6th Summer School on INtelligent signal processing for FrontIEr Research and Industry

30th August 2021, University Autónoma de Madrid

INFIERI

Introduction to Machine Learning and Deep Learning (Part II)

Juan Carlos San Miguel

Associate Professor at UAM & Researcher at VPULab &
Director of Master in Deep Learning for Audio and Video Signal Processing

Juancarlos.sanmiguel@uam.es

UAM Universidad Autónoma de Madrid

Escuela Politécnica Superior

VPU Video Processing and Understanding Lab

OUTLINE

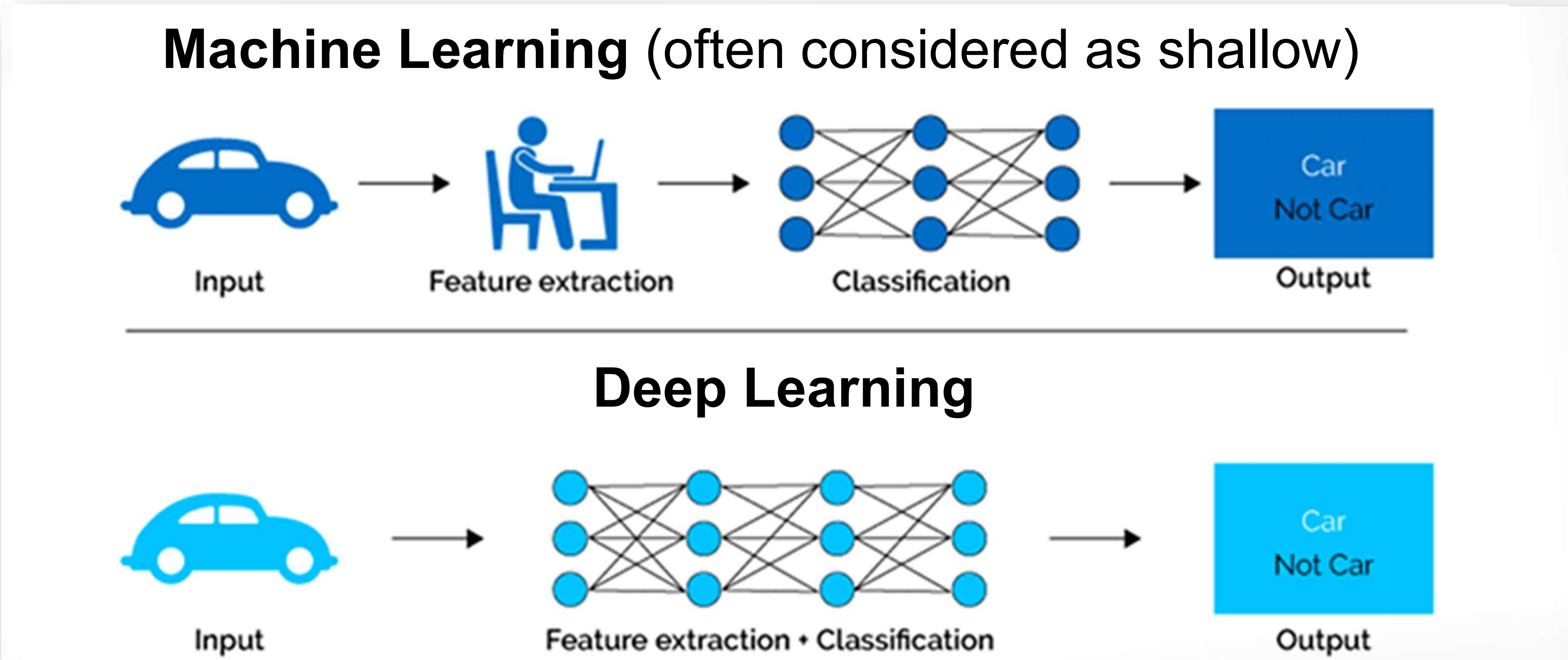
- What is Deep Learning?
- Key elements in Deep Learning
- Examples of Deep Learning algorithms
- Some applications in Physics
- Conclusions

WHAT IS DEEP LEARNING?

Machine Learning vs Deep Learning

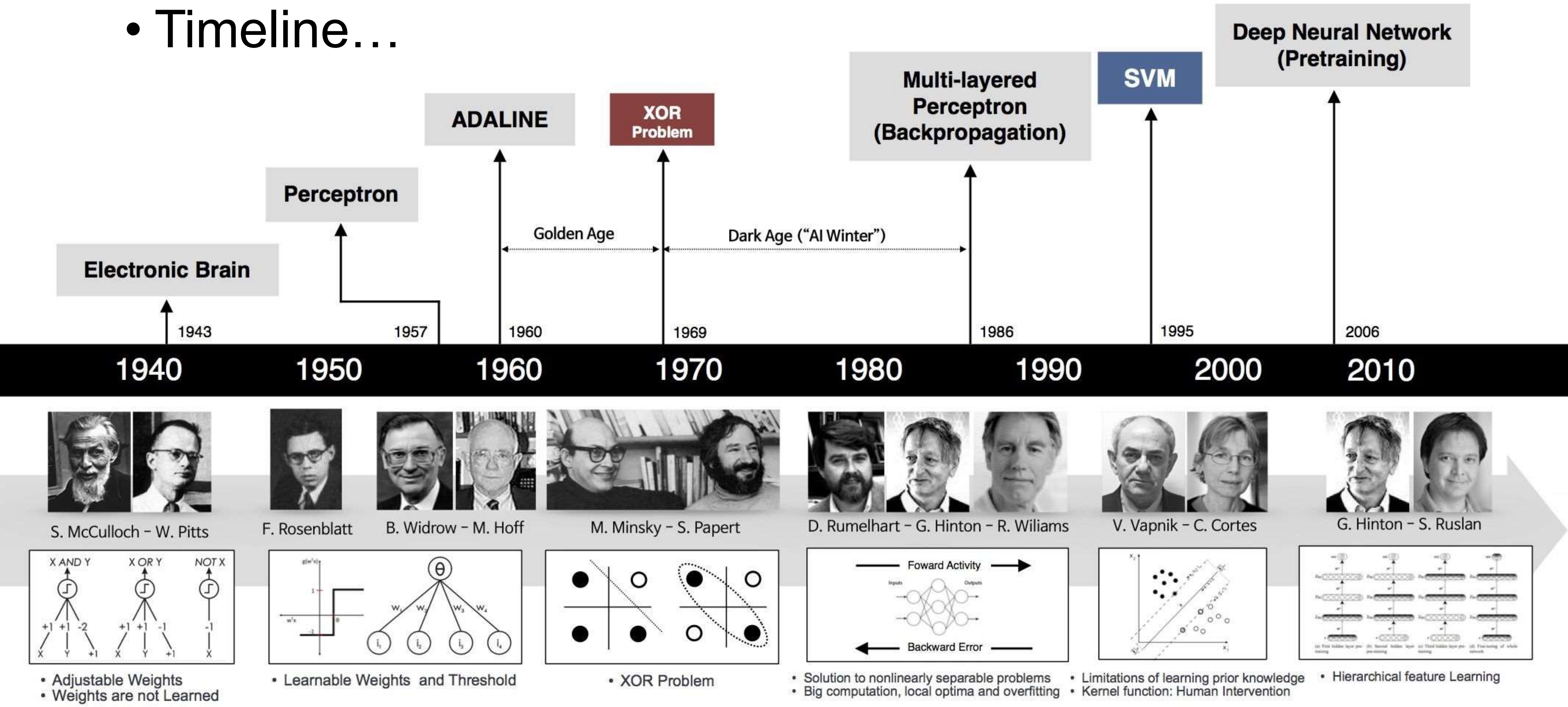
- Focus on learning the task
- Can be applied to small datasets
- Few layers/processing stages

- Focus on joint feature-task learning
- Requires large datasets
- Several layers/processing stages



Credit: https://blog.thinkwik.com

- Timeline…

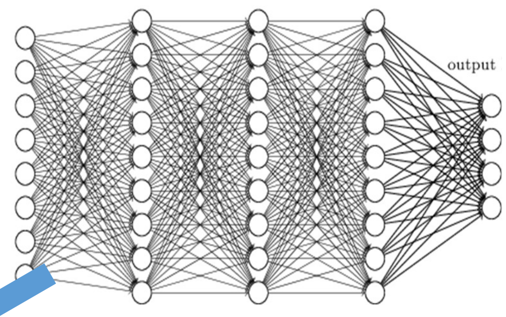


http://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

Why Deep Learning has become so popular nowadays?

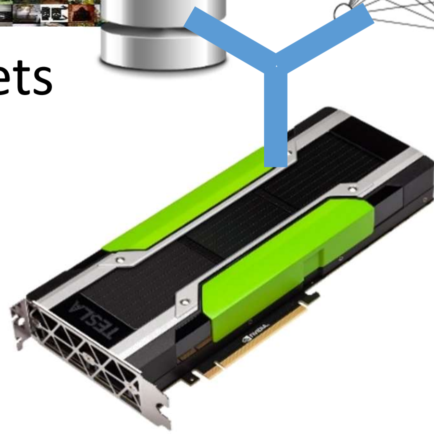
- Why now?

Large datasets
(Big Data)



Deep learning
frameworks

TensorFlow, PYTORCH, R, mxnet, Matlab, Keras, APACHE Spark, Microsoft CNTK



GPU technology

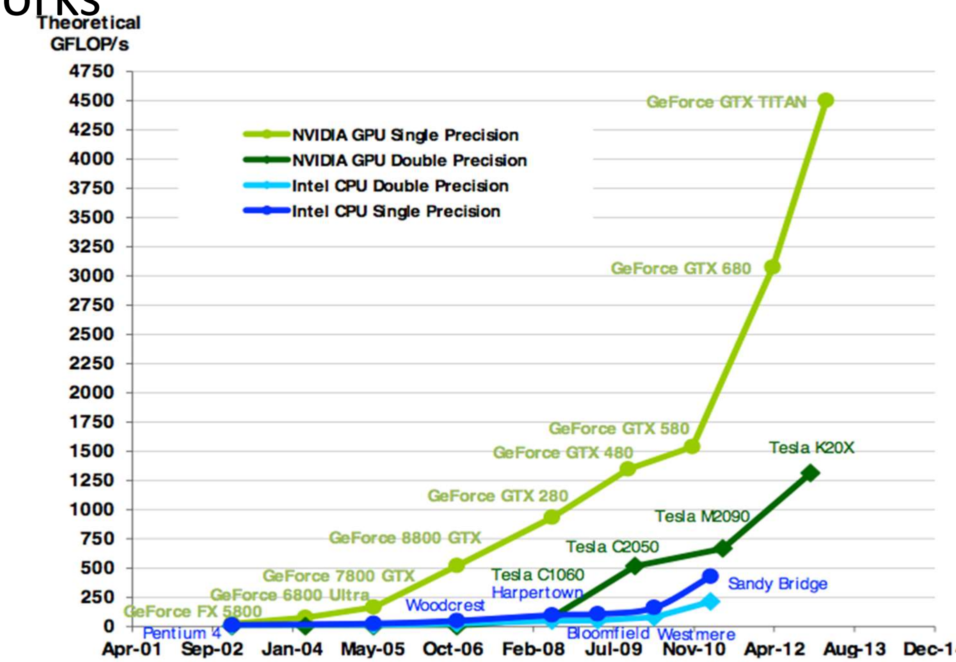
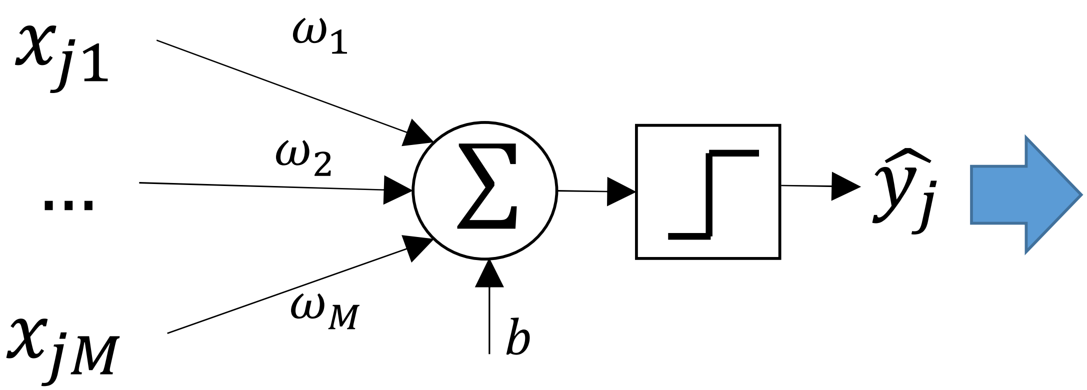


Figure 1 Floating-Point Operations per Second for the CPU and GPU

Credit graph: https://www.sciencedirect.com

DEEP LEARNING – KEY ELEMENTS

- Algorithms are based on Neural Networks[1]
  - Most basic Neural Network: Perceptron
    - Input data instance $x_j = (x_{j1} \dots x_{jN})$
    - Output unit/prediction $\hat{y}_j$
    - Parameters (to be learned) θ : weights ω_i and bias b
    - Activation function $f(z)$



$$\hat{y}_j = f\left(b + \sum_{i=1}^{M} \omega_i x_{ji}\right)$$

[1] M. Hagan, H. Demuth, & M. Beale. *Neural network design*. PWS Publishing Co.. 1997

- Optimization or loss function $L(x_j, y_j; \theta)$
  - Evaluates the error with current parameters values θ
  - Requires annotated data for its computation
  - Employed to find the optimal parameters

Number of processed data instances

$$\theta^* = argmin_\theta \sum_{j=1}^{N} L(x_j, y_j; \theta)$$

  - Examples:

Cross-entropy error (classification)

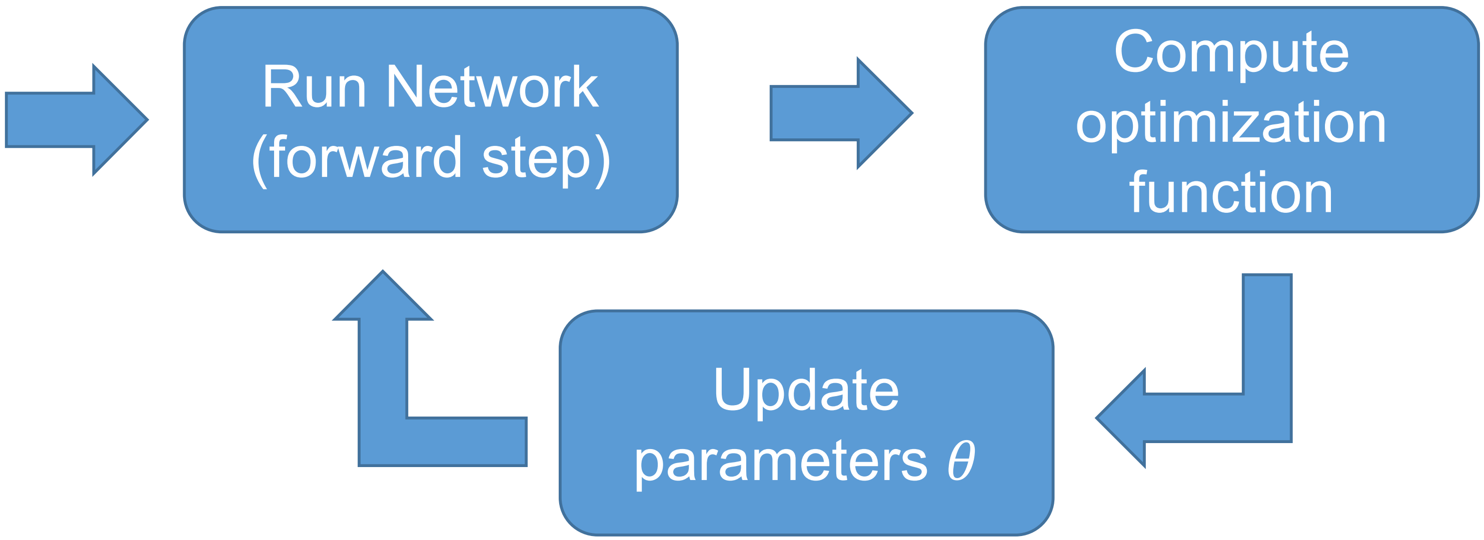
$$L_{CE}(\theta) = -\frac{1}{N}\sum_{j=1}^{N} \hat{y}_j \log(y_j) + (1-\hat{y}_j)\log(1-y_j)$$

Mean least-square error (regression)

$$L_{MSE}(\theta) = \frac{1}{N}\sum_{j=1}^{N} (\hat{y}_j - y_j)^2$$

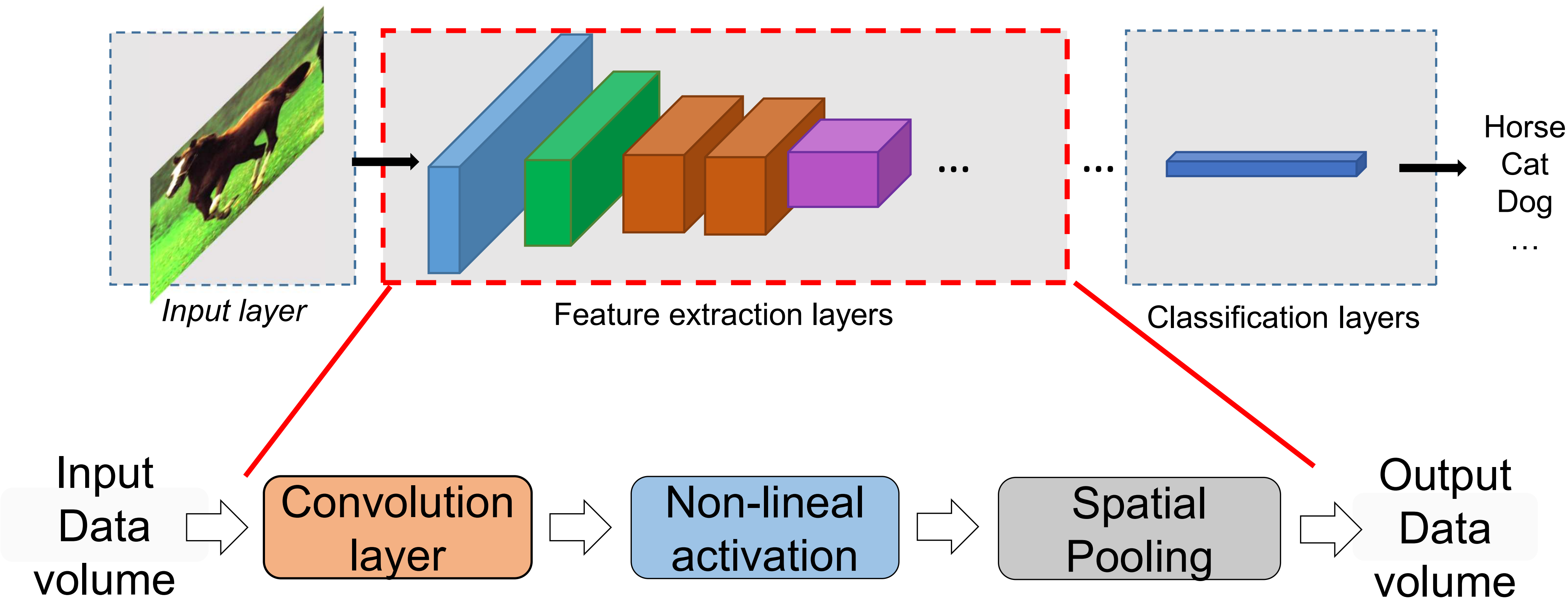
$\hat{y}_j$ is the network prediction for each data instance jth and y_j is the associated *ground-truth*

- Iterative learning
  - Allows to get the optimal value for parameters θ



  - This scheme is applied to batches of N instances of the dataset
  - “Update parameters” is often called optimization strategy:
    - Mostly based on backpropagation (backward step) to quantify dependency of parameters with the network output
    - Alternatives: Stochastic Gradient Descent, RMSprop, Adam,…

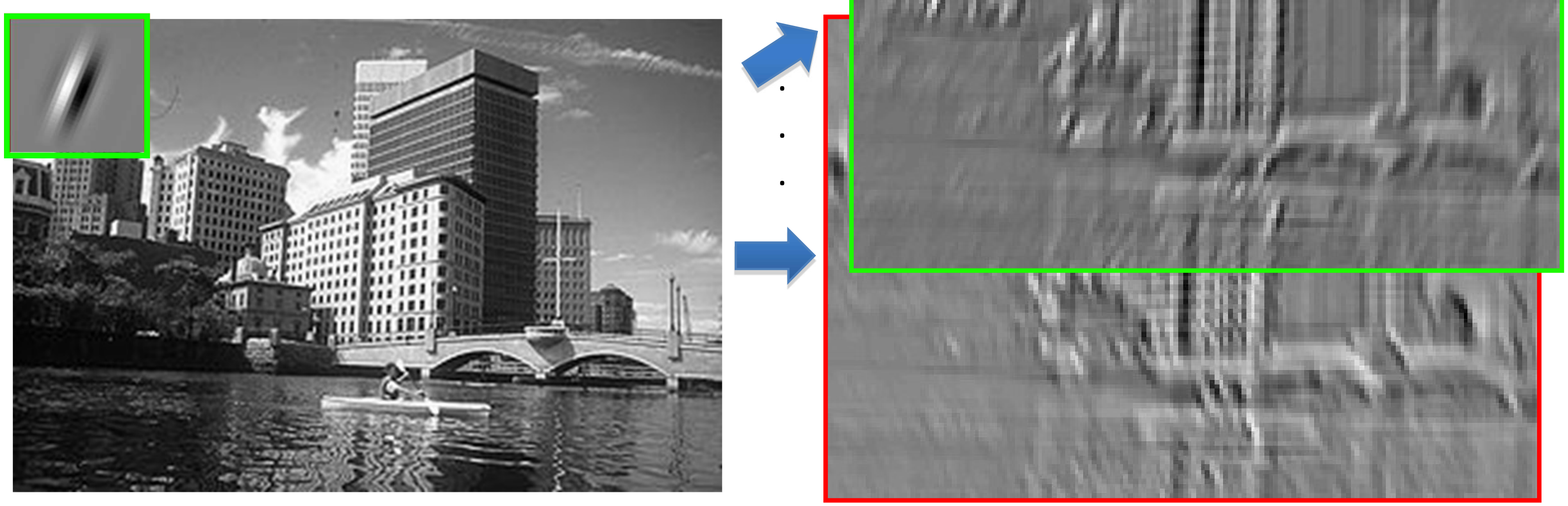
- Convolutional Neural Networks[1]
  - Neural Networks designed for classification of 2D data (e.g. images)
  - Sequential composition of various types of layers (processing stages) often conceptually organized into feature extraction and classification



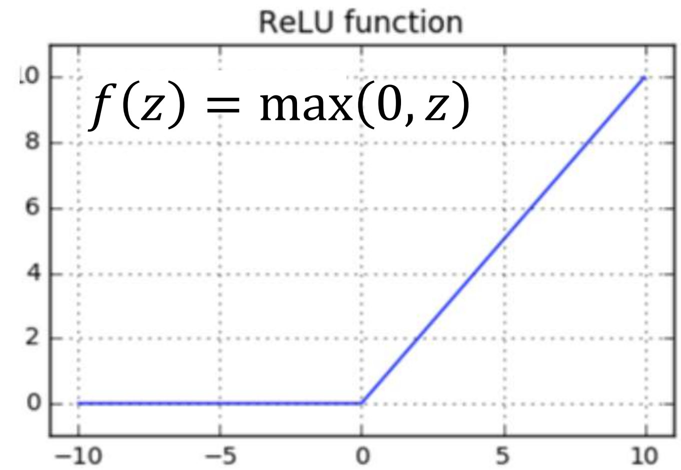
[1]K. O'Shea, & R. Nash. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

- Convolutional Neural Networks: Convolutional layer
  - Determines the features that can be extracted from a 2D signal x_i
  - Defined by multiple 2D kernels f_r of size $M \times M$ and the kernel values are the parameters to be learned during training
  - Output values $o_{ir}[m,n]$ are obtained by applying each kernel f_r

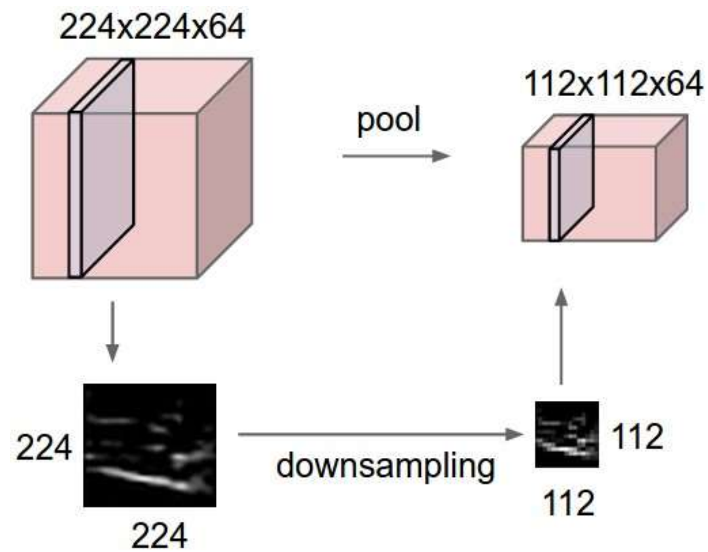
$$o_{ir}[m,n] = \sum_{k=1}^{M}\sum_{l=1}^{M} f\,[k,l]\,x_i[m+k,n+l]$$



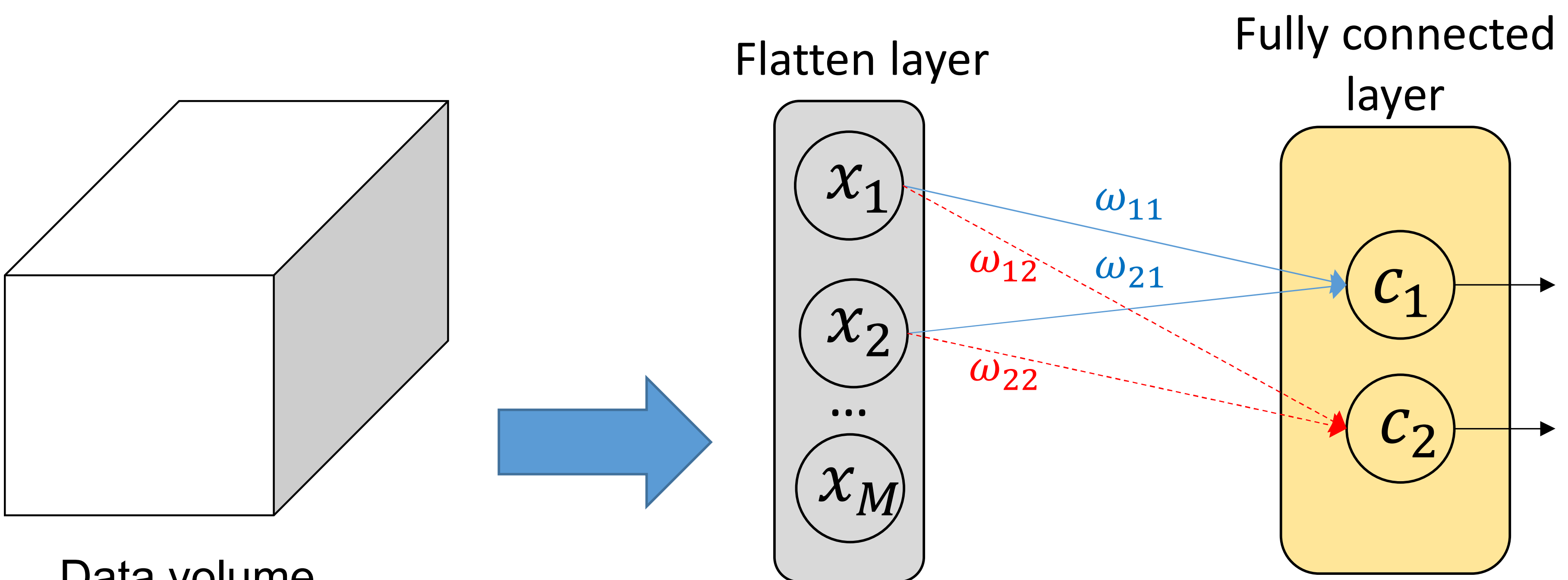
- Convolutional Neural Networks: other layers
  - Non-linear activation
    - Allows to solve non-linear problems
    - Reduces convergence time
    - Keeps bounded the processed data
    - Many alternatives available: sigmoid, tanh, ReLU, Leaky ReLU,….



  - Spatial Pooling
    - Reduces data dimensionality (only spatial dimensions)
    - Adds spatial independency to the location of extracted features
    - Decreases the number of parameters for subsequent layers in the network



- Convolutional Neural Networks: classification
  - Fully-connected layer (FC)
    - Classification composed by one or multiple sequential FC layers

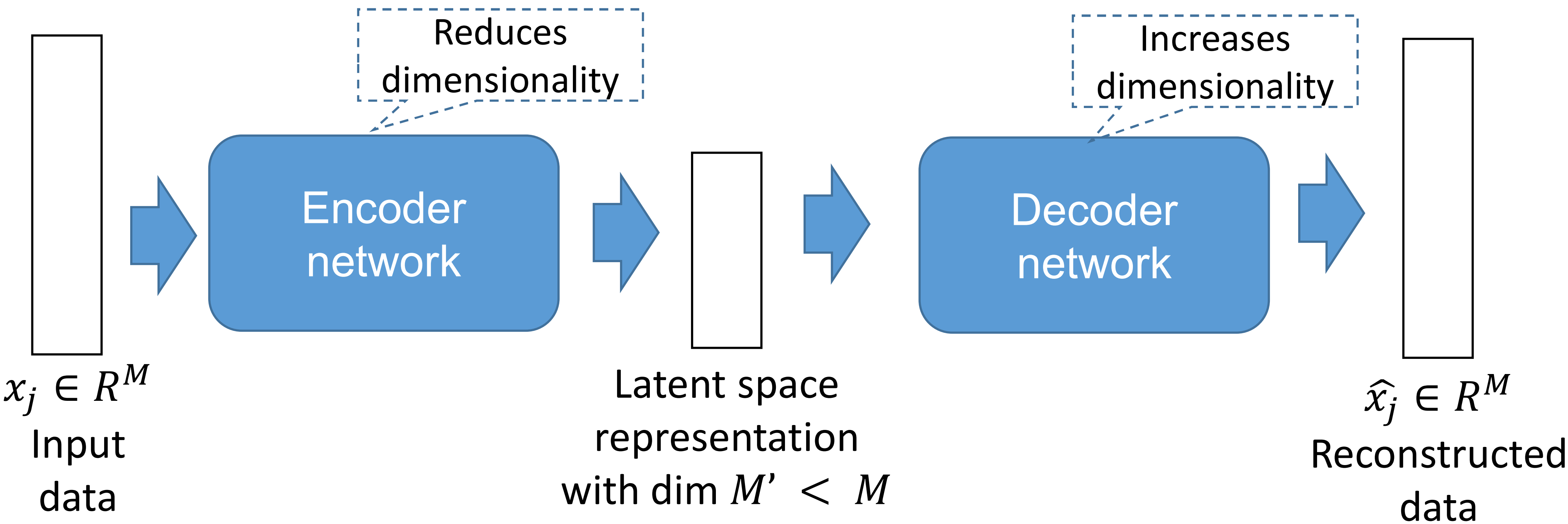


Data volume $W_1 \times H_1 \times D_1$ from feature extraction layers

Unidimensional data volume $1 \times M$ ($M = W_1 H_1 D_1$)

$$c_j = f\left(b_j + \sum_{i=1}^{M} \omega_{ij} x_i\right)$$

- Autoencoders[1]:
  - Unsupervised learning
  - Allows to learn a low-dimensional space representing input data x_j
  - Optimization function $L_{MSE}(\theta) = \frac{1}{N}\sum_{j=1}^{N}\left(\hat{x}_j - x_j\right)^2$ - No ground-truth



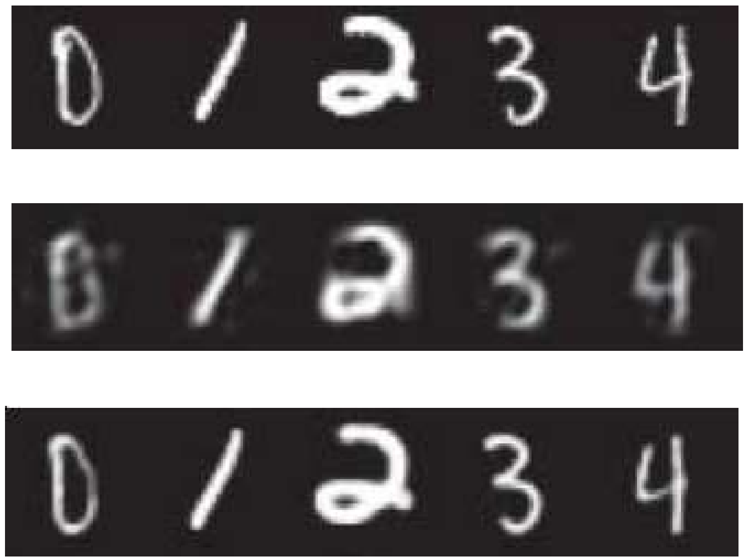
[1]G. Hinton, and R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313 (5786):504-7, 2006

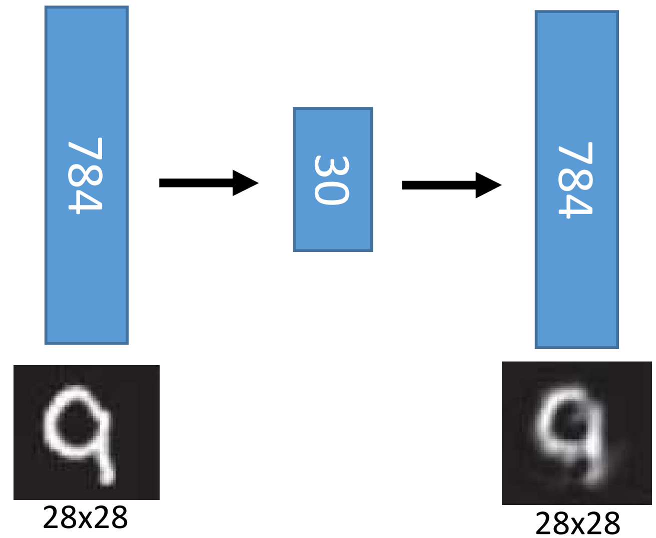
• Autoencoders - Example with only FC layers

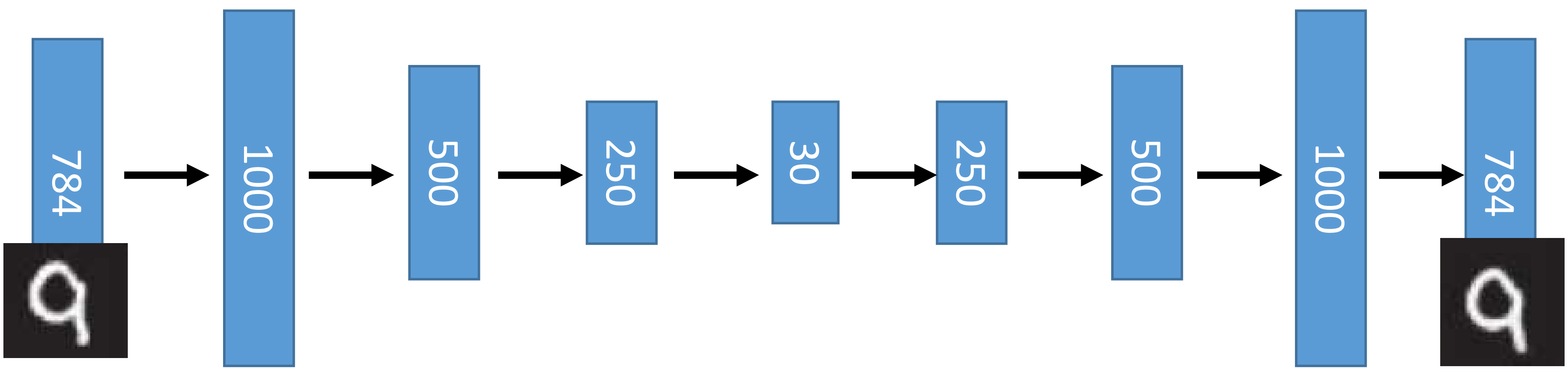
Original Image

PCA (ML)

Deep Auto-encoder







Slide credit Hung-yi Lee

- Recurrent Neural Networks (RNNs)[1]
  - Model temporal evolution of sequential data problems X_t
  - Has in-built “memory” (matrix A)
  - Defined by non-linear activations (functions $f(\cdot)$ and $g(\cdot)$) and linear operations (matrices $\mathcal{U}, \mathcal{V}, \mathcal{W}$). To obtain the output a time t

$$\left.\begin{array}{c} A_{i+1} = g(\mathcal{W} \cdot A_i + \mathcal{U} \cdot X_{i+1}) \\ h_{i+1} = f(\mathcal{V} \cdot A_i) \end{array}\right\} for\ every\ i = 1 \ldots . t$$

RNN architecture (folded version)

RNN architecture (unfolded version)

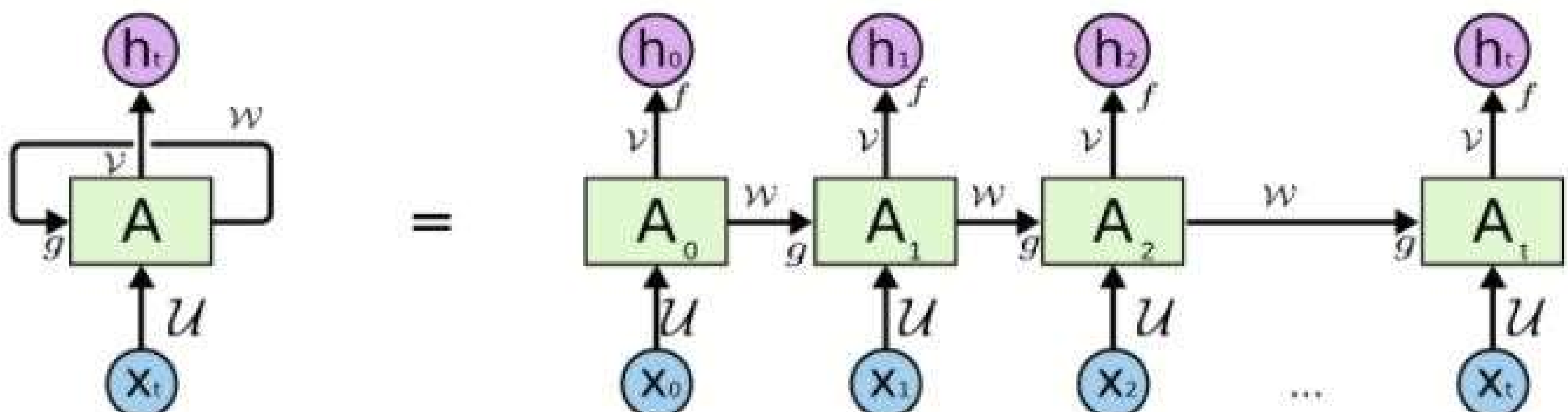
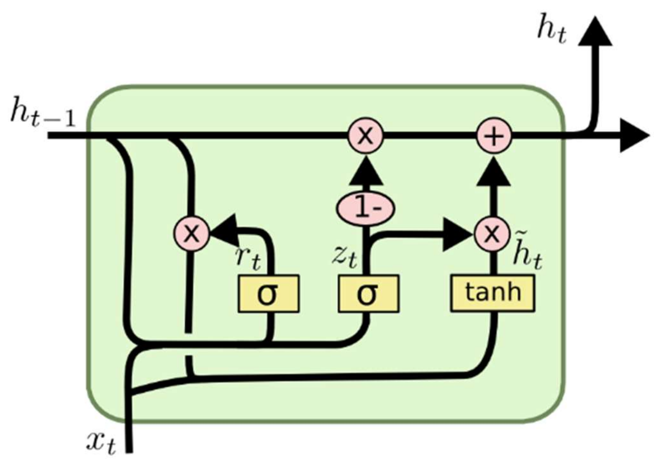


Image credit Christopher Olah

- Recurrent Neural Networks - Alternative approaches
  - A major RNN drawback is that time-dependency dilutes over timesteps (vanishing gradient) so RNNs are improved by gating
  - Gating adds difficulty to training as compared to vanilla RNNs
  - Gated Recurrent Unit (GRUs)[1]
    - Update gate z_t: how much of previous memory/result h_{t-1} to keep
    - Reset gate r_t: how much of previous memory/result h_{t-1} to forget



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

[1] K. Cho,et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". arXiv:1406.1078.

- Generative Adversarial Networks
  - Combination of two independent networks
    - Generator: obtains synthetic data (i.e. fake generator) similar to real data
    - Discriminator: given some input, determine if it is real or fake

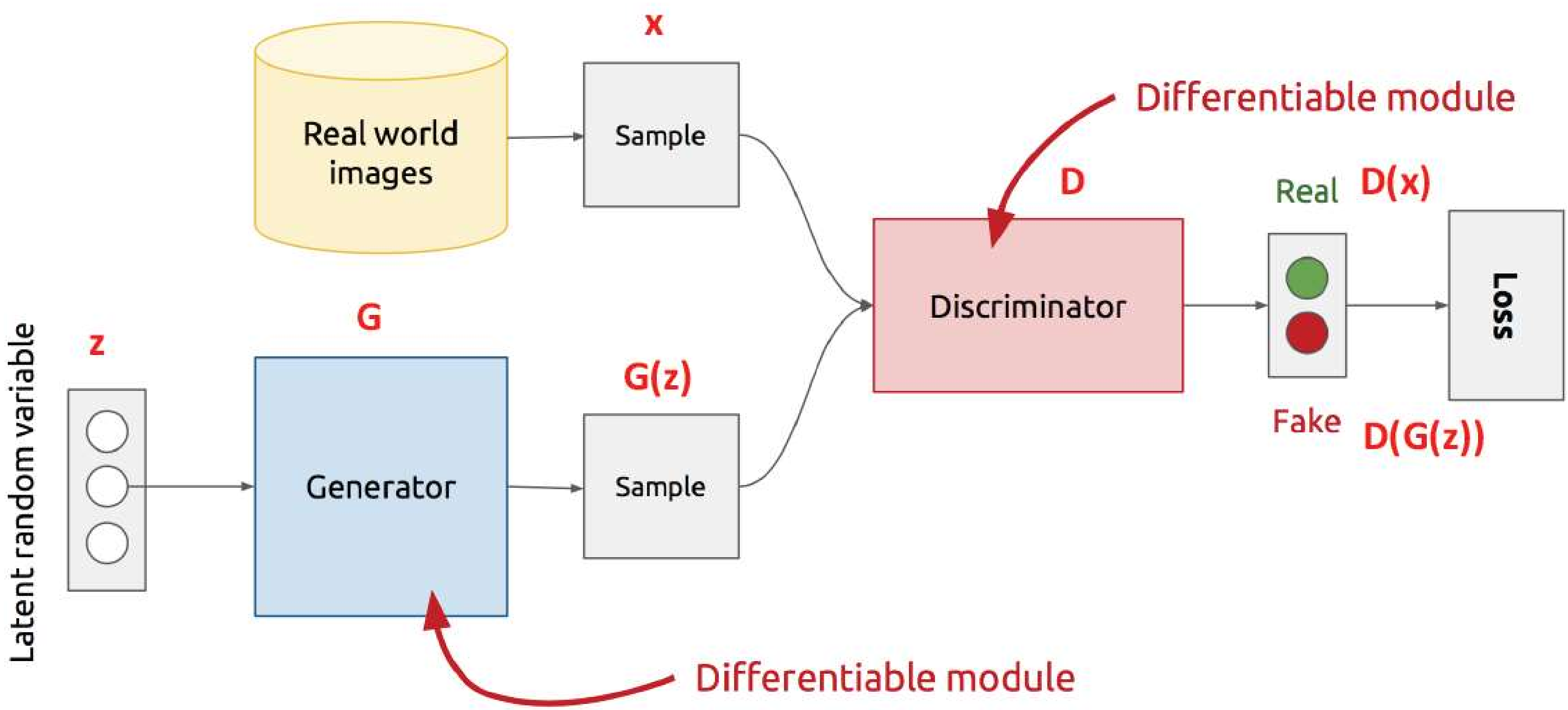
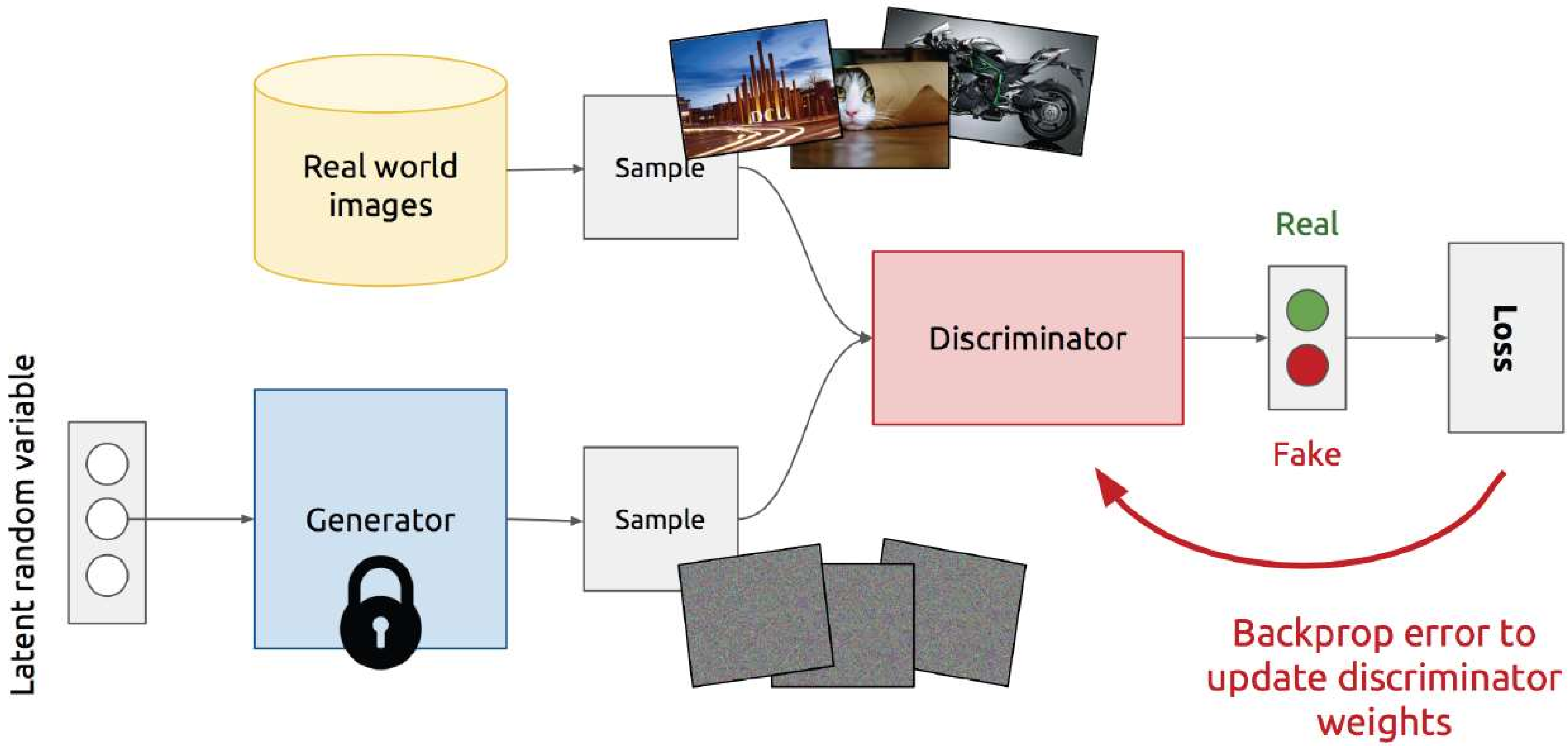
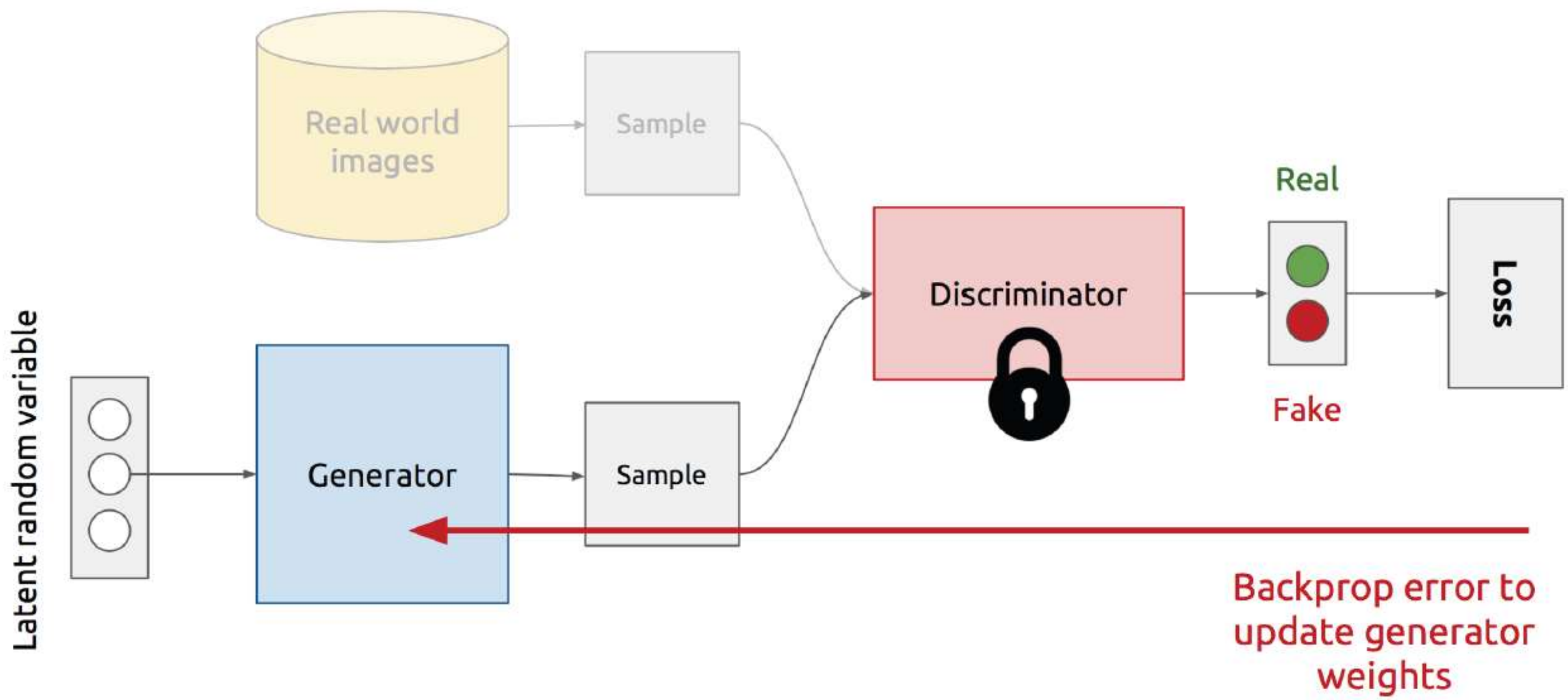


Image credit: https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

- Generative Adversarial Networks
  - Training takes place at three stages
    1. Run the network for real and fake images
    2. Then, freeze generator and update discriminator



- Generative Adversarial Networks
  - Training takes place at three stages
    3. Finally, Freeze discriminator and update generator

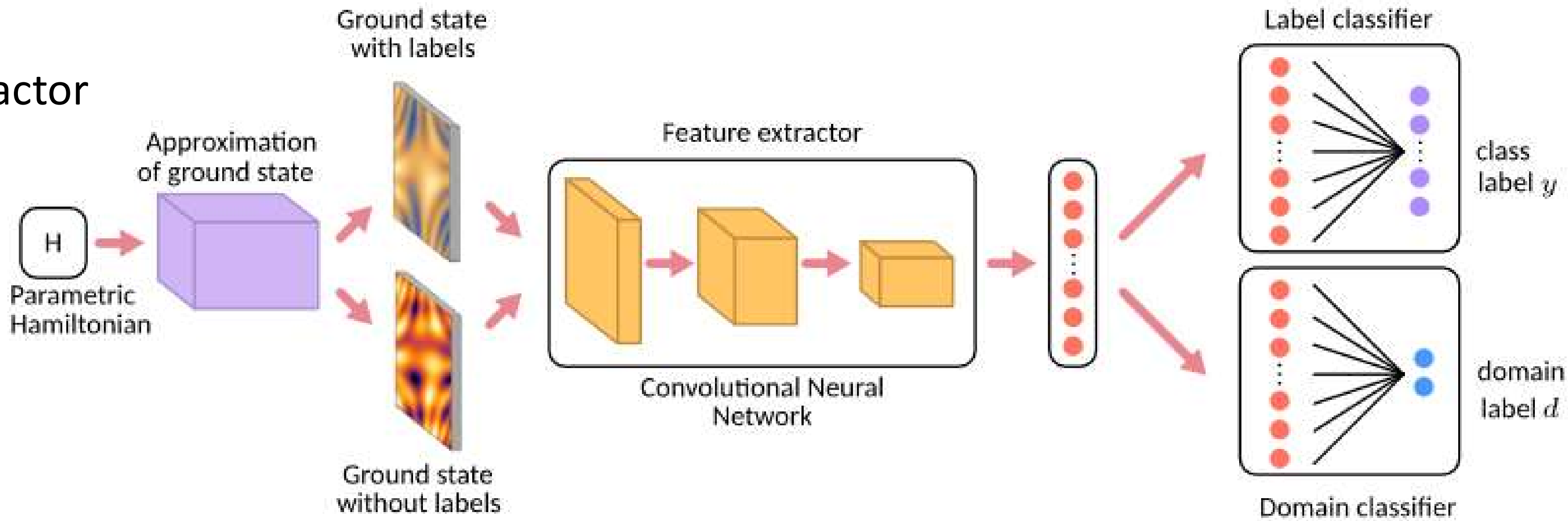


Optimization is formulated as a minimax game
- Discriminator tries to maximize its reward $\boldsymbol{V}(\boldsymbol{D}, \boldsymbol{G})$
- Generator tries to minimize Discriminator's reward

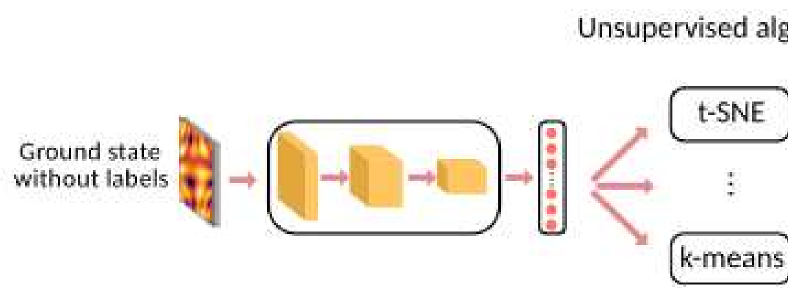
$$\min_G \max_D V(D, G)$$

- Identifying phases of matter in quantum mechanics

1) Learn the feature extractor



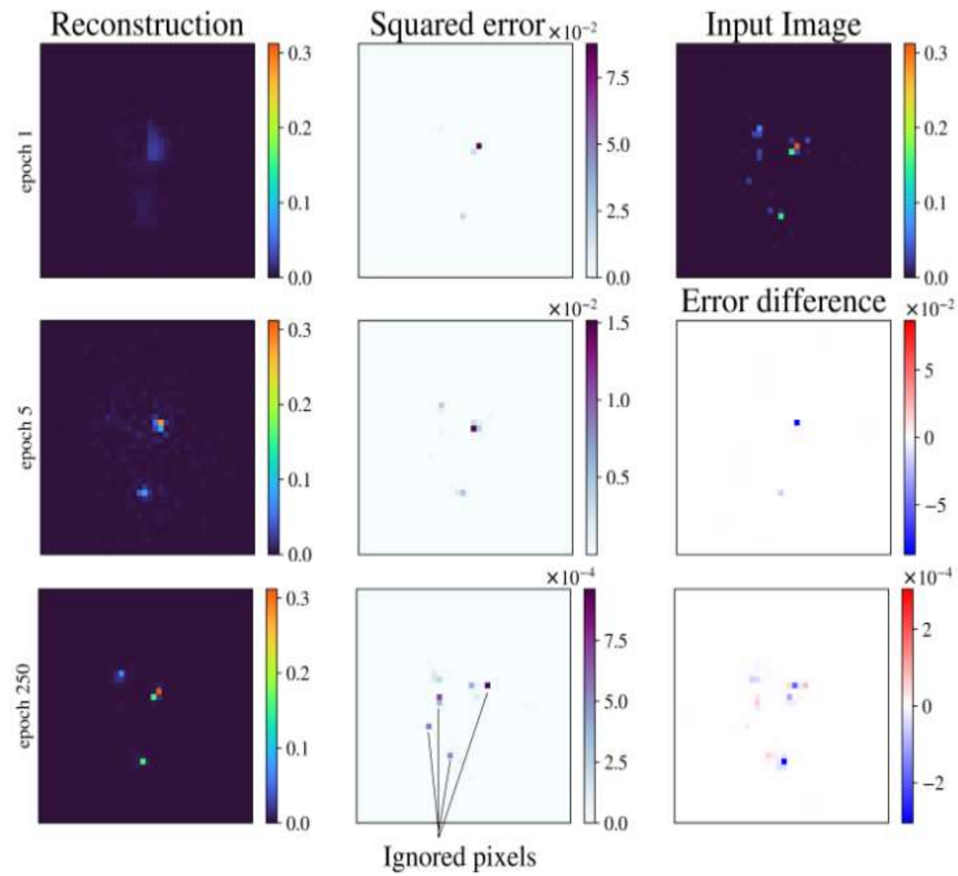
2) Employ the trained feature extractor to categorize phase transitions



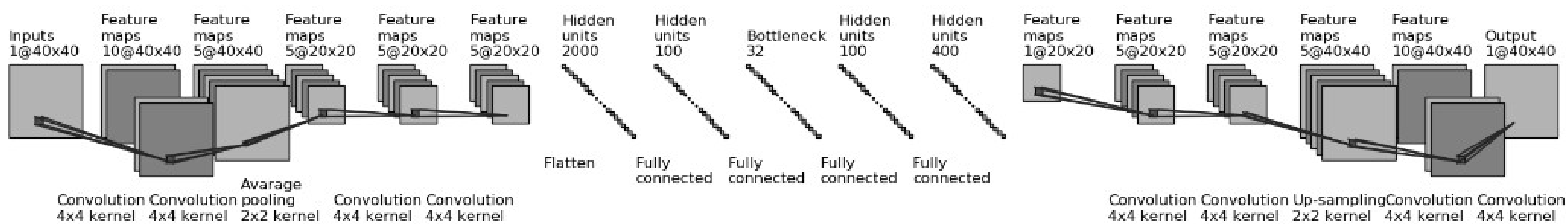
Huembeli, P., Dauphin, A., & Wittek, P. (2018). Identifying quantum phase transitions with adversarial neural networks. *Physical Review B*, *97*(13), 134109.

DEEP LEARNING APPLICATIONS

- Single collider events in LHC data cannot be labeled as signal or background due to the probabilistic nature of quantum mechanics
- Unsupervised learning is applied for tagging Top Jet and images as an anomaly detection approach
- Postprocessing is applied to boost tagging performance

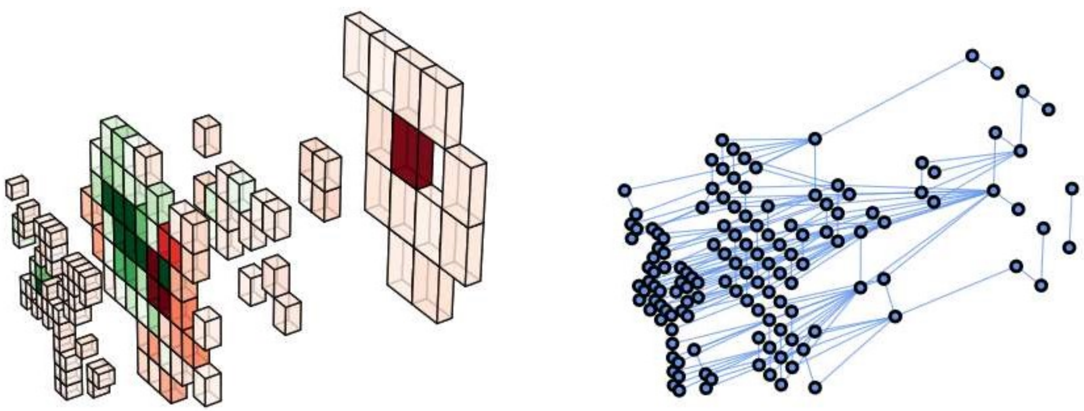


Finke, T., Krämer, M., Morandini, A. et al (2021). Autoencoders for unsupervised anomaly detection in high energy physics. *J. High Energ. Phys*. 2021,.

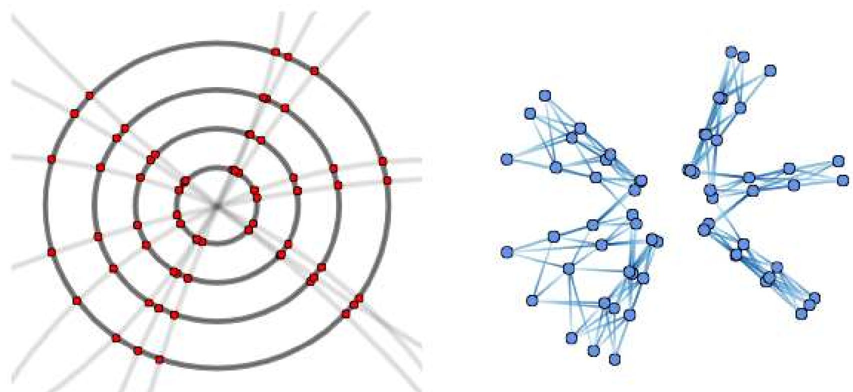


DEEP LEARNING APPLICATIONS

- Data in particle physics are often depicted by sets and graphs, so Graph Neural Networks (GNNs) are suitable tools here
- GNNs are trainable functions which operate on graphs, updating nodes' and edges' contents given some task



segmenting calorimeter cells



jet classification based on the particles associated to the jet.

Shlomi, J., Battaglia, P., & Vlimant, J. R. (2020). Graph neural networks in particle physics. *Machine Learning: Science and Technology*, *2*(2), 021001.

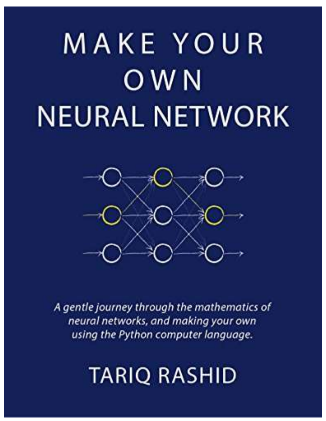
CONCLUSIONS

- Advantages of Deep Learning
  - Flexible structure that can be adapted to a plethora of problems
  - Can easily increase complexity by adding more layers
  - Huge open-source community with state-of-the-art algorithms
- Disadvantages of Deep Learning
  - Complex theorical analysis (sometimes is not even possible) that prevents from having a closed formulation for the neural network
  - High sensibility to local minima so multiple runs are needed
  - Requires a large quantity of data and high computational resources
  - Many design options make difficult to optimize hyperparameters and network structure
- Deep Learning is often applied on high level features derived from physics data. Improvements are expected when operating on lower-level information.

WANT TO LEARN MORE?

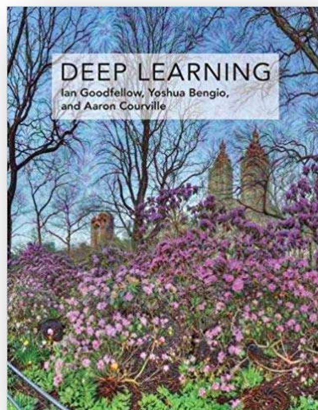
- Due to the DL novelty, research papers are the main source of knowledge but some books cover the fundamentals…

Beginner



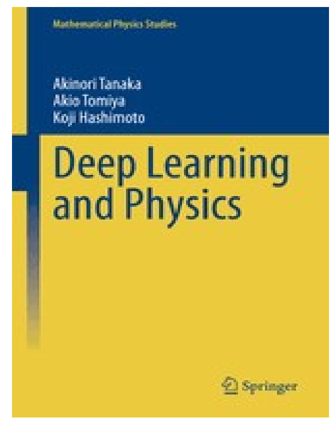
2nd Ed 2021
https://amzn.to/2TUhHXW

Intermediate-Expert



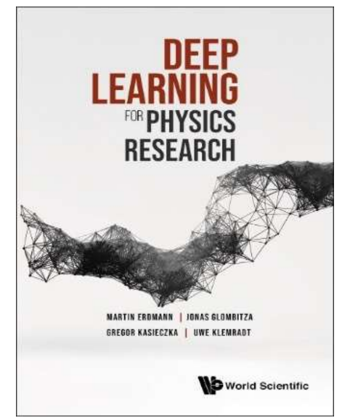
I. Goodfellow et al, "Deep Learning", MIT Press, 2016
http://www.deeplearningbook.org/

Expert



1st Ed 2021
https://bit.ly/3sN2hBH

Expert



1st 2021
https://amzn.to/3yiSp3G

- As for practical work, check popular DL frameworks (TensorFlow, PyTorch, MXNet, CNTK,…)

6th Summer School on INtelligent signal processing for FrontIEr Research and Industry

30th August 2021, University Autónoma de Madrid

Introduction to Machine Learning and Deep Learning (Part II)

Juan Carlos San Miguel

Associate Professor at UAM & Researcher at VPULab &
Director of Master in Deep Learning for Audio and Video Signal Processing

Juancarlos.sanmiguel@uam.es

ANY QUESTIONS?