Neural Network

Gian Marco Balia
Robotic Engineering - University of Genoa
s4398275@studenti.unige.it

***Abstract*—Artificial neural networks are a fundamental approach in machine learning due to their ability to model complex relationships and learn meaningful representations from data. Auto-encoders, a specific class of neural networks, are widely used for dimensionality reduction, data compression, and unsupervised learning tasks. This report explores the fundamental principles of neural networks and auto-encoders, analysing their architecture, training process, and practical applications.**

***Index Terms*—Artificial neural networks, auto-encoder**

I. Introduction

Artificial neural networks in the field of machine learning represent an innovative approach to tackling complex problems across various domains, from image processing to sequential data prediction. Auto-encoders are among the earliest neural network architectures designed to learn latent representations of data in an unsupervised manner. They operate by compressing data into a reduced-dimensional space and reconstructing the original input, aiming to capture relevant features and redundancies.
The importance of auto-encoders lies in their versatility: they have been successfully applied in tasks such as dimensionality reduction (Hinton e Salakhutdinov, 2006), anomaly detection (Chalapathy e Chawla, 2019), and deep network pre-training. Furthermore, recent advancements, such as the introduction of variational auto-encoders (Kingma e Welling, 2013), have expanded their applications to include synthetic data generation and probabilistic learning. This report aims to provide an in-depth analysis of the fundamental concepts related to neural networks and auto-encoders, illustrating practical examples and experiments that demonstrate their effectiveness in real-world applications.

II. Material and methods

A. Data processing

For this analysis, the MNIST dataset (Modified National Institute of Standards and Technology) was used, one of the most well-known datasets for image classification tasks. This dataset contains images of handwritten digits (0 to 9), where each image is greyscale, has dimensions of $28x28$ pixels, and each pixel has a value between 0 and 255, representing the intensity of the colour (0 = black, 255 = white).

After loading the data, they were divided into two main arrays:

- *(x_{train},y_{train})*: contains the training images, represented as $28x28$ matrices that describe the pixel values, and the corresponding training labels (from 0 to 9), which represent the digits associated with the images.
- *(x_{train},y_{train})*: contains the images for the test, represented as $28x28$ matrices that describe the pixel values, and the corresponding labels for the test (from 0 to 9), which represent the digits associated with the images.

Then, the following steps were done to make the data compatible with the model architecture.

- *Data Filtering*: the data were filtered to include only the digits 1 and 8. This reduced the dataset to two classes of interest. The filtered data were then split into distinct datasets: $x_{train18}$ and $y_{train18}$, 80% of the filtered data, used to train the model; x_{test18} and y_{test18}, that contain the remaining 20%, used to test the model.
- *Data Normalization*: to improve performance and ensure compatibility with the model architecture, the data were normalized, converting the pixel values from their original range (0-255) to a range between 0 and 1. This transformation is essential to accelerate convergence during training and reduce the risk of numerical instability.
- *Data Reshaping*: since the MNIST images, when loaded, have the shape like a three-dimensional array ($numsamples$, 28, 28), where: $numsamples$ is the number of images in the dataset, and $28x28$ are the dimensions of each image. Many deep learning models, require the data to be represented in a four-dimensional format: ($numsamples$, *height*, *width*, *channels*). In the case of MNIST images, the channels value is 1 because the images are greyscale. Thus, the data were reshaped from the form ($numsamples$, 28, 28) to ($numsamples$, 28, 28, 1) using the *reshape* command. This step explicitly adds the required depth channel, making the data compatible with the auto-encoder architecture.

B. Auto-encoder

The autoencoder is an artificial neural network model designed to learn a compact and meaningful representation of input data. The model consists of two main parts: the encoder, which receives the input data and compresses it into a lower-dimensional representation, and the decoder, which receives the compressed representation and reconstructs it into the original output data.

The autoencoder model used in this work is composed of the following parts:

- *Encoder*: the neural network that receives the input data and compresses it into a lower-dimensional representation. The encoder consists of a series of neural network layers, each applying a nonlinear transformation to the input data.
- *Decoder*: the neural network that receives the compressed representation and reconstructs it into the original output data. The decoder consists of a series of neural network layers, each applying a nonlinear transformation to the input data.
- *Autoencoder*: the complete model that combines the encoder and decoder. The autoencoder is trained to minimize the reconstruction error, which is the difference between the original input data and the reconstructed output data.

The autoencoder model operates as follows: the input data is fed to the encoder, which compresses the data into a lower-dimensional representation. The decoder receives this compressed representation and reconstructs it into the original output data. The reconstruction error is calculated as the difference between the original input data and the reconstructed output data. Finally, the model is trained to minimize the reconstruction error using an optimization algorithm.

C. Model evaluation

The methods for evaluating the correct implementation of the autoencoder model are diverse. The most intuitive method is visual inspection: it involves checking whether the digital reconstruction accurately reflects the original input data (see Fig. [boh]).

An alternative approach is the use of the VAF (Variance Accounted For), a specific metric for assessing the performance of autoencoders. The VAF indicates the quality of the data reconstruction by evaluating how representative and useful the compressed data is for reconstructing the original input. The formula used to calculate the VAF is as follows:

$$VAF = 1 - \left(\frac{variance(true\ data - predicted\ data)}{variance(true\ data)}\right) \cdot 100$$

where:

true data represents the original values; *predicted data* represents the values reconstructed by the autoencoder. The calculated values for each reconstructed data point are reported in Fig. [boh].

Finally, another useful method for evaluating the model's implementation is to compute the information loss during reconstruction. This aspect is graphically represented in Fig. [boh].

III. Results and Conclusion