GPGPU-BASED CORTICAL MODELING

THEODORE HILK

Abstract. Cortical modeling is an area of research seeking to model and simulate the cerebral cortex of the brain, which is of fundamental importance to conscious thought and action. Computational power is a major challenge in this field and the problem is inherently well-suited to SIMD architectures. This suggests the implementation of a general-purpose GPU framework for the development and execution of cortical models, and indeed, several such frameworks do exist. However, they suffer from hardware and software vendor lock-in and unnecessary assumptions that limit the generality of the models they can execute. In order to overcome these obstacles in preparation for anticipated future work by the author and others, we have implemented a new cortical modeling framework in OpenCL using PyOpenCL. The new framework has several notable advantages: it is open-source, it does not suffer from hardware or software vendor lock-in, it is cross-platform compatible, and in principle it can simulate any model expressed as a message-passing architecture on a defined graph of nodes grouped into atomically-executed regions. Further, a given model requires only a minimal amount of C code (using the industry-standard, vendor-neutral OpenCL C), with the remainder written in Python. All overhead associated with managing and executing the model is handled automatically by the framework. The user need only specify the node algorithm(s), network layout, regional execution order, and inter-node i/o definition, which are the essential components of any cortical model. We have opted to name the new framework "Phoenix".

Part 1. Cortical Modeling

The cerebral cortex plays a central role in perception, representation, memory, language, attention, and motor control. Indeed, it would be little exaggeration to claim that it is the location of these functions [9, 13, 23, 41, 44, 53].

Further, the structure of the cortex is largely uniform, comprised of six canonical layers organized into millions of narrow columnar structures containing a particular pattern of feedforward and feedback connections between neighboring columns and larger regions. While the relative strengths of many regional connections are not yet known electrophysiologically and regional variations in layer thickness do exist, the basic local structure and experimental evidence suggest a common algorithmic implementation with a relatively small number of specific modifications across the cortex [38, 9, 52, 22].

Date: December 17, 2012.

Cortical modeling, then, seeks to model and simulate the information-processing operations undertaken within the cortex. This is generally at a more abstract level than the attempted simulation of individual neurons. The goals of the field lie primarily in improving the interpretation of results from experimental neuroscience, developing a more complete understanding of the brain, and contributing to the development of better artificial intelligence systems [18, 22, 25, 12, 13, 33, 31, 36, 44, 45, 56, 63].

1. Cortical Modeling: Themes

A proposed cortical model for perception must be capable of explaining various known properties from neuroscience and experimental psychology. These key characteristics include online learning, hierarchical representations, spatial clustering, temporal clustering, feedback, prediction, dimensionality reduction, and the gestalt properties of emergence, reification, multistability, and invariance [18, 34, 60, 14, 13, 22, 25, 4].

Any putative cortical model must permit learning stored representations from incoming examples at the same time the model itself is classifying these examples. In machine learning parlance, this characteristic is known as "online learning," in contrast to "batch learning" methods that have separate training and test phases or modes of operation [60, 36, 22, 25, 42].

The representation of stored information in a cortical model must be hierarchical. It is clear from experimental data in neuroscience that the brain has a physically hierarchical structure. Afferent connections convey sensory information to the bottom layers of the hierarchy and it propagates upward through successively higher regions of cortex, with various side paths and bypass routes. Other experimental work has indicated that lower regions store more elementary feature representations, while higher ones store more complex and general ones. Correspondingly, the best-performing object recognition systems use hierarchical architectures based around this principle [18, 4, 63, 13].

Spatial clustering refers to the notion that inputs which are very similar to one another should be combined into a single representation. Because this occurs at all levels of the hierarchy, which corresponds to all levels of stored feature complexity, very little meaningful representational ability is lost. E.g., one cannot likely remember every single box of a particular brand of cereal that one has ever seen, but such recall is rarely important for any task. This process is also important in the development of invariant "prototypes" of objects, e.g. the generic notion of a chair, which have a long history of evidence in psychological and neuroscientific experiments [54, 55, 36, 22].

Temporal clustering means that the network stores common transition patterns between sensory representations – e.g. motion of objects through space, transformations of objects over time, etc. These patterns are likewise stored in a hierarchical manner, and are associated with the corresponding

spatial representations on multiple levels of abstraction. Note that spatial and temporal clustering are relevant to sensory modalities other than vision as well. For instance, if one has memorized a song, one has learned both the chords in the song (spatial structure in frequency space, extracted by the cochlea) and the specific transitions between these chords (temporal structure in frequency space, the change in the spectrogram over time) [60, 22, 25, 36, 52, 41].

Feedback refers to the notion that higher-level representations provide feedback signals to lower-level ones. The need for such feedback in a putative cortical model is supported by neuroscientific evidence for cortical circuits directed from higher regions of the brain to lower ones [9, 60, 4, 22]. From a representational standpoint, feedback enables the completion of occluded images, interrupted speech, etc. Cortical models implementing feedback in this manner have improved tolerance for errors and noise [21, 43, 60]. A back-and-forth process of constraint propagation between lower-level and higher-level regions seeking a "match" could enable initially incomplete inputs to trigger the corresponding higher-level representation in a manner similar to auto-associative memories [10, 1, 60, 13, 22]. (We note as an aside that this would correlate temporally with the feeling of attempting to recall the specific structure of an entity one has nearly forgotten.)

Prediction is the consequence of feedback applied to temporal patterns as opposed to spatial ones. This is because high-level spatiotemporal representations that provide feedback to lower-level ones can "prime" the lower-level ones for anticipated incoming stimuli. Further, if these stimuli are not subsequently present, the mismatch can act as a bottom-up signal for attention [60, 51, 22].

Dimensionality reduction occurs because the network must learn sparse features at multiple levels of abstraction from sensory inputs at the bottom layer that are extremely high-dimensional. For instance, the retina has about a hundred million photoreceptor cells, and even after these are consolidated into the optic nerve there are still millions of distinct visual inputs. The low-level region V1 of the visual cortex consolidates these into low-level features such as edges and colored regions in particular locations across the entire visual field, which are then presented as inputs to higher-level regions, starting with V2. These regions in turn recognize progressively more complex features with progressively less positional and orientational specificity, creating a "space of representation" that extends e.g. from lines and colored regions, to more complex intermediate shapes in V2 and V4, to representations of individual characters invariant to size and font type in IT, to representations for individual words in a language in higher regions. Note that a considerable amount of feedback is involved in search tasks and for ambiguous inputs, e.g. distinguishing the identity of an illegible character based on the word in which it appears [19, 60, 66, 13, 14, 23, 24].

FIGURE 1. Gestalt Properties: Emergence. This image depicts an entity, the nature of which will not be discussed here. Viewers not previously exposed to this image will note a sudden flash of insight when the descriptive complexity of the image is reduced by the realization of its high-level structure. This "emergence" of high-level structure from low-level features is the first Gestalt property of perception.

2. Cortical Modeling: Themes: Gestalt Principles

Gestalt properties are well-known perceptual abilities from psychology research on static images that any convincing cortical model must be capable of explaining. The first is "emergence": the notion that high-level features must be able to "emerge" out of patterns in low-level features [34, 53, 60]. See figure 1 for a rather dramatic example, particularly if one has not previously seen the image in question. (If one has viewed it and understood it in the past, the memory of its high-level structure – likely at least partly within the IT region of the visual cortex – causes one to immediately perceive it directly instead of through a later, sudden flash of insight.) We contend that a hierarchical model with spatial clustering is sufficient to explain emergence.

The second Gestalt property of perception, reification, refers to the ability of the mind to fill in the gaps in incomplete images and imbue them with a spatial character [34]. See figure 2 for sample images. We contend that a hierarchical model with spatial clustering and feedback would be sufficient to explain reification [60, 44, 13].

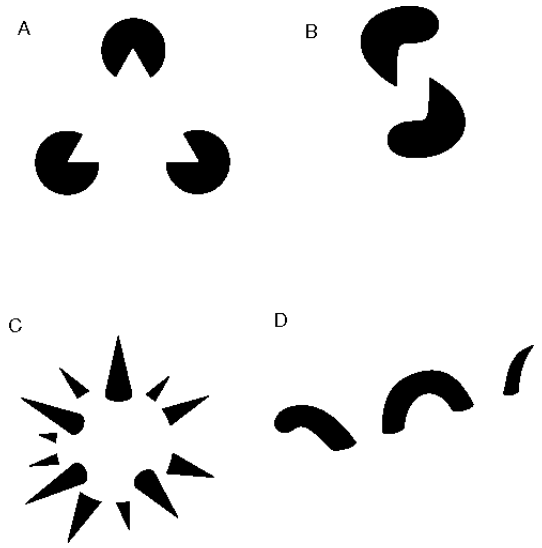

FIGURE 2. Gestalt properties: reification. The images above are all incomplete in some way, but the mind can complete them. Image (A) might be perceived as a triangle, but it does not in fact contain any triangle. Image (B) might appear akin to a worm wrapped around a cylinder, but it does not contain a complete image of the worm or even any three-dimensional information. Image (C) might seem to depict a spiky sphere, but again contains no three-dimensional information. Image (D) could be perceived as a "sea serpent" on a body of water, although again, no three-dimensional information is present, nor is most of the "serpent" or even the water *per se*.

Multistability is the third Gestalt property of perception. Ambiguous peceptions can have multiple mutually contradictory interpretations, each of which is individually stable. This is illustrated more concretely in figure 3. As Lehar wrote, "The significance for theories of visual processing is that perception cannot be considered as simply a feed-forward processing performed on the visual input to produce a perceptual output, as it is most often characterized in computational models of vision, but rather perception must involve some kind of dynamic process whose stable states represent the final percept," [34]. Indeed, we conjecture that the oscillatory constraint-propagation phenomenon discussed above would explain multistability. Stability would be characterized by a match between the inputs from a lower-level unit to a higher-level one and the latter's feedback to the former, as in, e.g., adaptive resonance theory or modified versions of the original hierarchical temporal memory model [10, 22, 60].

The fourth and most important Gestalt perceptual property is that of invariance. Invariance refers to the ability to identify objects in sensory input

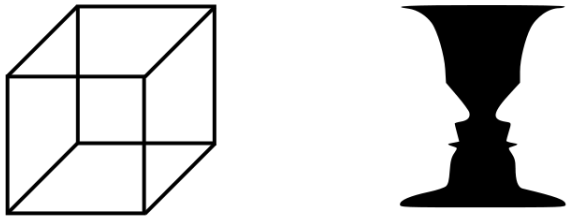

FIGURE 3. Gestalt properties: multistability. The "Necker cube" on the left may be envisioned in one of two orientations in three dimensions. Both are stable, but one tends to "flicker" between them and can even choose to focus on one or the other. (We conjecture that the latter is caused by top-down attentional signals from regions of cortex above IT and possibly above the association cortex.) Likewise, the image on the right may be alternately perceived as a vase or as two faces.

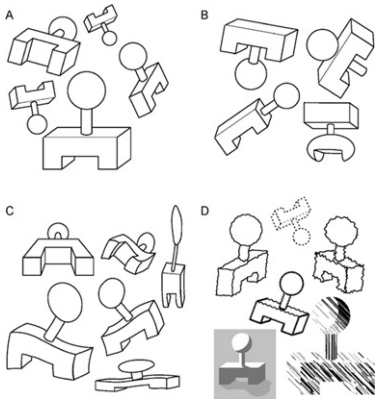

FIGURE 4. Gestalt properties: invariance. A) Object is rotated, but still easily distinguished as identical. B) Versions of object with rearranged parts are recognized as essentially similar to object, but not identical. C) Warped and scaled images are still clearly depictions of the same object. D) Views of object drawn according to different styles are still understood to be identical.

even when they are perceived under different conditions, orientations, transformations, and so on. Figure 4 provides more concrete examples. Invariance was the central focus of Gestalt theory, and has since become the central focus of computational models for perception. Research by Fukushima, Riesenhuber, and Poggio among others has convincingly demonstrated that

hierarchical feedforward networks with spatial clustering or an equivalent operation are sufficient to explain much of invariance, although work continues on the development of effective models for three-dimensional transformations [19, 60, 14, 44, 13].

3. Existing Cortical Models

A variety of models fall under the general classification of cortical modeling. Fukushima's neocognitron was one of the first [18, 20, 21]. Work by Riesenhuber on the "HMAX" model, as well as later developments by him, Serre, and Poggio do as well [56, 51, 61, 60, 45]. Miller and Lommel developed another model called the HQSOM [36]. George and Hawkins developed two models called the "HTM," the latter of which subsequently evolved into a commercial product for video processing and the analysis of other large, structured datasets [22, 25]. Deep-belief networks may also be viewed from the standpoint of cortical modeling, although they are not as specific to cortical functionality [27, 31, 32]. Many additional models beyond these have been developed as well[63, 44, 13].

4. Gaps in Current Models

While many models have been constructed that exhibit some or even most of these desirable qualities on some level, to date there exists no proposed model that can demonstrate all of them to an extent commensurate with that of the brain [44, 13, 63].

Meanwhile, viable extensions to such models to account for bottom-up and top-down attentional modulation are very much a work in progress, and at present some of the most successful models in the area rely on older, non-hierarchical networks [67, 15, 17, 30, 44, 51, 63].

Hierarchical models for motor control, the relationship between reflexes and voluntary motor control, and the relationship between innate or "instinctual" motor responses and voluntary motor control remain likewise nascent [29, 3, 6, 42, 57, 64].

Hierarchical models for the operant learning of motor control and top-down attention, commonly bundled into the term "executive function," are a very active area of research, mostly centered on the application of reinforcement learning to hierarchical models. Working memory is often associated with them and is likewise much-debated. A large variety of related research is ongoing and a number of models have been proposed, but less simulation and testing has been done than for perceptual models [2, 5, 8, 7, 16, 26, 49, 50, 59, 65, 68].

Attempts to model episodic memory, emotional responses, the "consolidation" of long-term memory, the emotional "tagging" of memories, and the effects of emotional tagging on memory consolidation within the context of low-level cortical models appear somewhat less common, although many older auto-associative models have been used. This may be related to the structure

of the hippocampus itself, however, parts of which are known to have dense networks of recurrent connections different from those in the neocortex. Also, a variety of conceptual models do exist in contrast to the reduced emphasis on implemented computational work [2, 11, 28, 35, 37, 47, 48, 58, 62].

The vastness of the remaining research necessary to understand the brain on an implementational level underscores the need for better tools and further progress in the field.

Part 2. GPUs in Cortical Modeling

5. General-Purpose GPU Computing

Commodity graphics cards can now provide multiple teraflops of (theoretical, heavily optimization-dependent, rather memory-bound) compute power for less than $200. This is a stark contrast to the present situation facing CPUs, where difficulties with continued die shrinks and limited power scaling have effectively begun to amend Moore's Law.

An array of general-purpose GPU computing architectures, of which NVIDIA's CUDA was the first to reach mainstream status, has considerably simplified the task of writing scientific applications for execution by GPUs. OpenCL, a more recent industry standard, permits generic GPU computing in a vendor-neutral fashion and even extends support to CPU multiprocessing if needed. The development of these toolchains and execution environments has permitted numerous applications in scientific computing across engineering, geophysics, meteorology, physics, bioinformatics, computational finance, and numerous other areas of research.

However, the excellent prima facie performance-price ratio for GPUs comes with a substantial limitation: they present compute resources in the form of a small number (typically 10-32) of wide-SIMD processors (typical width 40-80 scalar elements, often further subdivided into vector units of 4 similar to SSE). See figure 5 for an example. Until recently, NVIDIA relied on a RISC architecture while ATI used a very long instruction word (VLIW) architecture, but as of late 2011 ATI had switched to RISC and a greater focus on thread-level parallelism to further improve compute performance for complex tasks and simplify the development of compilers and applications.

Also, double-precision floating-point operations impose a substantial performance penalty on cards that permit them, although newer architectures have ameliorated this problem to some extent. Finally, memory bandwidth can raise substantial challenges in GPU computing, particularly with respect to transfers from main memory to on-card memory.

Consequently, only heavily parallel workloads are well-suited to GPU computing, particularly those that can be easily framed as vector math, that do not rely on double-precision floating point operations, and that do not require significant memory bandwidth to main RAM.

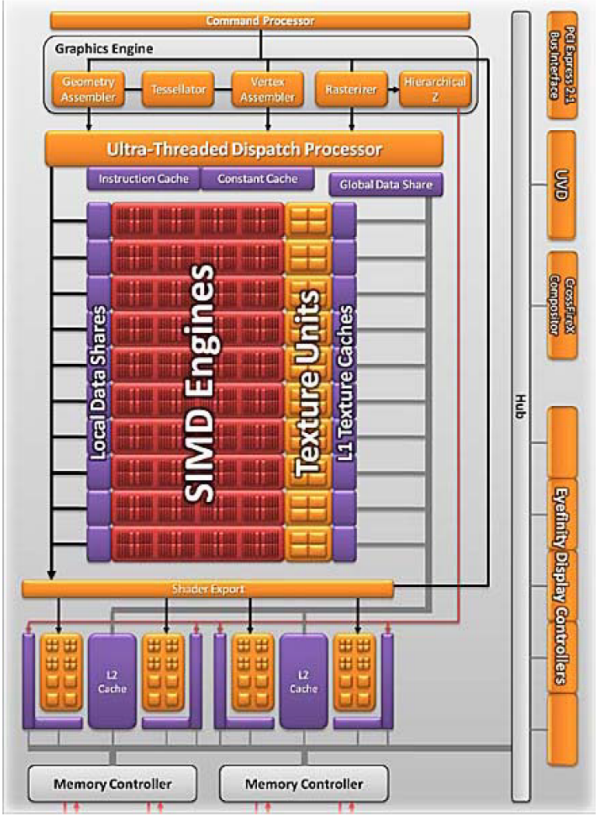

FIGURE 5. A typical GPU architecture. This diagram depicts the ATI Radeon HD 6870 "Barts XT," which was released on Oct. 22, 2010 and at the time of this writing can be purchased for roughly $USD 160. It contains 1120 vertex shaders, 56 geometry shaders, and 32 pixel shaders operating at 900MHz, for a combined hypothetical 2.016 teraflops. These resources are accessible across 14 distinctly-addressable units, called simply "SIMDs" in Radeon parlance or "compute units" in OpenCL. The 255 mm^2, 40 nm process die contains 1.7 billion transistors. The newer HD 7970 "Tahiti XT" has a 2048:128:32 configuration and 32 distinctly-addressable units, achieves a hypothetical 3.788 teraflops, and has a 352 mm^2 28 nm process die with 4.313 billion transistors for $USD 420.

6. GPUs in Cortical Modeling

Conveniently, however, the brain is a massively-parallel system [38, 63]. More concretely, the cerebral cortex consists of a relatively small number of regions that each contain a very large number of cortical microcolumns, all of which operate in parallel and send information upward, downward, and laterally within the hierarchy's layers [38, 60, 22]. Further, as a biological

system with a high degree of redundancy, extreme tolerances in internal representations and output actions are a non-issue to the extent that they are not introduced as a requirement by one's choice of implementation and numerical methods.

Examining these characteristics in turn from the standpoint of a cortical model, we note that most clustering algorithms and various simpler, special-purpose approximations used in cortical modeling consist entirely of vector arithmetic [44, 22, 21, 36]. Further, an execution approach based on simulating modeled cortical layers in turn on the GPU (as a stand-in for simultaneous operation within the biological system) would be required to wait until all layers had a chance to execute before beginning work on the input for the next time step. This bodes well for memory bandwidth. Thus, we conclude that general-purpose GPU computing is well-suited to cortical modeling.

7. Existing GPGPU Cortical Modeling Frameworks

Two major frameworks for the simulation of cortical models on GPUs currently exist. The first, CNS, was developed in 2009. It is written in Matlab, so the software necessary to use it is not free or open-source. As an aside, the author prefers Python to Matlab regardless, but reader preferences may likewise vary. CNS also relies on NVIDIA's proprietary CUDA architecture, so it cannot run on ATI GPUs. This particularly unfortunate since ATI GPUs have historically performed better than NVIDIA ones on compute-bound workloads like those generated by most cortical modeling applications, although this is subject to ongoing technical developments [46].

CNS automatically performs a considerable number of host-GPU memory transfers with no direct control by the user, which could conceivably become problematic in light of memory bandwidth concerns. This is of particular concern for consumer-grade cards, which are approximately an order of magnitude cheaper than specialized ones but have poorer host-GPU memory bandwidth at a given level of computing power. It also double-buffers inter-node i/o on the assumption that users might otherwise make mistakes, which consumes additional memory. Also, CNS does not support double-precision floating-point operations even if needed by a given model [46].

Finally, the framework imposes a variety of assumptions on network structure that can limit the generality of the models one might seek to implement, the somewhat complex and multifaceted details of which will not be discussed at great length here. At a high level, to summarize, the framework largely assumes individual cells to have the same connectivity structure, regions to be uniform and regular, execution order to follow one of a small number of preset patterns, and networks to fall within one of a small number of classes that affect the nature of these constraints. Some mitigations exist, but they are incomplete. Overall, CNS was the first of its kind, was subject to the architectural limitations and library availability of 2009, and was designed

for researchers who do not need full generality. The authors noted many of the limitations above, but some simply could not be easily circumvented at the time [46].

The second framework does not appear to have a name, but was developed by Nere et. al. at the University of Wisconsin–Madison in 2010. This framework is likewise dependent on CUDA. It also imposes much stricter, quasi-biological constraints on the models being simulated. The framework does not appear to be designed for other researchers to be able to use it for more general work than the model that its author had in mind. Indeed, in a later, 2012 paper, the additional applications discussed did not rely on the original framework at all, so it was specific to the particular model in question at that point. However, this did permit Nere to conduct a substantial amount of performance optimization, some of which would have been incompatible with a more general layout. Nere also wrote a runtime profiling tool that is not present in CNS or the model we developed [40, 39].

Overall, existing GPGPU cortical modeling frameworks lack cross-platform compatibility, the ability to script execution without resorting to non-free software, and most importantly a significant level of modeling generality. The latter is particularly useful for the development of new cortical models that may not adhere to the same constraints as prior work.

Part 3. Project Overview

8. Prior Work: 6.867 Final Project

For a final project in a previous class, the fall 2011 iteration of 6.867 Machine Learning, the author collaborated with fellow student Joseph Lynch to implement the HQSOM model from Miller and Lommel 2006 [36]. We were interested in this model because, unlike HMAX and almost all of the other cortical models we surveyed at the time, it performed temporal clustering and did not rely on a hand-created feature dictionary [36, 60]. The only other model we found that met these constraints was the second version of HTM, but we considered it too complex and unwieldy compared to the HQSOM and had read that it sometimes failed to generalize well [25, 22].

During our project work, we implemented the network as described in the paper and successfully replicated its authors' results on several visual classification tasks. We made a variety of improvements to the algorithm itself in the process, including the use of a mean-squared-error-based activation function to improve performance in the presence of noise, a peripherally-inhibitory "Mexican hat" neighborhood function to enhance competitive learning, an adaptive learning rate and neighboorhood function bandwidth to reduce the amount of time required for training, and a "reset" function to increase the efficiency of the training process.

We then extended the model to audio classification. We preprocessed audio files with a windowed fast Fourier transform to generate spectrograms, motivated by the well-established notion from neurology that the cochlea of

the inner ear performs something very much akin to an FFT prior to relaying signals via the cochlear nerve. We also constructed a different network topology with a one-dimensional input space to match that of the spectrogram and three layers instead of two for longer sequence memory, then fed the spectrogram line by line into the model as input. After "playing" one song from each of three musical genres to the network in this manner, we tested its performance in classifying three different songs from the same respective genres. Classification was successful in each case.

However, trials were still very time-consuming in spite of our modifications. For a single-threaded implementation of the algorithm in numpy, the required training cycles for one of the visual classification tasks took several hours on a state-of-the art CPU at the time. We were able to run multiple training cycles for parameter sensitivity studies on separate cores, but it would have been more useful to run the trials in series if we had been able to do so quickly. Computational difficulties limited our ability to apply the network to more ambitious learning and classification problems, or even to perform more extensive experiments with our audio system. It became clear to us that a suitable GPU framework would be a practical necessity in order to work on more complex networks, learn a larger set of representations, or experiment with other modeling approaches.

9. 6.338/18.337 Project Overview

In order to overcome the aforementioned obstacles and the limitations of other GPU frameworks as noted above, the author has implemented a new cortical modeling framework in OpenCL using PyOpenCL, called "Phoenix". This platform has several key advantages: it is open-source and written in a language that is in turn open-source, it does not suffer from hardware or software vendor lock-in, it is cross-platform compatible, and in principle it can simulate any model expressed as a message-passing architecture on a defined graph of nodes grouped into atomically-executed regions. Further, a given model requires only a minimal amount of C code (using the industry-standard, vendor-neutral OpenCL C), with the remainder written in Python.

All overhead associated with managing and executing a given model is handled automatically by the framework. The user need only specify the node algorithm(s), network layout, regional execution order, and inter-node i/o definition, which are the essential components of any cortical model. It is our sincere hope that this simulation environment will be useful in both our own future work in the area and that of others who may have found themselves limited by existing solutions.

10. The Node Representation

Nodes in Phoenix are implemented as subclasses of a generic parent "Model" class. This class contains an __init__() method to handle setup and various utility methods used for inter-node i/o and certain interactions with

PyOpenCL. Alltogether, these methods handle all aspects of feedforward and feedback i/o between nodes according to defined input and output sizes for both feedforward and feedback (any of which may be zero if not used) and references to its parent and child nodes (either of which may be empty if not present). In more detail:

The __init__() method acquires the OpenCL context and queue, obtains a reference to the target device, looks up and applies appropriate build options, computes and stores offsets for i/o buffers for child and parent nodes, and allocates the necessary on-device buffers. It then calls load_kernels().

The load_kernels() method reads the kernels specified for a given model subclass from the appropriate files, loads the kernels into the OpenCL context, and builds them.

The collect_input() method populates the feedforward and feedback input buffers from child and parent nodes respectively, keeping all operations in GPU memory. Note that if double-buffering similar to that in CNS is desired, the user must include it in the internal buffer structure of his or her model. The author did not wish to apply this procedure by default due to the resulting increase in memory requirements.

The execute() method is defined by the subclass, but should always start by calling collect_input().

On a different note, a generic "InputModel" class is also included to assist in reading vector input data into the network from an external source. These could include images, pre-processed feature activation vectors, video frames, audio spectrograms, datasets from which one is trying to learn some structure, etc.

11. The Layout Manager

While all nodes can specify their connectivity patterns independently, the layout manager includes Network and NetworkRegion classes for defining the higher-level structure of the network.

A Network is the top-level abstraction for the entire network of nodes. It contains a list of NetworkRegion instances, and is the appropriate location for functionality involving e.g. network status monitoring, performance monitoring, etc. It is designed to permit inheritance by specific models for model-implementation-dependent features.

A NetworkRegion is the atomic unit of execution in Phoenix. All Model instances within a given NetworkRegion instance will be executed between barrier applications on the GPU, so no timing guarantees within a NetworkRegion can be provided and NetworkRegion execution should be considered all-or-nothing. The NetworkRegion abstraction is the most fundamental contributing factor to the ability of cortical models in Phoenix to be executed efficiently on GPUs.

A NodeTypeDef is a class used to represent the class identities and i/o configurations of nodes. This could be used to facilitate the construction of

networks with many copies of the same types of nodes without needing to include routines to configure them within the network-building code.

12. The Execution Manager

The execution manager is responsible for executing models in Phoenix on the GPU (or, hypothetically, any other OpenCL compute device including the CPU).

Execution order for a given model is specified with a list of references to regions in the order in which they should be executed. The execution manager then simply loops through the list, executing each region in turn and separating them with OpenCL barriers to ensure synchronization. The execution of a region simply involves executing each individual node within that region, using the node execute() method discussed above.

Because model computation is performed on the GPU, there is enough latency between successive waves of node executions for the execution manager to queue up additional work to follow. This is arguably one of the central benefits of PyOpenCL: because of this latency and PyOpenCL's own implementation optimizations, such management can be done in a convenient scripting language with no performance penalty whatsoever.

13. Other Utilities

A device information script is included to simplify OpenCL configuration and any related troubleshooting. This script can also be helpful in better understanding device capabilities.

A few examples of OpenCL kernels and the use of some of the more basic abstractions in PyOpenCL are included as well.

14. Potential Framework Extensions

Support for multiple GPUs has not yet been implemented, but is planned for the near future and should not prove too challenging. Support for distributed clusters is another possibility, although the author notes that interconnect bandwidth and latency could begin to impose nontrivial constraints on network design.

With sufficient metaprogramming effort, some of the contents of the node __init__() method could potentially be hard-coded into the kernels associated with a given algorithm. The author has not yet fully investigated this possibility, but it could perhaps yield performance improvements.

In general, further convenience methods could be included (e.g. an optional generic network factory that could be used if desired to build a network given a parametric representation of its connectivity structure and node class composition, perhaps a generic double-buffering implementation if feasible without violating design principles), as could additional examples. We anticipate developing some of these things over the course of our own subsequent

work with the framework, since this will show us which of them might be most useful in pursuing it.

Finally, it has recently come to the attention of the author that the present implementation of Phoenix does not permit lateral connectivity within a region. Although very few cortical models include direct lateral connectivity, it is known to be present in the cerebral cortex. The author hopes to remedy this matter within the coming weeks.

Part 4. Conclusion

"In this City's center, a thing grows, an auto-catalytic tree, in almost-life, feeding through the roots of thought on the rich decay of its own shed images, and ramifying, through myriad lightning-branches, up, up, toward the hidden light of vision,"

– Gibson and Sterling, The Difference Engine

References

[1] M.A. Arbib. *The handbook of brain theory and neural networks*. MIT press, 2002.

[2] H.E. Atallah, M.J. Frank, and R.C. O'Reilly. Hippocampus, cortex, and basal ganglia: Insights from computational models of complementary learning systems. *Neurobiology of Learning and Memory*, 82(3):253–267, 2004.

[3] D. Badre and M. D'Esposito. Is the rostro-caudal axis of the frontal lobe hierarchical? *Nature Reviews Neuroscience*, 10(9):659–669, 2009.

[4] S. Behnke and R. Rojas. Neural abstraction pyramid: A hierarchical image understanding architecture. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 2, pages 820–825. IEEE, 1998.

[5] T.E.J. Behrens and G. Jocham. How to perfect a chocolate soufflé and other important problems. *Neuron*, 71(2):203–205, 2011.

[6] M. Berniker, A. Jarc, E. Bizzi, and M.C. Tresch. Simplified and effective motor control based on muscle synergies to exploit musculoskeletal dynamics. *Proceedings of the National Academy of Sciences*, 106(18):7601–7606, 2009.

[7] M.M. Botvinick. Hierarchical reinforcement learning and decision making. *Current Opinion in Neurobiology*, 2012.

[8] M.M. Botvinick, Y. Niv, and A.C. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

[9] D.P. Buxhoeveden and M.F. Casanova. The minicolumn hypothesis in neuroscience. *Brain*, 125(5):935–951, 2002.

[10] G.A. Carpenter and S. Grossberg. Adaptive resonance theory. *CAS/CNS Technical Report Series*, (008), 2010.

[11] E. Ciaramelli, C.L. Grady, and M. Moscovitch. Top-down and bottom-up attention to memory: a hypothesis (atom) on the role of the posterior parietal cortex in memory retrieval. *Neuropsychologia*, 2008.

[12] A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience*, 6(3):300–308, 2003.

[13] J.J. DiCarlo, D. Zoccolan, and N.C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[14] S.J. Dickinson, A. Leonardis, B. Schiele, and M.J. Tarr. *Object categorization: computer and human vision perspectives*. Cambridge University Press, 2009.

[15] A. Fazl, S. Grossberg, and E. Mingolla. View-invariant object category learning, recognition, and search: How spatial and object attention are coordinated using surface-based attentional shrouds. *CAS/CNS Technical Report Series*, (011), 2010.

[16] C.J. Fiebach and R.I. Schubotz. Dynamic anticipatory processing of hierarchical sequential events: a common role for broca's area and ventral premotor cortex across domains? *Cortex*, 42(4):499–502, 2006.

[17] N.C. Foley, S. Grossberg, and E. Mingolla. Neural dynamics of object-based multifocal visual spatial attention and priming: Object cueing, useful-field-of-view, and crowding. *Cognitive psychology*, 65(1):77–117, 2012.

[18] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[19] K. Fukushima. Analysis of the process of visual pattern recognition by the neocognitron. *Neural Networks*, 2(6):413–420, 1989.

[20] K. Fukushima. Neocognitron. *Scholarpedia*, 2(1):1717, 2007.

[21] K. Fukushima. Increasing robustness against background noise: Visual pattern recognition by a neocognitron. *Neural networks*, 24(7):767–778, 2011.

[22] D. George. *How the brain might work: A hierarchical and temporal model for learning and recognition*. PhD thesis, Stanford University, 2008.

[23] L.S. Glezer, X. Jiang, and M. Riesenhuber. Evidence for highly selective neuronal tuning to whole words in the "visual word form area". *Neuron*, 62(2):199–204, 2009.

[24] K. Grill-Spector and N. Witthoft. Deos the bairn not raed ervey lteter by istlef, but the wrod as a wlohe? *Neuron*, 62(2):161, 2009.

[25] J. Hawkins, S. Ahmad, and D. Dubinsky. Hierarchical temporal memory including htm cortical learning algorithms. *Techical report, Numenta, Inc, Palto Alto http://www. numenta. com/htmoverview/education/HTM_ CorticalLearningAlgorithms. pdf*, 2010.

[26] TE Hazy, MJ Frank, RC O? Reilly, et al. Banishing the homunculus: making working memory work. *Neuroscience*, 139(1):105–118, 2006.

[27] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[28] M.R. Hunsaker, B. Lee, and R.P. Kesner. Evaluating the temporal context of episodic memory: The role of ca3 and ca1. *Behavioural brain research*, 188(2):310–315, 2008.

[29] T. Kiritani, I.R. Wickersham, H.S. Seung, and G.M.G. Shepherd. Hierarchical connectivity and connection-specific dynamics in the corticospinal–corticostriatal microcircuit in mouse motor cortex. *The Journal of Neuroscience*, 32(14):4992–5001, 2012.

[30] M.E. Large and P.A. McMullen. Hierarchical attention in discriminating objects at different levels of specificity. *Attention, Perception, & Psychophysics*, 68(5):845–860, 2006.

[31] H. Lee, C. Ekanadham, and A. Ng. Sparse deep belief net model for visual area v2. *Advances in neural information processing systems*, 20:873–880, 2008.

[32] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[33] T.S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *JOSA A*, 20(7):1434–1448, 2003.

[34] S. Lehar. Gestalt isomorphism and the primacy of subjective conscious experience: A gestalt bubble model. *Behavioral and Brain Sciences*, 26(4):375–408, 2003.

[35] J.E. Lisman. Relating hippocampal circuitry viewpoint to function: Recall of memory sequences by reciprocal dentate–ca3 interactions. *Neuron*, 22:233–242, 1999.

[36] J.W. Miller and P.H. Lommel. Biomimetic sensory abstraction using hierarchical quilted self-organizing maps. In *Optics East 2006*, pages 63840A–63840A. International Society for Optics and Photonics, 2006.

[37] E.I. Moser and M.B. Moser. One-shot memory in hippocampal ca3 networks. *Neuron*, 38(2):147–148, 2003.

[38] V.B. Mountcastle. The columnar organization of the neocortex. *Brain*, 120(4):701–722, 1997.

[39] A. Nere, S. Franey, A. Hashmi, and M. Lipasti. Simulating cortical networks on heterogeneous multi-gpu systems. *Journal of Parallel and Distributed Computing*, 2012.

[40] A. Nere and M. Lipasti. Cortical architectures on a gpgpu. In *Proceedings of the 3rd workshop on general-purpose computation on graphics processing units*, pages 12–18. ACM, 2010.

[41] S.A. Overduin, A.G. Richardson, and E. Bizzi. Cortical processing during dynamic motor adaptation. *Progress in Motor Control*, pages 423–438, 2009.

[42] S.A. Overduin, A.G. Richardson, E. Bizzi, and D.Z. Press. Simultaneous sensorimotor adaptation and sequence learning. *Experimental Brain Research*, 184(3):451–456, 2008.

[43] R. Pinto and P. Engel. Loopsom: a robust som variant using self-organizing temporal feedback connections. In *Proceedings of the 7th ENIA-BRAZILIAN meeting on Artificial Intelligence*, 2009.

[44] T. Poggio. The computational magic of the ventral stream. 2011.

[45] T. Poggio. The levels of understanding framework, revised. 2012.

[46] T. Poggio, U. Knoblich, and J. Mutch. Cns: a gpu-based framework for simulating cortically-organized networks. 2010.

[47] C. Ranganath. Binding items and contexts the cognitive neuroscience of episodic memory. *Current Directions in Psychological Science*, 19(3):131–137, 2010.

[48] C. Ranganath. A unified framework for the functional organization of the medial temporal lobes and the phenomenology of episodic memory. *Hippocampus*, 20(11):1263–1290, 2010.

[49] J.R. Reynolds and R.C. O'Reilly. Developing pfc representations using reinforcement learning. *Cognition*, 113(3):281–292, 2009.

[50] J.J.F. Ribas-Fernandes, A. Solway, C. Diuk, J.T. McGuire, A.G. Barto, Y. Niv, and M.M. Botvinick. A neural signature of hierarchical reinforcement learning. *Neuron*, 71(2):370–379, 2011.

[51] M. Riesenhuber. Object recognition in cortex: Neural mechanisms, and possible roles for attention. *Neurobiology of attention*, pages 279–287, 2005.

[52] M. Riesenhuber. Appearance isn't everything: news on object representation in cortex. *Neuron*, 55(3):341–344, 2007.

[53] M. Riesenhuber and T. Poggio. Are cortical models really bound by the "binding problem"? *Neuron*, 24:87–93, 1999.

[54] M. Riesenhuber and T. Poggio. A note on object class representation and categorical perception. 1999.

[55] M. Riesenhuber and T. Poggio. The individual is nothing, the class everything: Psychophysics and modeling of recognition in obect classes. 2000.

[56] M. Riesenhuber, T. Poggio, et al. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2:1019–1025, 1999.

[57] U. Rokni, A.G. Richardson, E. Bizzi, and H.S. Seung. Motor learning with unstable neural representations. *Neuron*, 54(4):653–666, 2007.

[58] E.T. Rolls. A computational theory of episodic memory formation in the hippocampus. *Behavioural brain research*, 215(2):180, 2010.

[59] R.I. Schubotz and C.J. Fiebach. *Integrative Models of Broca's Area and the Ventral Premotor Cortex*. Elsevier srl, 2006.

[60] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio. A theory of object recognition: computations and circuits in the feedforward path of the ventral stream in primate visual cortex. Technical report, DTIC Document, 2005.

[61] T. Serre, L. Wolf, and T. Poggio. A new biologically motivated framework for robust object recognition. 2004.

[62] C. Sestieri, M. Corbetta, G.L. Romani, and G.L. Shulman. Episodic memory retrieval, parietal cortex, and the default mode network: functional and topographic analyses. *The Journal of Neuroscience*, 31(12):4407–4420, 2011.

[63] C. Tan, J. Leibo, and T. Poggio. Throwing down the visual intelligence gauntlet. *Machine Learning for Computer Vision*, pages 1–15, 2012.

[64] J. Tani and S. Nolfi. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12(7):1131–1141, 1999.

[65] H.T. van Schie, I. Toni, and H. Bekkering. Special issue: Position paper comparable mechanisms for action and language: Neural systems behind intentions, goals, and means. *Cortex*, 42:495–498, 2006.

[66] D. Walther, L. Itti, M. Riesenhuber, T. Poggio, and C. Koch. Attentional selection for object recognition–a gentle way. In *Biologically Motivated Computer Vision*, pages 251–267. Springer, 2002.

[67] D.B. Walther and C. Koch. Attention in hierarchical models of object recognition. *Progress in brain research*, 165:57–78, 2007.

[68] C.R.E. Wilson, D. Gaffan, P.G.F. Browning, and M.G. Baxter. Functional localization within the prefrontal cortex: missing the forest for the trees? *Trends in neurosciences*, 33(12):533–540, 2010.