Emergence of complex like cells in a temporal product network and its smooth generalization

Karol Gregor, Yann LeCun
Department of Computer Science
New York University
715 Broadway Fl 12,
New York, NY, 10003
karol.gregor@gmail.com

One of the important ingredients in deep learning is the creation of invariant representations at every level in the hierarchy in an unsupervised fashion. In the cortex, at a low level there are complex cells that respond to edges of specific orientation and frequency but are invariant to a range of positions. At higher levels there are neurons that respond to specific objects and are invariant to object position, lighting, orientation, specific shape or even type. At the same time the algorithms that we use need to be able to work with real world data such as large enough sized natural videos obtaining good performance for example on classification.

In this work we introduce two concepts. The first is an algorithm for producing invariant representations based on breaking down an input into two complementary representations, one of which is an invariant representation learned using a new implementation of temporal consistency. Training on patches of video produces cells with complex cell like properties. The second is a network that applies to entire images, not just image patches and has smooth geometry in a sense that it has no artificial cuts such as the ones produced in sub-sampling. We have a reason to believe that its representation is more efficient than that of the convolutional neural network, which is a limiting case of this one.

Approximate properties of simple cells can be obtained using for example the simple sparse coding algorithm[1]. Invariance properties or properties of simple cells are harder to obtain. Ideas include topographically organizing simple cells and pooling the neighbors[3], making the evolution of hidden states slow[4] when training on moving patches, producing a transitional matrix and forming groups[5] and a two layer statistical model of images in Ref 6. Our algorithm is quite different, it contains product interactions, extracts invariance using temporal consistency which is different from slow feature analysis and has an encoder for fast feed-forward inference. We also provide a smooth generalization to medium sized videos.

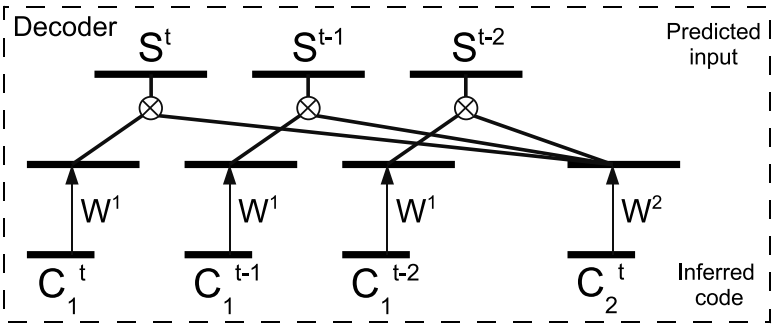

FIG. 1: The architecture of the temporal product network. The S is a sparse representation such as that of simple cells. Time is used to separate invariant complex cells C_2 from the complementary cells C_1 by requiring the same invariant cells to reconstruct the simple cells at various time steps. The position cells then complement the reconstruction by picking specific input at a given time.

The first of the two concepts, which we call temporal product network can be explained as follows. Given a sparse representation S, we want to represent it as a product of two properties P and C (see Ref. 7 for related ideas). Say the S is the standard sparse representations of the input approximating simple cells[1]. The P and C are collections of cells representing position and orientation. A given cell in each P or C has strong connections to all the S cells having the corresponding property. For the complex cell they are all the edges of a given orientation at a range of positions. For the position cell they are edges of different orientations at the same position. If we activate one complex cell and one position cell and activate the simple cells that have strong connections to both (by means of the product of the activities of simple cells the position or complex cell prefer) we obtain one simple cell having the two properties. Of course we use sparse representations instead of activating just a single cell.

Another ingredient is a temporal training that separates the two representations, giving the network in the Figure 1. The second code, corresponding to complex cells attempts to reconstruct all the inputs that appear at several time

steps. The first code has different values at different time steps and is able to select the particular input that appears.

When training on patches of video it leads to the results shown in the Figure 2. Any particular cell in the second code (complex cell) has strong connections to several simple cells of similar orientations and frequencies that are located in a range of positions. Each complex cell responds to edges in a similar range of orientations and frequencies as a simple cell but in much larger range of positions. The responses are smooth.

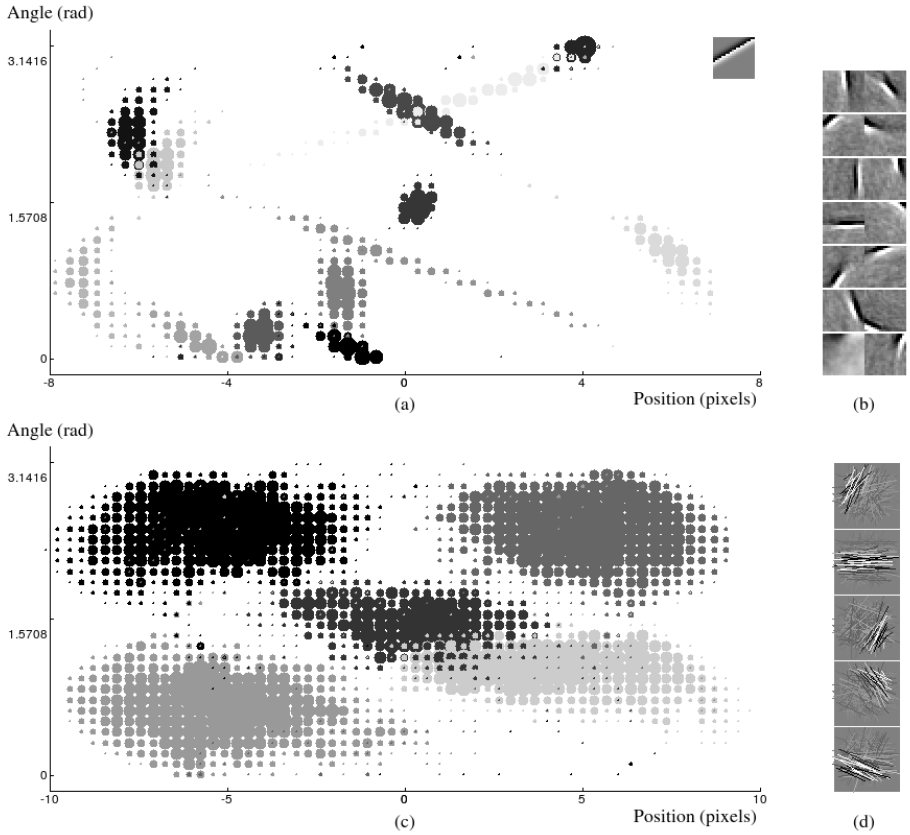

FIG. 2: (a) Responses of selected simple cells to an edge shown in the upper right corner. The angle is the orientation of the edge and the position is its distance from the center. Circles of different shading correspond to different cells and the size of the circle is proportional to its activity. (b) The corresponding simple cell filters. (c) Responses of the complex cells to an edge moving at 0.0312 pixels per frame (practically stationary). (d) Each square correspond to one complex cell in (c). We fitted a gabor function to each simple cell filter, extracted its orientation and position and plotted the line with this orientation and position. The intensity of the line is proportional to the strength of the connection of a given complex cell to the simple cells. We see that each complex cell is connected to simple cells of similar orientation in a range of positions. We also see that complex cells respond smoothly to input edges of a given orientation in a much larger range of positions then the simple cells, whereas the range of angles is very similar. Increasing the speed of the motion smoothes the responses.

Note that this temporal training is different from a slow feature analysis[4]. Here, at every time we use a new code to reconstruct several time steps. This prevents old representations from affecting the current one and the correct representation is figured out at every time step. We also note that at present our position cells don't encode position, they are merely a complementary cells that are connected to different orientations.

The next component is adding an encoder. The encoder quickly infers an approximation to the code[2]. This is used as an initial guess in the optimization and would be used in a recognition system for fast inference if we build one in the future. For nonovercomplete code and video patches the encoder performs quite well, similarly to the encoder of the simple cell network.

Now we turn to the second idea mentioned at the beginning. We design a network that applies to full-sized images, has local receptive fields and is smooth. Consider an image and a layer of hidden units, corresponding to simple cells. Spread out the simple cells uniformly above the input and connect each simple cell to a neighborhood of inputs directly below it (receptive field). Then train it as a decoder using the standard sparse coding algorithm. The result will be the usual Gabor filters of various orientations and frequencies but this time located at different positions. To speed up the training we note that the connections that are far enough are not competing. Thus we bind all

connections whose coordinates are multiples of an integer away from each other. This leads to much smaller number of connections and just as nice features.

For periodicity equal to one pixel this reduces to a convolutional neural network (CNN). The over-completeness of the code equals the number of features of the CNN. We think that the representation above with periodicity large then one is more efficient then CNN for the following reason. When we take a complete (same number of hidden units as the input) smooth network, edges of different frequency and orientation arise. The corresponding CNN would have only one feature and thus useless. When the network is overcomplete by a factor n the CNN has n features but the good thing is that all the positions are represented for each feature. For the smooth network, while we have similar features at various positions, all the features are distributed the best they can. Instead of having the same feature at all the positions, only whatever positions are needed most are used and there is more room to fit in different looking filters.

The temporal product network can be build in smooth way analogously to the simple cell network. We train the system on natural videos of size roughly 200×200 and we obtain complex like cells as well. We can also compare images that are generated using simple and complex cell network and we find that later have more edge like features.

---

[1] Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images, Olshausen BA, Field DJ. Nature, 381: 607-609 (1996)

[2] Marc'Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau and Yann LeCun: Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition, Proc. Computer Vision and Pattern Recognition Conference (CVPR'07), IEEE Press, 2007,

[3] Aapo Hyvarinen, Patrik O. Hoyer, A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images, Vision Research 41 (2001) 24132423

[4] Pietro Berkes, Laurenz Wiskott, Slow feature analysis yields a rich repertoire of complex cell properties, Journal of Vision (2005) 5, 579-602

[5] Numenta's HTM Whitepaper: Concepts, Theory and Terminology

[6] Y. Karklin and M. S. Lewicki, Emergence of complex cell properties by learning to generalize in natural scenes, Nature, 457: 83-86, 2009.

[7] Andrew Brown, Geoffrey Hinton, Products of Hidden Markov Models. T. Jaakkola and T. Richardson eds., Proceedings of Artificial Intelligence and Statistics 2001, Morgan Kaufmann, pp 3-11, and related product works