PyLissom: A tool for modeling computational maps of the visual cortex in PyTorch

Anonymous Author(s)
Affiliation
Address
email

Abstract
Despite impressive advancements in the last years, human vision is still much more robust than machine vision. This article presents PyLissom, a novel software library for the modeling of the cortex maps in the visual system. The software was implemented in PyTorch, a modern deep learning framework, and it allows full integration with other PyTorch modules. We hypothesize that PyLissom could act as a bridge between the neuroscience and machine learning communities, driving to advancements in both fields.

1 Introduction
The fields of neuroscience and artificial intelligence (AI) have a long and intertwined history. Better understanding biological brains could play a vital role in building intelligent machines. However, communication and collaboration between the two fields have become less commonplace [3]. Computational tools that integrate approaches to neuroscience and machine learning, in accessible and documented form, are very scarce in the literature. The availability of these tools could be fruitful for the interaction between neuroscience groups and machine learning groups, and the emergence of new ideas and collaborations between these fields.

The term visual map refers to the existence of a non-random relationship between the positions of neurons in the visual centres of the brain (e.g. in the visual cortex) and the values of one or more of the receptive field properties of those neurons. The term is usually qualified by reference to the property concerned. For instance, a property of the visual neurons is that they react to some orientations in the stimulus more than to others. Orientation map refers to the orderly mapping of orientation preference [5].

Laterally interconnected synergetically self-organizing map (LISSOM) have been proposed in the literature as a computational model of maps in the visual cortex in primates [6].

The main contribution of this work is the design and development of a new software library called PyLissom (section 3) that implements for the first time the LISSOM networks (section 2) in a modern deep learning framework.

2 LISSOM model
The structure of the mammalian early visual areas is well understood. Nerve fibers from the retina project to an intermediate region called the lateral geniculate nucleus (LGN), from which the fibers project to the primary visual cortex (V1). The locations on the retina to which a neuron responds are called the receptive field of the neuron. At a given location on the cortical sheet, the neurons in a vertical section through the cortex respond most strongly to the same eye of origin, stimulus orientation, spatial frequency, and direction of movement. It is customary to refer to such a section
as a column \[^2\]. Neurons in a vertical column in the cortex have similar receptive fields and feature preferences. Nearby columns generally have similar, but not identical, preferences; slightly more distant columns have more dissimilar preferences. The feature preferences gradually vary across the surface of the cortex in characteristic spatial patterns called cortical feature maps.

The core of the LISSOM model consists of a two-dimensional array of computational units representing columns in V1. These units receive input from the retina through the ON/OFF afferent channels of the LGN, and from other columns in V1 through lateral connections (Fig. 1). The units learn through Hebbian adaptation, and compete with other units in a self-organizing map structure \[^4, 7, 8\].

The activation mechanism of a neuron in V1 is computed by combining the afferent contributions from the LGN and the lateral contributions from other V1 neurons. First, the afferent stimulation \(s_{ij}\) of V1 neuron \((i, j)\) is calculated as a weighted sum of activations in its receptive fields on the LGN:

\[
s_{ij} = \gamma_A \left( \sum_{ab \in \text{ON}} \xi_{ab} A_{ab,ij} + \sum_{ab \in \text{OFF}} \xi_{ab} A_{ab,ij} \right)
\] (1)

where \(\xi_{ab}\) is the activation of neuron \((a, b)\) in the receptive field of neuron \((i, j)\) in the ON or OFF channels, \(A_{ab,ij}\) is the corresponding afferent weight, and \(\gamma_A\) is a constant scaling factor. The afferent stimulation is squashed using a sigmoid activation function \(\sigma\), forming the neuron’s initial response as \(\eta_{ij}(0) = \sigma(s_{ij})\). After the initial response, lateral interaction sharpens and strengthens the cortical activity over a very short time scale. At each of these subsequent discrete time steps, the neuron combines the afferent stimulation \(s\) with lateral excitation and inhibition:

\[
\eta_{ij}(t) = \sigma \left( s_{ij} + \gamma_E \sum_{kl} \eta_{kl}(t-1) E_{kl,ij} - \gamma_I \sum_{kl} \eta_{kl}(t-1) I_{kl,ij} \right)
\] (2)

where \(\eta_{kl}(t-1)\) is the activity of another V1 neuron \((k, l)\) during the previous time step, \(E_{kl,ij}\) is the excitatory lateral connection weight on the connection from that neuron to neuron \((i, j)\), and \(I_{kl,ij}\) is the inhibitory connection weight. The scaling factors \(\gamma_E\) and \(\gamma_I\) represent the relative strengths of excitatory and inhibitory lateral interactions.

After the activity has settled, the connection weights of each cortical neuron are modified using the Hebb rule \[^4\] with divisive postsynaptic normalization:

\[
w'_{pq,ij} = \frac{w_{pq,ij} + \alpha X_{pq} \eta_{ij}}{\sum_{uv}(w_{uv,ij} + \alpha X_{uv} \eta_{ij})}
\] (3)
Figure 2: Example of an orientation map obtained by PyLissom. The vertical bar at the right shows the oriented preference of the neuron. For instance, the red color denotes that a neuron achieves its highest activation when the input is horizontal lines.

\[
\text{where } w_{pq,ij} \text{ is the current afferent or lateral connection weight (either } A, E \text{ or } I \text{) from } (p, q) \text{ to } (i, j), \\
w'_{pq,ij} \text{ is the new weight to be used until the end of the next settling process, } \alpha \text{ is the learning rate for each type of connection (} \alpha_A \text{ for afferent weights, } \alpha_E \text{ for excitatory, and } \alpha_I \text{ for inhibitory), } X_{pq} \text{ is the presynaptic activity after settling (} \xi \text{ for afferent, } \eta \text{ for lateral), and } \eta_{ij} \text{ stands for the activity of neuron } (i, j) \text{ after settling. Afferent inputs (i.e. both ON and OFF channels together), lateral excitatory inputs, and lateral inhibitory inputs are normalized separately.}
\]

3 PyLissom

PyLissom is a software developed in PyTorch that implements the LISSOM model (section 2). A picture of an orientation map of the visual system obtained by PyLissom is shown in Fig. 2. PyLissom’s main features are the following:

- it extends Pytorch’s neural network module by implementing Lissom layers, allowing to build hierarchical models of the visual system
- it allows to be used for machine learning applications since it can combine the LISSOM modules with other PyTorch modules
- PyLissom has an object-oriented design, and this increases its adaptability and reuse
- PyLissom is implemented in Python and PyTorch, which are programming languages currently highly used in the neuroscience and machine learning communities
- PyLissom can parallelize their computations on GPUs

The code of PyLissom is at the following link: [https://test.pypl.org/project/pylissom](https://test.pypl.org/project/pylissom)
The user manual of PyLissom is at the following link: [https://pylissom.readthedocs.io](https://pylissom.readthedocs.io)

4 Related work

LISSOM networks were implemented previously in a computational system called Topographica [1]. The use case of the Topographica software is the neuroscience community. The Topographica software has been successfully used by some researchers to validate computational models in neuroscience. However, due to its design, it is complicated to adapt Topographica for machine learning use cases.

5 Conclusions

The software developed in this work was designed as a joint use case for the machine learning and the neuroscience communities. It allows to obtain fully usable LISSOM modules in PyTorch. We expect that PyLissom could be fruitful for the interaction between neuroscience groups and machine learning groups, and the emergence of new ideas and collaborations between these fields.
References


