Contents

Introduction

Second generation artificial neural networks – mainly characterized by fully connected layers that deal with continuous output values – have allowed us to make enormous progresses in many relevant scientific fields. In the same way, also the third generation of artificial neural networks seems to be very promising and it is expected to bridge the gap between neuroscience and machine learning, by using more realistic *spiking* neuron models. These 3rd generation networks are called "Spiking Neural Networks"and are different from usual networks, since they do not deal with continuous values, but they communicate through spikes, that are discrete events that occur at certain points in time. In fact, when a (membrane of a) neuron reaches a certain potential value, it spikes, so it resets its value. This behaviour is inspired by real biological processes and is expected to better model reality. This may seem not so "innovative", since discrete signals are used instead of continuous ones. Instead, this modelling allows us to handle spatio-temporal data, that are real-world sensory data. These spatial and temporal aspects refer to the fact that neurons are connected to neurons local to them and that temporal information of the spikes (that are lost in binary encoding) are now preserved. Indeed, it has been proven that spiking neurons are computationally more powerful than traditional artificial neurons.
In this report we describe a project in the frame of the "Neural Networks"course of Sapienza Università di Roma, that is inspired from the "STDP-based spiking deep convolutional neural networks for object recognition"paper of Kheradpisheh, Ganjtabesh, Thorpe and Masquelier. In particular, the implementation of a Spiking Convolutional Neural Network for object classification is described.

1 Network architecture

An STDP-based spiking deep neural network (SDNN) with a spike-time neural coding has been implemented. In the very beginning of this network, the input image is passed to the temporal-coding layer,that uses Difference of Gaussians (DoG) filters to convert the image into an asynchronous spike train. This is used as input for the successive layer, that is a convolutional one, in which input spikes are integrated and new spikes are emitted if a certain threshold is exceeded, that is when a visual feature is detected. Learning is done through STDP, that is used to learn visual features, and so the weights of the network. Neurons that fire earlier in this phase prevent the others from firing. The convolutional layer is followed by a pooling one, where the visual information is

compressed. Three convolutional and pooling layers (arranged in an alternate and consecutive order) have been implemented to handle larger and more complex visual features through the network. In the end, in the last pooling layer, a global max pooling is performed for the classification, to train a linear classifier. The results are used to find the proper category to which the input image belongs.

Figure 1: SDNN architecture

1.1 Temporal coding

In the very first part of the neural network, the input signal is encoded into temporal-discrete spike events. A Difference of Gaussians (DoG) filter is used to detect the contrasts in the input image and emit a spike, accordingly. The higher the contrast in a certain cell, the more strongly this one is activated in order to fire. The firing time (i.e. the time when the spike is emitted) of a DoG cell is inversely proportional to its activation value. In particular, if an r output is obtained in a cell after DoG, its corresponding firing time will be $\tau = 1/r$. DoG cells are sensitive to both positive and negative contrasts, depending on their two ON-centre and OFF-centre maps and they emit when their activation values exceed a predefined threshold.
An example of DoG filter output is shown in figure **??**. In this picture, a face and a motor (image categories that compose the learning and training dataset) are encoded into discrete spike events.

1.2 Convolutional and pooling layers

A convolutional layer is represented by several neuronal maps, whose role is to detect visual features at different locations. To this extent, the synaptic weights assigned to the same neuronal map will be the same. Each neuron in the convolutional layer generates spikes according to what it receives from the pre-synaptic neurons and when its internal membrane potential reaches a certain threshold. So, at each time step t:

$$V_i(t) = V_i(t-1) + \sum_j W_{j,i} S_j(t-1)$$

the internal potential $V_i(t)$ of the *i-th* convolutional neuron is updated by adding the spike train $S_j(t-1)$ of the *j-th* pre-synaptic neuron, multiplied to the synaptic weight