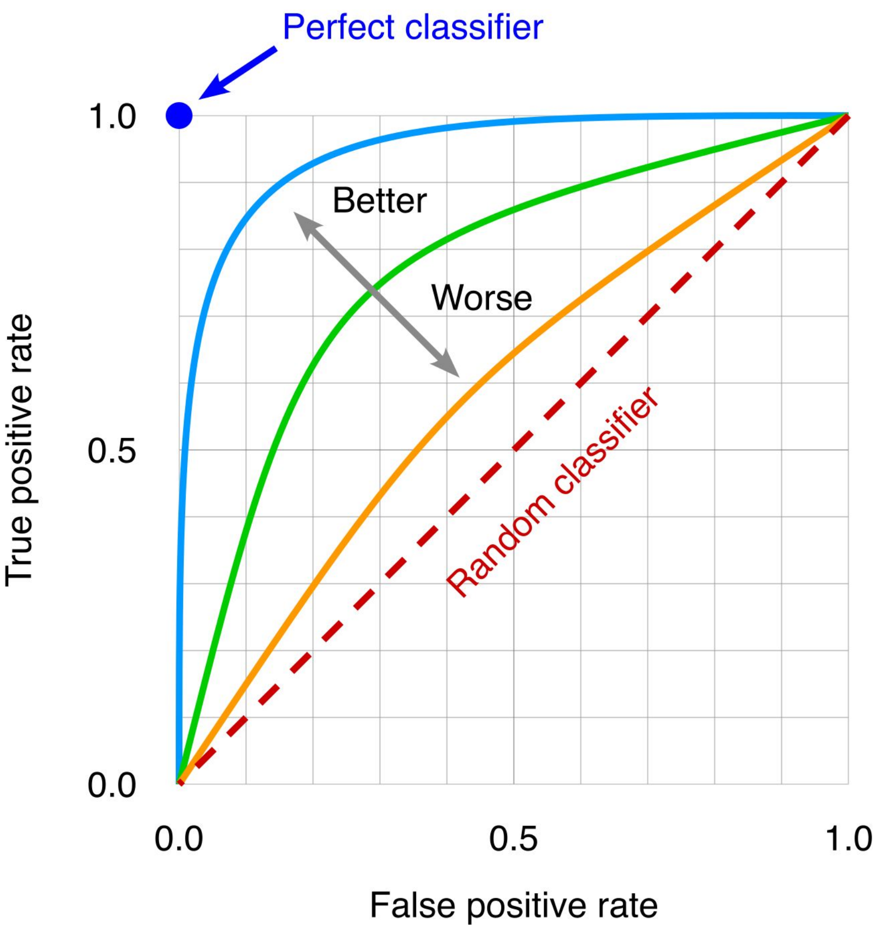
↘ TPR

↗ FPR

Receiver-Operating Characteristic (ROC)

The **Area Under Curve (AUC)** is the integral of the ROC curve, from FPR = 0 ⟶ 1

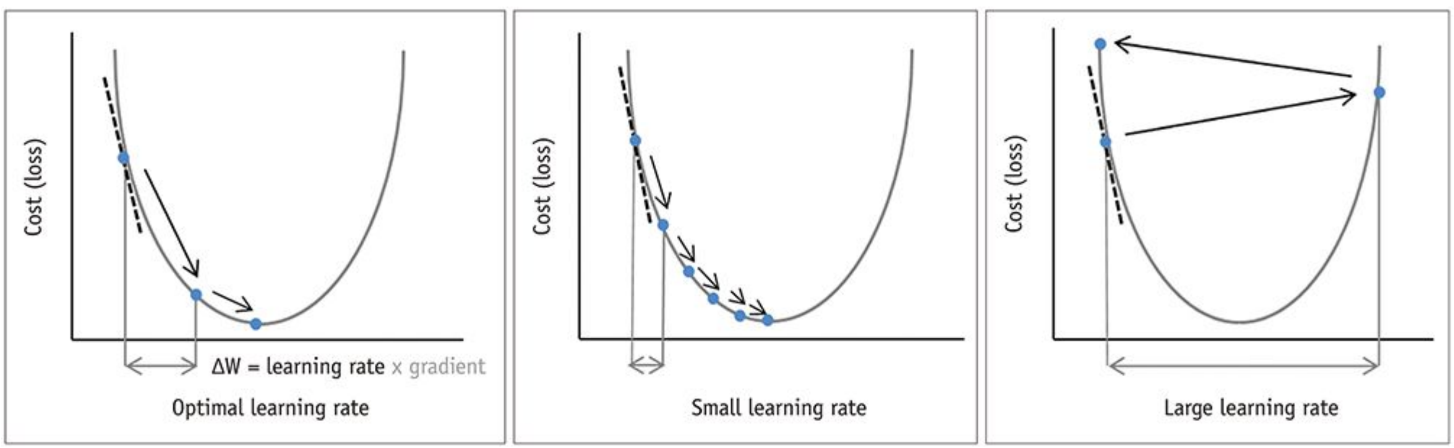
A perfect classifier has AUC = 1.



C. David

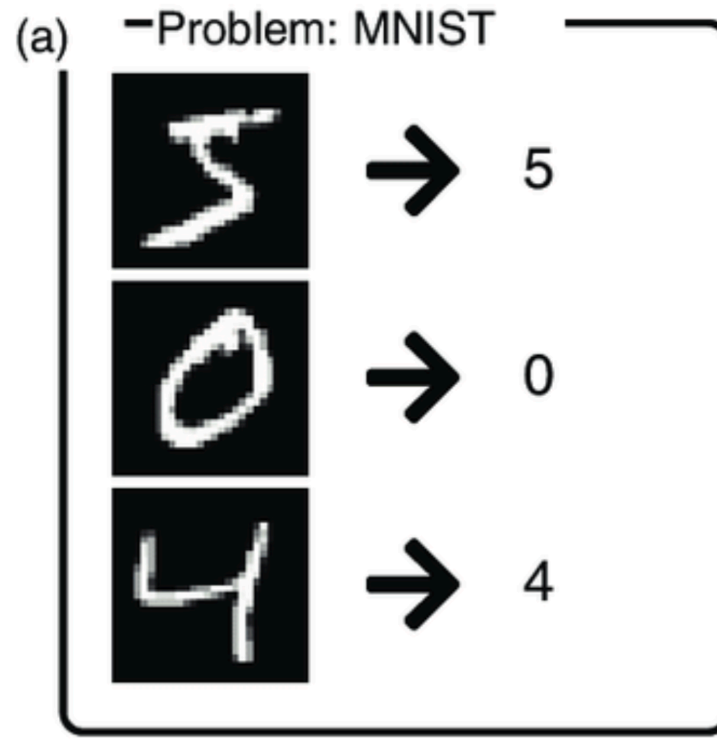
5. Optimization

- Model parameters = basic structure (number of layers/nodes, weights/biases)
- **Hyperparameters**: choosable parameters of model that can significantly affect performance
    - Must be well-matched to complexity of training dataset and input modeling
        - Batch size (how many events feed into a single update)
        - Train-test split ratio
        - Learning rate: how large is the step in your gradient descent? (below)



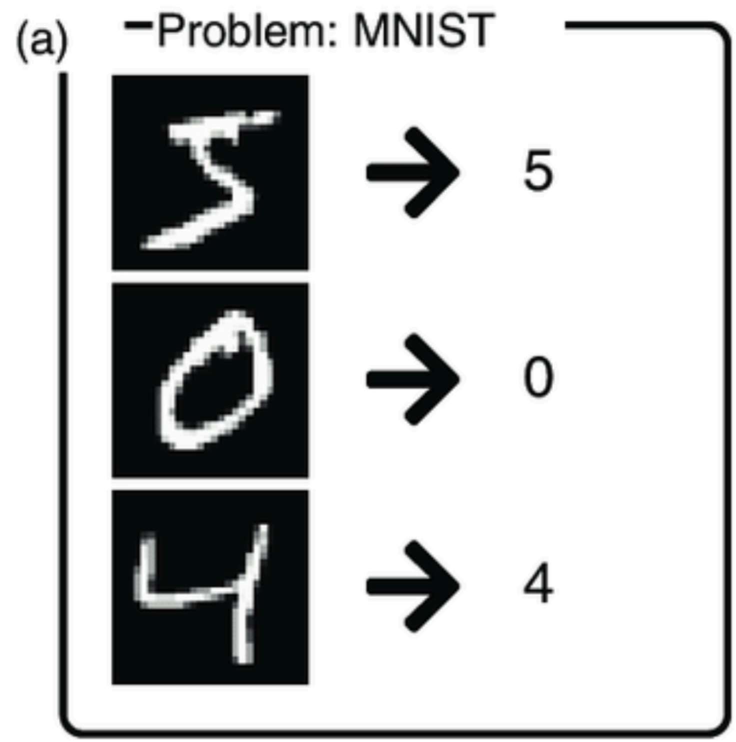
Good 😊 Bad 😑 Ugly 😱

Example: MNIST Numbers



Applied Physics Reviews 8:011310 (2011)

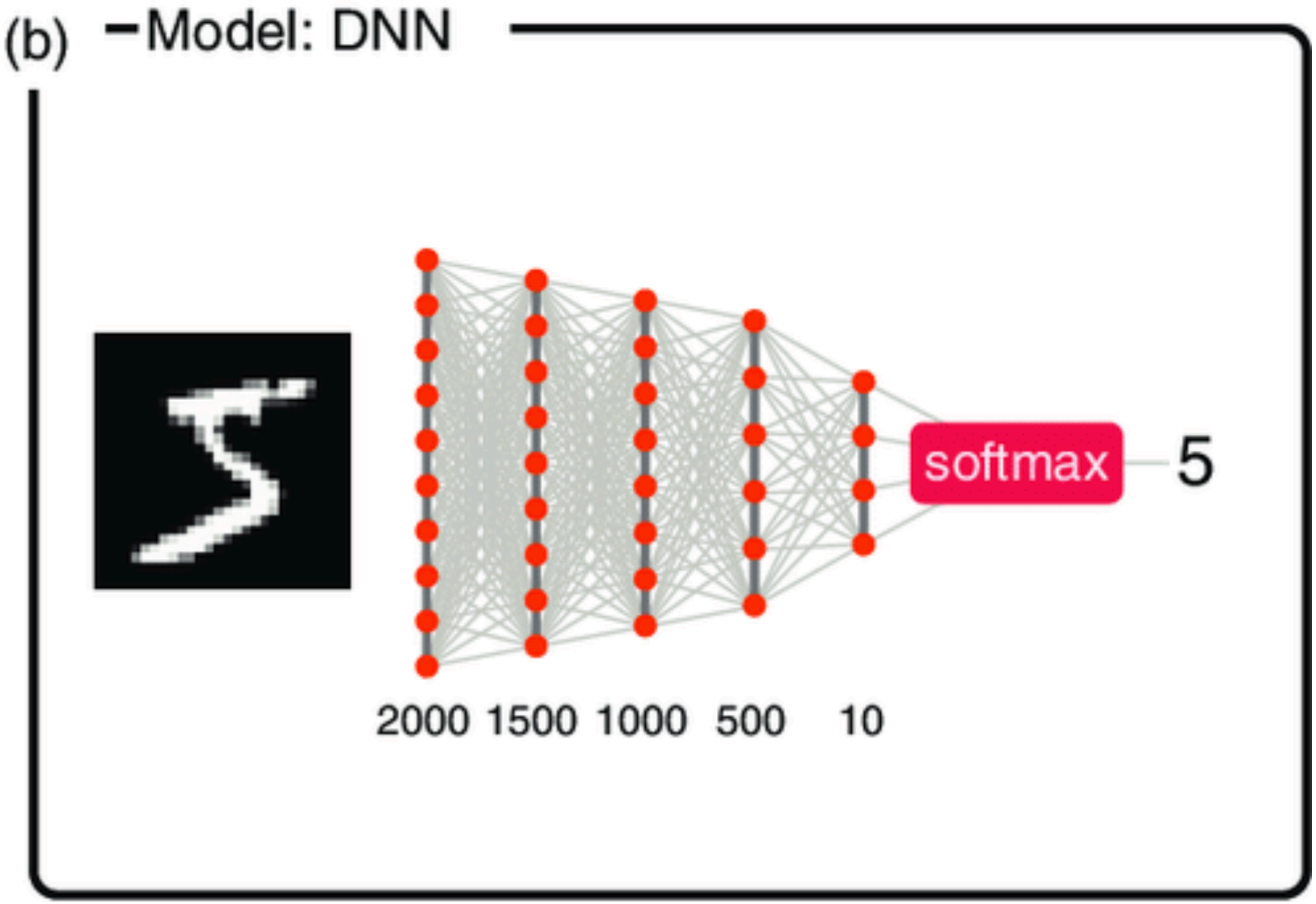
Example: MNIST Numbers



Classification

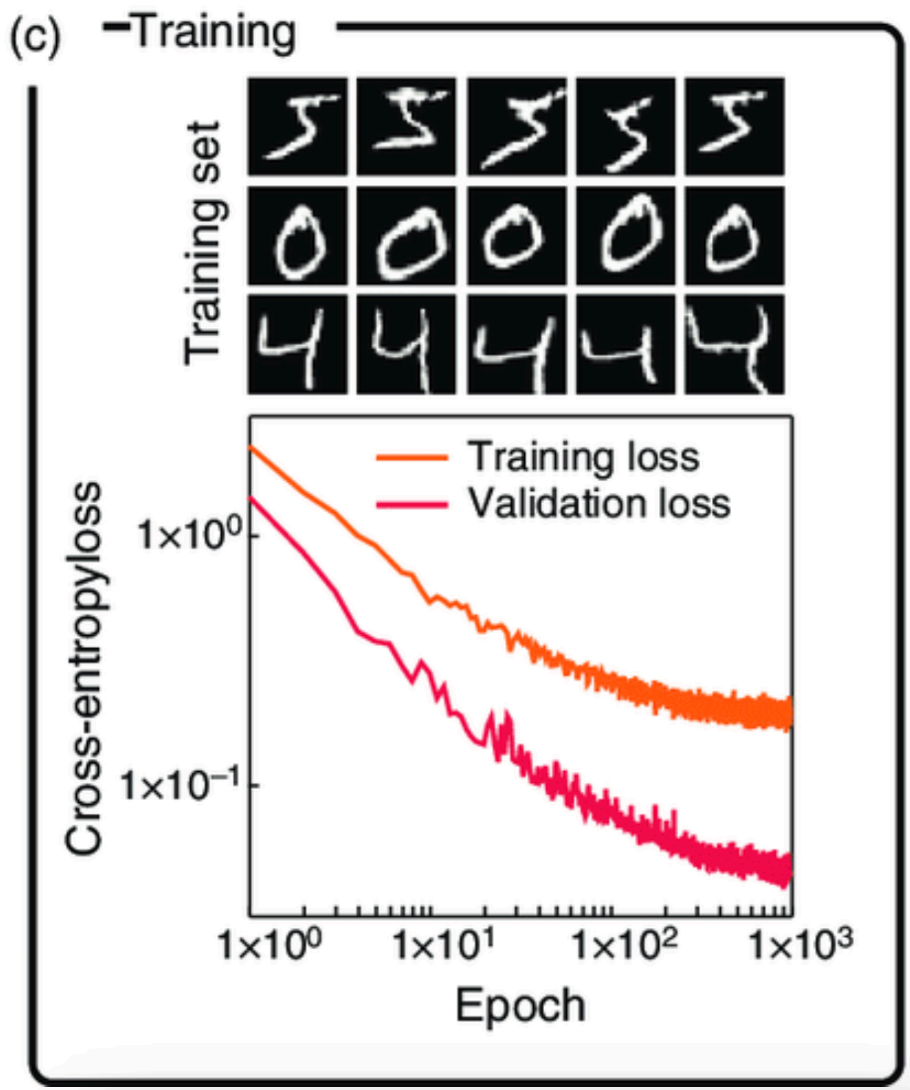
Applied Physics Reviews 8:011310 (2011)

Example: MNIST Numbers



Applied Physics Reviews 8:011310 (2011)

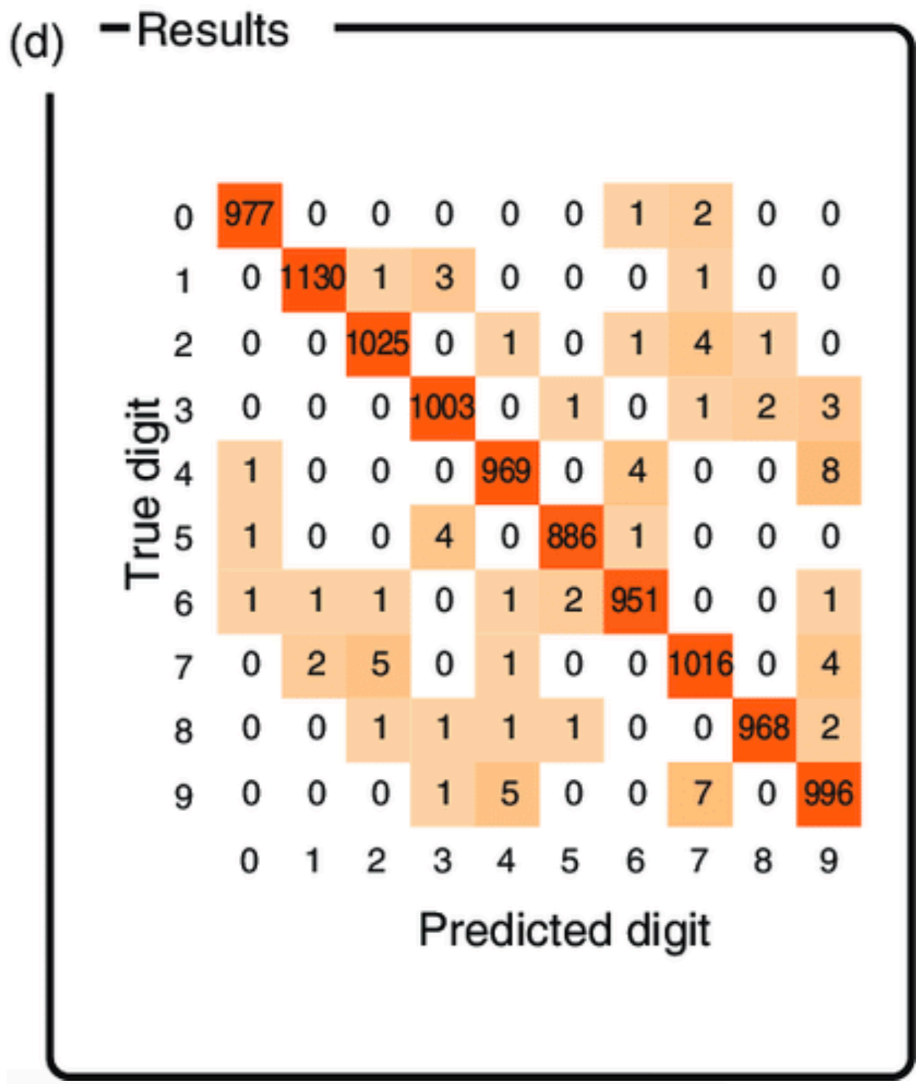
Example: MNIST Numbers



Loss should decrease as epochs increase → model is learning

Applied Physics Reviews 8:011310 (2011)

Example: MNIST Numbers



Confusion matrix: model is very accurate!

Applied Physics Reviews 8:011310 (2011)

Recap

Recap

- ML at the most basic level is a high-dimensional *non-linear* fitting procedure
  - Described by minimizing **loss function** using **gradient descent**

- Building an ML model requires training/test data, a choice of input modeling, and decisions on model architecture & hyperparameters

- ML performance can be assessed by studying the loss vs. epoch and the ROC/AUC (along with many other diagnostics)

- Developing ML methods on your own is facilitated by many preexisting python packages!

Conclusions

- **Why AI/ML?**
  - Exciting and rapidly growing technology that will change our world in ways yet to be seen (and physics research is no different!)

- What we discussed so far:
  - Motivation for AI/ML in physics
  - Basic math of simple neural nets
  - Optimizing & qualifying models

- *Coming up next*
  - Applications to physics research fields
  - Advanced models: anomaly detection, graphs
  - Real-time ML and hardware systems

Backup