

Deep Learning et Réseaux de Neurones

Marie Beurton-Aimar, LE Van Linh

September 23, 2026

Deep Learning Models

What is Deep Learning?

- Deep learning is a branch of machine learning that uses multi-layered artificial neural networks to model complex data.

Deep Learning Models

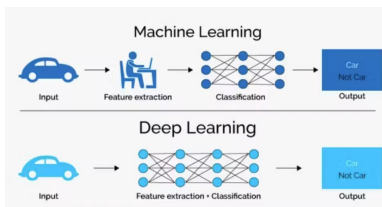
What is Deep Learning?

- Deep learning is a branch of machine learning that uses multi-layered artificial neural networks to model complex data.
- A subset of machine learning that uses large multi-layer neural networks to automatically learn complex patterns from data.

Deep Learning Models

What is Deep Learning?

- Deep learning is a branch of machine learning that uses multi-layered artificial neural networks to model complex data.
- A subset of machine learning that uses large multi-layer neural networks to automatically learn complex patterns from data.
- These models discover increasingly abstract representations on their own.



Deep Learning Models

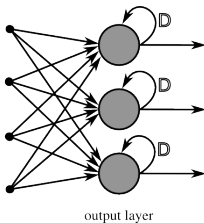
From LeCun

- Deep learning discovers intricate structure in large data sets by using the **backpropagation algorithm** to indicate how a machine should change its **internal parameters** that are used to compute the representation in each layer from the representation in the previous layer.
- Deep Learning is constructing networks of **parameterized functional modules** and **training them from examples using gradient-based optimization**.

Deep Learning Models

Different types of Models

- **Convolutional Neural Networks:** image processing, backbone of a lot of architecture.
- **Recurrent Neural Networks:** text analysis, translation, temporal series ...



Deep Learning Models

Different types of Models

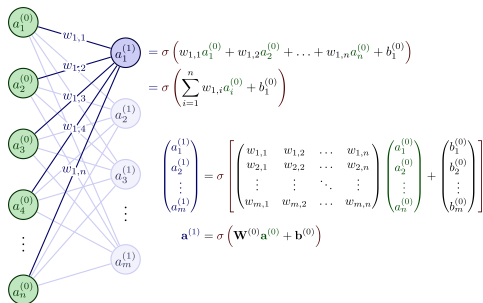
- **Long-Short Time Memory:** kind of recurrent network, keep in memory some past inputs.
- **Transformer and Attention module:** associated to Large Language Models (LLM), used in automatic language treatment (TAL in french). Better than LSTM but needs a lot of data to be trained.
- **Siamese Network:** to compare several inputs two by two. Most often CNN on each branch. Could also be used for multimodal applications.
- **Generative Adversarial Network:** trained to generate data with the same features than real one. Being the ancestor for generative IA. Used in video for example.



Back to the beginning: Multi-Layers Architecture

Neuron Unit

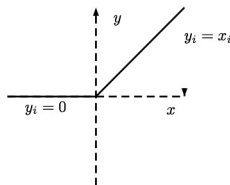
- A weight
- An activation function
- An output value



Activation Functions

Rectified Linear Unit

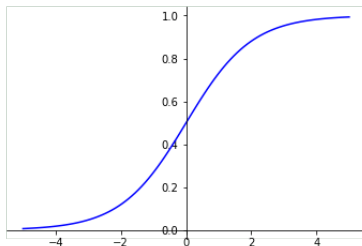
- The most used function in Deep Learning.
- Apply a filter to the layer output by keeping only positive value.
- Relu_Function = $\max(0, x)$ - Consequently the negative values are replaced by 0.
- The maximum value can be fixed by user, when output value are greater than the max they are replaced by it. This limits the output value range.



Activation Functions

Sigmoid

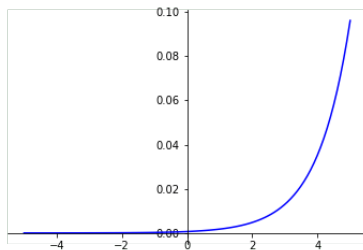
- Used by the last network layer to apply the binary classification task.
- Equation : $\text{Sigmoid_function}(x) = 1 / (1 + \exp(-x))$
- Can be view as the probability to belong to one class.
- In the case of a binary classification, the sum of all values is equal to 1.



Activation Functions

Softmax

- Also used to do a multi-class classification task.
- Can be used at the last layer level.
- Each output is between 0 and 1, the sum is equal to 1.
- $\text{Softmax_function}(x) = \exp(x) / \sum(\exp(x_i))$



Activation Functions

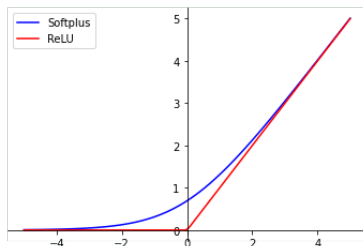
Softmax

- Results can be considered as probability to belong to a class.
- Each value is associated to a class.
- Sigmoid can't be used for the last layer into multi-class classification because in that case the sum is not equal to 1. Softmax has to be used in that case.

Activation Functions

Softplus

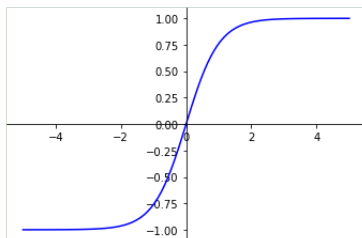
- Soft approximation of ReLU function.
- If the input values are positive, the behavior is the same than ReLU.
- Difference with ReLU: no filter for negative value but values tend toward 0.
- $\text{Softplus_function}(x) = \log(\exp(x) + 1)$



Activation Functions

tanh

- Normalize the input values.
- Can replace Sigmoid function in binary classification task in the last layer.
- Range result values is -1 to 1, it can't be considered as a probability value .
- $\tanh_function(x) = ((\exp(x) - \exp(-x)) / (\exp(x) + \exp(-x)))$.

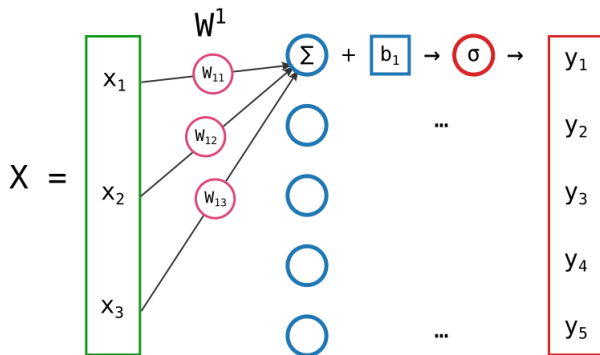


Activation Functions

Resume

- It exists a lot of another activation functions and more it is possible to design a new specific one for a purpose.
- Activation functions belong to the core of deep learning models.
- They apply non-linear transformations to data and allow to modify their spatial representation.
- Knowledge about their behavior is essential in deep learning design. Performances could be undermined by a wrong choice.

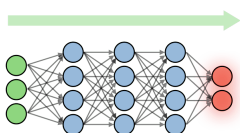
Forward Propagation



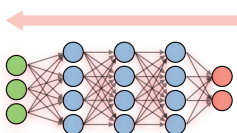
Learning Resume

3 steps

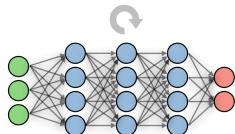
- 1 Take the training data and apply forward propagation
- 2 Back propagate the loss to obtain the loss gradient of each coefficient.
- 3 Use the gradient to update the network coefficients.



① Propagation avant



② Rétropropagation



③ Mise à jour des coefficients

Backpropagation

Goal

- Is a learning algorithm to improve the outputs.
- Correct the error by using the loss function.
- The loss function measures the error level at each epoch in order to tell to the next one performances of the current weight values.
- To do it, gradient descent is computed and weights are adjusted to run the next epoch.
- Gradient descent with momentum, with a memory of previous gradient values, can be used to accelerate algorithm convergence.

Backward Propagation

Loss Functions

- Goal : minimize the error rates between prediction and groundtruth.
- Depending on data type, different functions have to be chosen.
- The result is after analyzed by gradient descent algorithm.
- Several optimizations are/can be done.

Break !

Tensor ! What is it ?

- Data structure essential in Deep learning.
- Can be viewed as multi-dimensional matrix.
- In PyTorch they are used for everything: input values, weights and bias, output results.
- All current matrix operations are available for tensors.
- Take care about the exponential explosion with the increase of dimensions.

Loss Functions

Which type ?

- Regression function : to predict continuous results from independent variables.
- Classification function: to provide discrete labels corresponding to a class.

Methods

- Regression : mean quadratic error (L2) mean absolute error (MAE).
Idea : minimize the square error between predicted and target values.
- Classification : Cross-entropy, log

Loss Functions

Performance evaluation

- Mean Square Error : Allows to penalize the outlier values.
- Cross entropie/log loss : computing from the difference between the two distributions values.

$$\text{Equation : } L(y, f(x)) = -[y * \log(f(x)) + (1-y) * \log(1-f(x))]$$

where y is the binary value 0,1 of the label and $f(x)$ the probability to belong to the positive class.

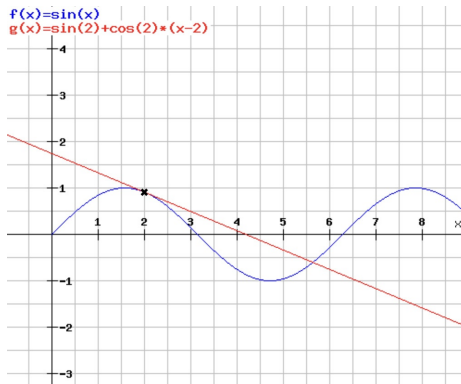
Penalize the wrong predictions - with a significative difference value from the positive class.



Gradient Descent

Methods

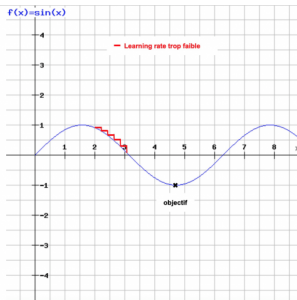
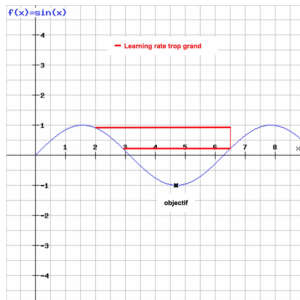
- Could be consider as the derivative function for tensors.
- Used to compute the evolution of $f(x)$.
- Chosing the slope direction.



Gradient Descent

Parameters

- Choose the learning rate : distance value to move on the slope.
 - if it is too small the time to compute will be enlarge and you can fall into a local minimum.
 - if it is too large the update can move you to an unreliable place on the slope



Gradient Descent

Parameters

- Choose the iteration number : how many times you move x on the slope.
 - if it is too small the descent will not have enough step to be realized and no optimisation.
 - if it is too large your computing time will be enlarge. In deep learning it is important to keep your time.

Gradient Descent

Backpropagation again !

- Applying the chain rule (https://en.wikipedia.org/wiki/Chain_rule):
- Gradient is computed for all weights and for all layers from the last to the first one.
- The goal is to compute the impact of each parameter and to modify it consequently.
- The loss is minimized through all iterations.

Network Performances

Methods

- Mean Absolute Error : mean sum of residual absolute values.
To be used to mesure regression results. Several versions of this error are available. Can be used to check over-fitting.
- Accuracy : $\text{Acc} = \text{number of correct prediction} / \text{number of inputs}$.
used for classification.
- Precision :
 $\text{Prec} = \text{number of True Positive} / (\text{number of True Positive} + \text{number of False Positive})$
- Recall :
 $\text{Rec} = \text{number of True Positive} / (\text{number of True Positive} + \text{number of False Negative})$
- F1-score : average between precision and recall
 $\text{F1} = (\text{Prec} * \text{Rec}) / (\text{Prec} + \text{Rec})$

Data Set

Choose the right sample

- Divide the data set into several parts :
 - Test set - exclude 10 % of the whole set and keep it aside.
 - Training set - most often 80% of the rest.
 - Validation set - 20 % that remain.

Usage

- Test set has to be never seen at the training step.
- At each epoch training set is entirely seen. Inputs can be load in several steps depending the batch size.
- At the end of the epoch, the validation set is seen to measure the current network configuration performances.
- When the model is fixed, considered as trained, the test set is load to verify the capability to generalize the treatment.

Convolutional Neural Network

AlexNet (from Alex Krizhevsky)

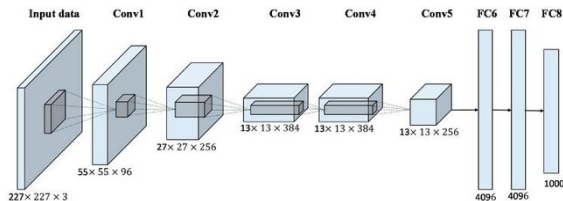
- To classify ImageNet items.
- Won the ILSVRC challenge with an error rate of 15.3 %



Convolutional Neural Network

Layer size

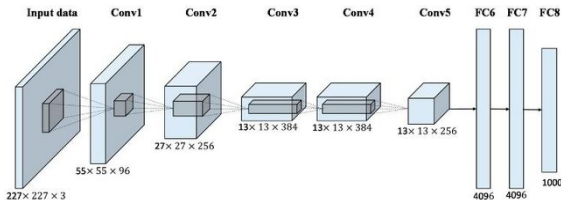
- How to adjust each layer size from input?



Convolutional Neural Network

Layer size

- How to adjust each layer size from input?
- Equation to respect : $(W - F + 2P)/S + 1$
where W is the input size, P is the zero padding and F the convolutional filter size. S is the value of stride.
- From the first layer to the second one : $(227 - 11)/4 + 1 = 55$
stride=4, no padding and filter = 11



Convolutional Neural Network

ReLU function

- The activation function suppressing the neagative value : : $F(x) = \max(0, x)$
- May be a problem and block the gradient backpropagation.
- Leaky Relu : $F(x) = \max(0.1x, x)$. Keep active all neurons.
- Most often chosen in segmentation task.
- Several versions of parametric ReLU exist with an addition of a parameter to change the threshold of activation.
- Gradients could disappear after several layer with more and more small size.
- Replace the tanh function used in LeNet.

AlexNet vs LeNet

Comparison: AlexNet vs. LeNet

Feature	LeNet	AlexNet
Input Size	32x32 grayscale	224x224 RGB
Number of Layers	7	8
Dataset	MNIST (digit recognition)	ImageNet (1,000 classes)
Activation	Tanh	ReLU
Regularization	None	Dropout
Pooling	Average Pooling	Max Pooling
GPU Utilization	No	Yes (2 GPUs in parallel)
Parameter Count	~60,000	~60 million

Convolutional Neural Network

Dropout Layer

- To prevent overfitting during training.
- It is a regularization technique.
- Randomly remove a part of neurons, typically between 20% and 50% in each layer.
- Each neuron has a chance to be temporarily “dropped”, i.e. set to zero during a particular step.
- Force the network to rely on multiple neurons to represent features at training step.
- Increase the robustness : at the test step, all neurons are used but their weights are scaled down to match the effective capacity of training.
- Can be added after convolutional layers or full connected layers.

Convolutional Neural Network

BatchNorm

- Goal : modifying the tensor data in order to obtain an average value equal to 0 and a variance equal to 1.
- Allow to obtain data in the same range than the activation function which are often between 0 and 1.
- As each chanel has the same distribution, they can be mixed and updated in the same way with the same learning rate.
- It is called batch normalisation because this normalisation is done for each batch.

Convolutional Neural Network

Data Augmentation

- More the network is huge more data you need.
- To reduce overfitting, modifying data to generate new samples is the usual way to do.
- Operations available : random cropping, horizontal flipping, angle rotation, modifying color channel, filter addition, light changes, creating patches ...



Convolutional Neural Network

Synthetic Data

- GAN : Generative Adversarial Networks

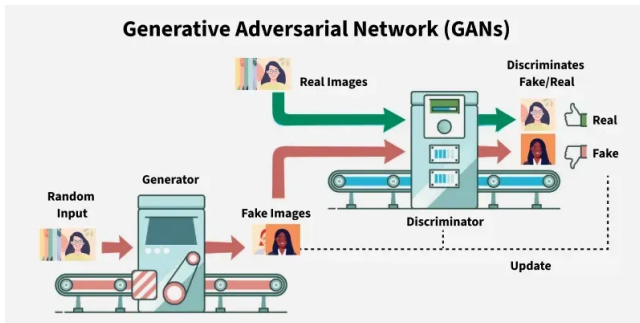


Figure: from <https://www.geeksforgeeks.org/deep-learning/generative-adversarial-network-gan/>

Convolutional Neural Network

VGG - the next one

- After AlexNet victory, new architectures have been proposed for the ILSVRC challenge.
- In 2014 - from Simonyan and Zisserman, Visual Geometry Group : VGG16 and VGG19.
- Main idea :
 - reduce the convolution kernel to 3X3.
 - add more convolution layers.

VGG Architecture

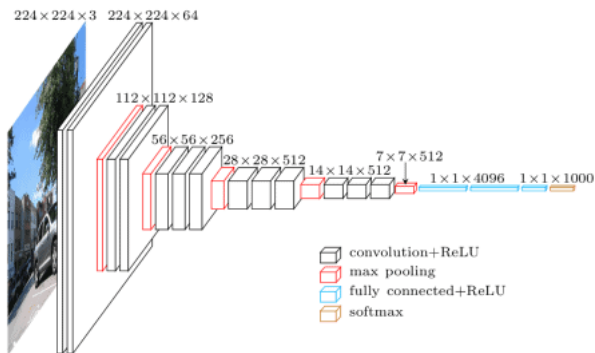
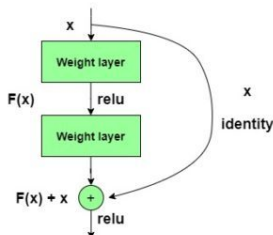


Figure: from <https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/>

ResNet Architecture

What is ResNet?

- From Kaiming He et al. in 2015 “Deep Residual Learning for Image Recognition” (computer vision research paper).
- Problem to solve : when the number of layers increases the accuracy level can get saturated, not by overfitting but vanishing or exploding gradient values.
- Introducing the skip connections to set up shortcuts for the gradient to pass through. $y = F(x) + x$



ResNet Architecture

Residual Block

- Each residual block consists of:
 - A 3x3 convolution layer followed by a batch normalisation layer and a ReLU activation.
 - Another 3x3 convolution layer and a batch normalisation layer.
 - The skip connection skips both convolution layers and adds the inputs directly before the ReLU activation function.

ResNet Architecture

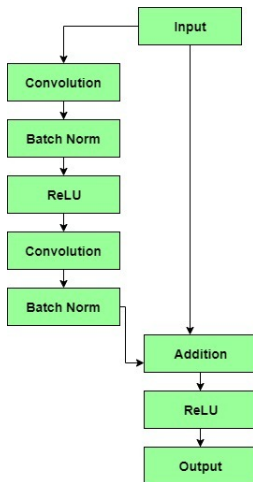


Figure: from <https://www.geeksforgeeks.org/machine-learning/introduction-to-residual-networks/>

ResNet Architecture

Usage

- ResNet is usually used as a main backbone in a lot of architectures.
- ResNet50 is the most popular.
- Mainly used for all image processing : image classification, semantic segmentation, object detection ...
- The backbone role is to extract features from the images and to produce feature maps at different levels of resolution.
- The feature maps at the lower level contain spatial information.
- The feature maps at the higher level have richer semantic information due to their large receptive field.

Feature Maps

Description

- A way to capture the output activations of convolution layers.
- Provide insights into how the network processes and interprets input data at various stages.
- Feature maps are outputs of filters or kernels applied to input image by convolution layers.
- Able to capture patterns present in the input image.
- In PyTorch it is possible to visualize with the help of torchvision.

Feature Maps



Figure: from <https://www.geeksforgeeks.org/deep-learning/visualizing-feature-maps-using-pytorch/>

Feature Maps

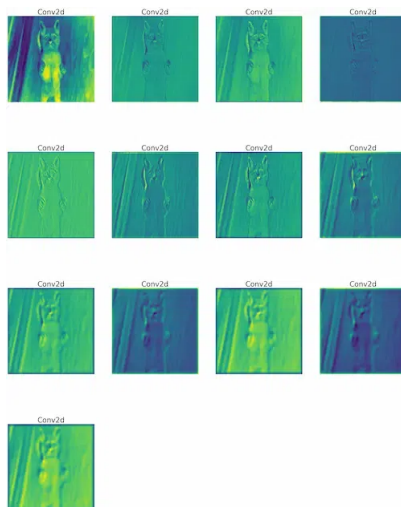


Figure: from <https://www.geeksforgeeks.org/deep-learning/visualizing-feature-maps-using-pytorch/>