AFRL-RI-RS-TR-2019-004

THE CORTICAL GRAPH (CORGRAPH) PROJECT – AN INVESTIGATION INTO THE REPRESENTATION OF COMPLEX CONTEXTUAL RELATIONSHIPS BASED ON THE STRUCTURE OF CEREBRAL CORTEX

PORTLAND STATE UNIVERSITY

JANUARY 2019

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2019-004 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
MARK BARNELL
Work Unit Manager

/ S /
JOSEPH CAROLI
Branch Chief, High Performance
Systems Branch
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JANUARY 2019	FINAL TECHNICAL REPORT	APR 2017 – JUN 2018

4. TITLE AND SUBTITLE

THE CORTICAL GRAPH (CORGRAPH) PROJECT – AN INVESTIGATION INTO THE REPRESENTATION OF COMPLEX CONTEXTUAL RELATIONSHIPS BASED ON THE STRUCTURE OF CEREBRAL CORTEX

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-17-1-0099

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Dan Hammerstrom

5d. PROJECT NUMBER
T2BS

5e. TASK NUMBER
PO

5f. WORK UNIT NUMBER
RT

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Portland State University
1825 SW Broadway
Portland, OR 97201

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2019-004

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT
A key element of any kind of intelligent computing is the management and recognition of sensor data. And although applications are being studied for a range of sensor modalities, the most complicated and perhaps most pervasive is that of analyzing and understanding moving images. The purpose of the research described here is to explore neural inspired algorithms, loosely based on cortical structures, as new approaches to capturing, representing, and then performing inference over "higher order" structure of features in a still or moving image. The approach is to study the use of hierarchical, modular, sparsely distributed structures loosely based on the HTM family of algorithms.

15. SUBJECT TERMS
Video Image Processing, Machine Learning, Cortical Algorithms, Graph Models, Graph Networks, Computational Neuroscience

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**MARK BARNELL**
U	U	U	UU	20	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

Table of Contents

List of Figures

1 SUMMARY

A key element of any kind of intelligent computing is the management and recognition of sensor data. Although applications are being studied for a range of sensor modalities, the most complicated and perhaps most pervasive is that of analyzing and understanding moving-image (video) data. The pervasiveness of embedded processors and image sensors is resulting in significant research in this space. This is true in both the civilian and government sectors. While much of the past growth in imaging applications is based on the adoption of low-cost CMOS image sensors in smartphones and digital cameras, that growth is now spreading as embedded digital imaging from automotive, to security and medical applications, and computing at the edge in the Internet of Things (IoT).

Interpreting streams of data requires capturing multiple, hierarchical levels of information from the raw pixels of a video. These levels allow the use of structure and abstraction in processing the data. "Contextual information" generally leverages such structure. Processing these data typically occurs in a pipeline. At the front of the pipeline one finds basic per pixel operations such as filtering and smoothing, color correction, perspective transformation and some motion compensation (for vibrating platforms). Further into the pipeline, features and features of features are extracted, usually in some kind of translation and scale invariant manner. As one nears the end of the pipeline, the various features come together to form objects and then groups of objects create scenarios. A visual object is defined by the features and their relative (translation and scale invariant) positions with respect to each other.

Finding objects by understanding the features of the object and their spatial and temporal relationships is a complex and compute intensive task. And although algorithms exist, there are not many effective approaches. Generally, these relationships can be captured in a graph. The objects in the image are represented as graphs, with object features being the nodes in the graph, and the edges between nodes representing the spatial information on the features' relationships to each other. The recognition task is to find an inexact subgraph isomorphism of known objects with respect to the observed features. This is a probabilistic task since there is rarely an exact match, what you seek is a best match in a Bayesian most likely sense.

Because algorithms to create and then do inference over such graphs are compute intensive, few vision systems actually do this. Instead a "bag of features" approach is used. Although used extensively (until the advent of Deep Learning), it is not particularly accurate. The entire approach becomes even more complex when the recognition of moving objects (both in position and pose) is required. In fact, what we are really interested in is the object's relationship to other objects in the image, leading to the recognition of complex scenarios.

Currently the deep network approaches based on incremental, gradient descent and distributed representations, are the most effective approach to recognizing objects in images. However, they have a number of shortcomings as they still struggle with moving images and with scenario recognition. And even for simple still image recognition they can be stymied, "hacked" even. They break in unusual and non-biologically plausible ways.

The purpose of the research described here is to explore neural inspired algorithms, loosely based on cortical structures, as new approaches to capturing and then representing the graphical structure of features in a still or moving image. The approach is to study the use of hierarchical, modular, sparsely distributed structures loosely based on the HTM family of algorithms. There does not seem to be any research in using bioinspired algorithms to solve this problem.

2 INTRODUCTION

A key element of intelligent computing is the management and recognition of sensor data. And although applications are being studied for a range of sensor modalities, the most complicated and perhaps most pervasive is that of analyzing and understanding moving-image (video) data. Interpreting streams of data requires capturing multiple, hierarchical levels of information from raw pixels. Processing these data typically occurs in a pipeline. At the front of the pipeline one finds basic per pixel operations such as filtering and smoothing, color correction, perspective transformation and some motion compensation (for vibrating platforms). Further into the pipeline features and features of features are extracted, usually in some kind of translation and scale invariant manner. Finally, near the end of the pipeline, the various features come together to form objects. A visual object is defined by the features and their relative (translation and scale invariant) positions with respect to each other. One can also extend the ideas to defining a scenario or "context" as the set of objects in some kind of spatial and temporal relationships with each other.

Finding objects by understanding the features of the object and their spatial and temporal relationships is a complex and compute intensive task. And although algorithms exist, they are few and far between. Generally, these relationships can be captured in a graph. The objects in the image are represented as graphs, with object features being the nodes in the graph, and the edges between nodes representing the spatial information about the features' relationships to each other. In recognizing a scenario or context, the relationships would then be between objects. The recognition task then is to find the inexact subgraph isomorphism of known objects with respect to the observed features. This is a probabilistic task since there is rarely an exact match, what we seek is a best match in a Bayesian most likely sense.

Because object recognition using graphs is so compute intensive, few vision systems actually do this. Instead a "bag of features" approach is used, where multiple independent features are extracted, along with as much information in the immediate vicinity around the feature. This information is "normalized" leading to position, scale, and orientation invariant features. However, the result is somewhat inaccurate object recognition. And, the entire approach becomes even more complex and compute bound when the recognition of multiple moving objects is required, since what we are really interested in is the object's relationship to other objects in the image, leading to an understanding of the situation in the image for the recognition of complex scenarios or context.

Currently, the deep network approaches, based on incremental, gradient descent, distributed representations, have proven to be the most effective approach to recognizing objects in images. However, they have a number of short comings, not the least of which is that they do not recognize images in any way similar to biological approaches. Consequently, they still struggle with moving images and with more complex scenario recognition. They are also easily fooled and break easily in non-biological ways.

Graphs are a natural representation of knowledge, which, in general, is hierarchical and has sparsely connected elements. So it is to be expected that graph techniques could be useful for these kinds of problems. Unfortunately, doing various kinds of operations over graphs, such as approximate subgraph isomorphism determination, are very compute intensive. It has been shown that accurately capturing a graph is an NP-Hard problem and then inference over the graph is also NP-Hard. Yet, biological systems appear to do this seemingly effortlessly, leading to the question as to, how?

It is generally believed that among other things, cognition involves accessing and performing inference over complex, structured knowledge. The Oxford English Dictionary defines

"cognition" as "the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses." So being able to capture hierarchical knowledge, represent it internally and then do inference over those representations may not be sufficient, but they are certainly necessary to computational intelligence.

There is a nice description of the problem in, "Relational inductive biases, deep learning, and graph networks" [1]:

> *A key signature of human intelligence is the ability to make use of finite means, in which a small set of elements (such as words) can be productively composed in limitless ways (such as into new sentences). This reflects the principle of combinatorial generalization, that is, constructing new inferences, predictions, and behaviors from known building blocks. Here we explore how to improve modern AI's capacity for combinatorial generalization by biasing learning towards structured representations and computations, and in particular, systems that operate on graphs.*
>
> *Despite deep learning's successes, however, important critiques … have highlighted key challenges it faces in complex language and scene understanding, reasoning about structured data, transferring learning beyond the training conditions, and learning from small amounts of experience. These challenges demand combinatorial generalization, and so it is perhaps not surprising that an approach which eschews compositionality and explicit structure struggles to meet them.*
>
> *We define structure as the product of composing a set of known building blocks. "Structured representations" capture this composition (i.e., the arrangement of the elements) and structured computations operate over the elements and their composition as a whole. Relational reasoning, then, involves manipulating structured representations of entities and relations, using rules for how they can be composed. We use these terms to capture notions from cognitive science, theoretical computer science, and AI.*

The fundamental objective of the research supported by this grant is to pursue biologically inspired algorithms for capturing, representing, and performing inference over structured data. What about these techniques makes them attractive for the complex approximate graph matching that is being proposed here? The primary motivation is the notion of representing data in a hierarchical, sparse distributed fashion and then being able to do a probabilistic best match of the image being input to stored internal patterns; this means inferring the most likely mapping of a graphs of learned objects to some arbitrary subgraph of the visual image. This kind of inference is done quickly and in a massively parallel manner. Furthermore, these networks can be configured in connection limited, modular, hierarchical configurations that are extremely efficient pattern matchers and can be emulated efficiently by specialized hardware architectures.

Through the year we spent quite a bit of time trying to understand graphical structures and how cortical circuits might implement them. As a result we departed from the original Task Table for the grant (see Table 1). We followed the same basic flow, but found that creating cortical structures to represent and then do subgraph isomorphisms was more complex and we were unable to find a simple "cortical" like representation that solved the problem. As mentioned, working with graphs is computationally intensive. We knew there were potential problems, all of which ended up creating difficulties for us.

The bottom line is that it is unlikely that these systems are doing subgraph matching over arbitrarily complex graphs, but rather are representing objects and doing subgraph match over very

limited, mostly likely planar, hierarchical graphs. Furthermore, they seem to be learning and doing inference over "graphical fragments."

3 PROJECT EFFORTS

3.1 Methods, Assumptions and Procedures

During the first year we primarily explored several existing approaches to representing structure. We found that Hinton's Capsule networks and Numenta's HTM/CLA were capturing and representing some structured information and, in the case of Numenta's models, using sparse, distributed representations. We also developed a tool for taking a graph and breaking it down into sparse distributed representation ("sparsification"). The tool generates input vectors (training and test) from simple 2D graphics images of simple block structures.

The general strategy of this effort is to start with simple problems that can leverage graph methods and simple cortical like algorithms to represent structure in data, incrementally adding complexity until we reach the point where we can, with reasonable confidence, identify a path forward towards a general solution to the problem. Each step in the process as described below was mostly conceptual with some informal simulation to validate and demonstrate the concepts. Theory will only take us so far, so there will be an increasing reliance on algorithm implementation and experimentation.

In the original proposal, we proposed that the first two quarters of work would focus on developing a more thorough understanding of the role that graphs and graph algorithms play in computer vision applications, and building a reasonable model of cortex as a foundation in which to develop graph algorithms. Initially the model should be simple enough to understand and use, without too much arbitrary parameterization.

Although there is some work in using graphs to represent complex patterns and leverage graph isomorphism in pattern recognition, the idea has had only limited application in Computer Vision. In fact, most image processing and computer vision books spend very little time on graphical models. One of the most widely used books on Computer Vision, *Computer Vision Algorithms and Applications*, R. Szeliski [2], devotes 2 pages to graph based image segmentation. There is some literature on the topic, but even there, the number of papers is small. This is probably due to the fact that, though very descriptive of the objects to be recognized, using graphs is too compute intensive and the various methods difficult to use. This is particularly true when trying to do probabilistic "most likely" approximate matching. A key task then is to study the state of the art and understand the strengths and weaknesses of these models in existing computer vision environments.

The subgraph isomorphism problem is a computational task in which two graphs *A* and *B* are given as input, and one must determine whether *A* contains a subgraph that is isomorphic to *B*. One graph could be developed from a large image, where the graph represents all relationships in the image, possibly representing several objects, not just the object in question. Subgraph isomorphism is then performed to find a subgraph of the total image graph that represents any objects we want to identify. Subgraph isomorphism is a generalization of both the maximum clique problem and the problem of testing whether a graph contains a Hamiltonian cycle, and is therefore NP-complete. However certain constrained examples of subgraph isomorphism, such as over 2D planar graphs, are solvable in polynomial time. As mentioned, it is most likely that cortical implementations do not represent anything close to a real graph.

The other question is whether hierarchies actually ease the computational load of subgraph identification. Unfortunately, there appears to be no research on that particular topic. Likewise there does not seem to be any research on capturing, representing and analyzing graphs based on sparse distributed data representations.

For this project, our initial focus was on the first five objectives in the Statement of Work:

4.0 Statement of Work

- 4.1 Study Graphical Models in Vision: This task involves a literature search and the production of a report. COMPLETED: The status reports, including this final report represent the documentation of our intensive literature search.
- 4.2 Develop Basic Cortical Models: Based on the literature search a set of model characteristics of abstract cortical models, this task will result in a report. NOT COMPLETED: This was not done. After looking at dozens of models, we are unable to reach a consensus on what characteristics the model needs. This work is on-going.
- 4.3 Study Graph Isomorphism: The subgraph isomorphism problem is a computational task in which two graphs *A* and *B* are given as input, and one must determine whether *A* contains a subgraph that is isomorphic to *B*. In this task, the state of the art in graph isomorphisms were studied, the result is a report on the best and most capable algorithms. COMPLETED: Aakanksha Murthia developed a very nice subgraph isomorphism technique based on the Ullman sub-graph isomorphism detection algorithm [3].
- 4.4 Develop test data: Networks will require input data to "train" and "test" the algorithms. The data will involve images from a simple 2D (not 3D) blocks world. Such images demonstrate structure in a simplified manner, which will be easier to capture. The algorithms that are developed will be tested with the 2D data. NOT COMPLETED: Mark Musil developed an initial 2D blocks world generator. However, it did not give us enough flexibility in generating composite objects, multiple simple objects combined to generate a more complex object. It was a good exercise, we are now developing a more advanced version to generate composite Block's World objects. These data will use Aakanksha's graph sparsifier to generate input to the cortical models we are building.
- 4.5 Simple graph isomorphism I – Index Dependence: In this task a cortical like algorithm for doing simple graph isomorphism will be implemented in Matlab and a set of experiments performed, which will be summarized in a report. This work will continue in the second year. NOT COMPLETED.

Table 1Tasks in SOW

Task	Year 1			
	Q1	Q2	Q3	Q4
4.1 Graphical Models for Vision				
4.2 Develop Basic Cortical Models				
4.3 Graph Isomorphism				
4.4 Test Data				
4.5 Simple Graph Isomorphism I				
4.6 Simple Graph Isomorphism II				

3.1.1 Graph Representation in Neural Networks

Assume each node in a graph is a feature and is represented in some way by the neural model. Furthermore, the nodes have relationships (edges) which are in turn represented by correlations between the nodes. Data representation can then be thought of as being in a spectrum from completely local to fully distributed, with many possibilities in between. At one end is a totally localized representation, where each node represents a data element. Most modern database systems fit this model. At the other end is the fully distributed representation, where all nodes participate in all representations. This is typical of most machine learning systems, though primarily at the "fully connected" output. Local representations can represent lots of things simultaneously. Fully distributed representations can only represent one thing at a time.

In the middle are sparse distributed representations (SDR). They spread over a few nodes, but are only partially activated. Although not to the same degree as a fully localized representation, some simultaneous representations are possible.

The first issue concerns the notion of the distribution of representations. So the fundamental question is how distributed is distributed? The second question as to how sparse, first depends on how distributed. For most distributed representations, we can generally determine the degree of sparseness.

- In a fully localized representation each node is a complete representation of a feature. Learning is fast and on-line; storage capacity and ability to generalize are poor.
- In a fully distributed representation each node participates in each feature. Learning is very slow and off-line; storage capacity and ability to generalize are good. Since there are many parameters large training sets are required.
- In a sparsely distributed representation each node participates in a subset (typically small) number of representations. Learning is fast and on-line; storage capacity and ability to generalize are very good.

A number of people have shown that cortical structures most likely implement SDRs (though how sparse is still an open question). A good description of SDRs is the Numenta paper,

"Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory" [4]. Sparse distributed representations are obviously the way to go, but how sparse and how distributed should they be?

If we are representing a large complex graph, even with sparse distributed representations we cannot have all nodes present at once in each representation. So how do we train a network on a graph? On subsets of the graph? Which subsets, those which are connected? And how are feature relationships (graph edges) represented? Should we describe them via a sequential traversal of the graph, either single nodes or groups of connected nodes?

For example, how does one learn the graph to the right? With localized nodes it is easy, you can present in the input any arbitrary combination of the nodes, then build edge correlations by presenting vectors of connected nodes to the network. However, with a fully distributed representation then you have to have some kind of temporal sequencing, since you can only represent one node at a time on the input.

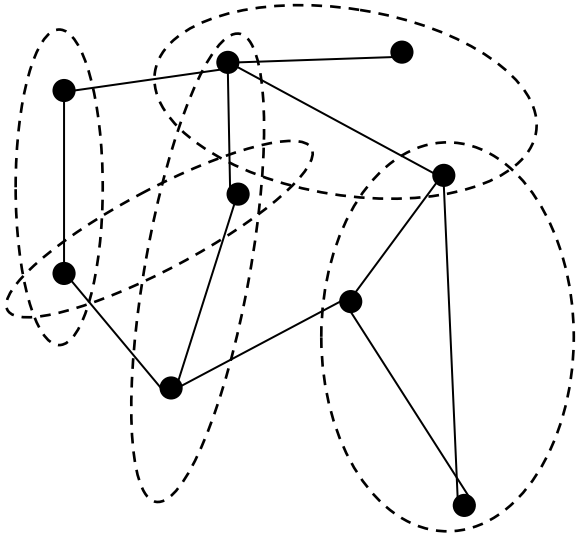

Figure 1 Sparse Graph Partition

Since most mammals use "active vision" where they recognize an object by saccading to salient locations, such a sequential input of small subgraphs may actually be value.

In a sparsely distributed representation the numbers of nodes which are represented simultaneously depends on how many nodes you can fit at once in the input, which is determined by how distributed they are. Do you use a few distinct regions and then just a few features per region, OR, do you apply a union of the connected nodes in subsets of the graph (to the right)? How many representations does a single node participate in? We are looking to answer these questions in the second year of the grant.

We looked at other efforts to represent structure in machine learning applications. In Fei Fei Li's work at Stanford, they use a localized representation of the nodes, but distributed representations from a Deep Learning network to learn the relationships between the nodes.

However, there are two very promising approaches that we spent quite a bit of time studying in detail in the first year: 1) Hinton Capsule Networks, and 2) Numenta's 2D HTM Cortical model. These are now discussed in a little more detail.

3.1.2 Capsule Networks

There is not much information available on Hinton's work, the only real publication is [5]. According to the paper, "A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or object part."

As Geoff might say, capsules represent an attempt to undo the disastrous effects of CNN spatial pooling, which loses position relative information ("topological structure"). In a CNN, the internal data representation of a convolutional neural network does not take into account important spatial hierarchies between simple and complex objects. Max pooling throws away information that is important, and also does not encode relative spatial relationships between features. A number of typical "deep net" failures are due to these effects, where location information is lost and the network is looking for raw features in a "bag of features" approach.

The primary capsules are the lowest level of multi-dimensional entities and, from an inverse graphics perspective, activating the primary capsules corresponds to inverting the rendering process. Capsules are hierarchical so that increasingly complex object representations are possible.

Hinton argues that in order to correctly do classification and object recognition, it is important to preserve hierarchical pose relationships between object parts. This is the key intuition to help understand why capsule theory is important. It incorporates relative relationships between objects and it is represented numerically as a 4D pose matrix. Resulting in position and rotation invariant object recognition. When these relationships are built into a somewhat localized internal representation of data, it becomes very easy for a model to understand that the thing that it sees is just another "pose" of something that it has seen before.

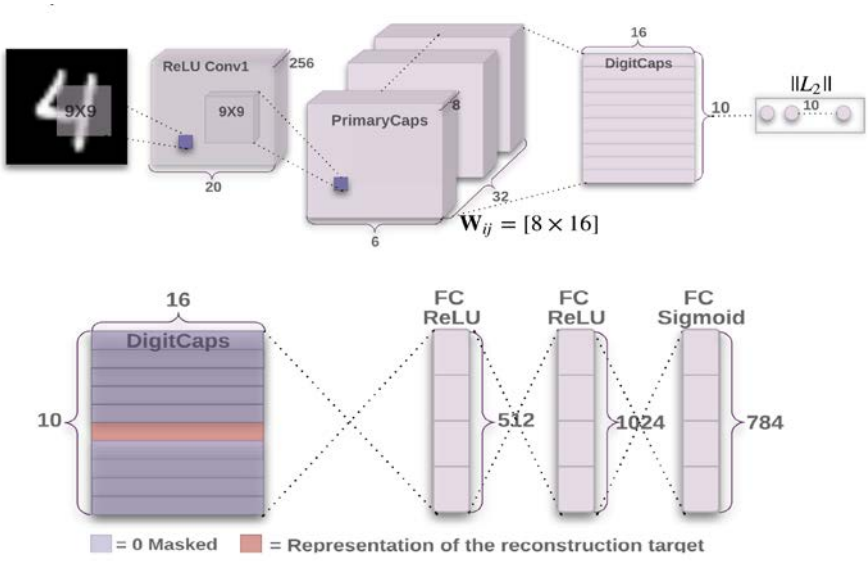

Figure 2 A Capsule Network

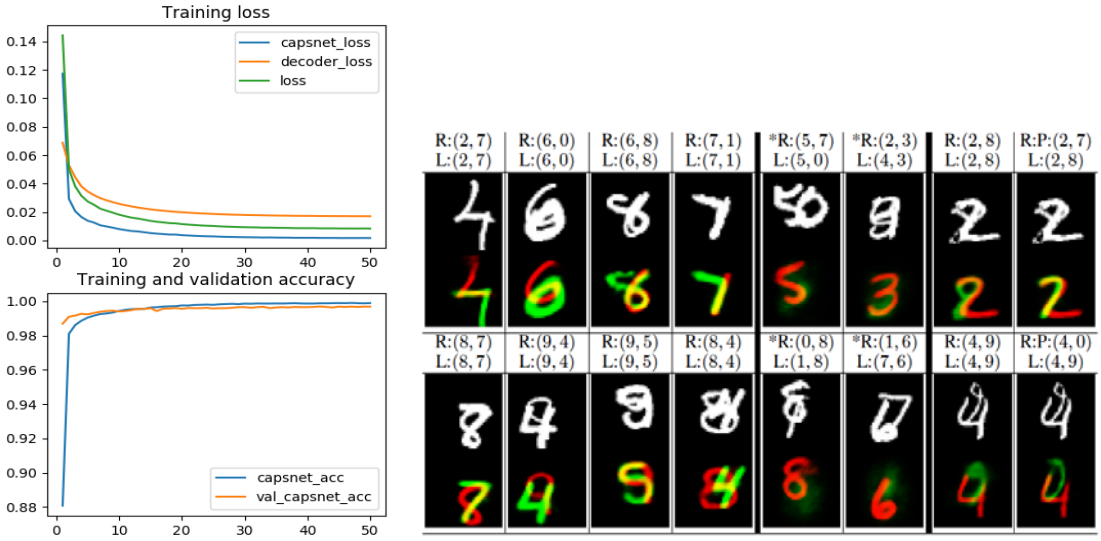

Figure 3 Capsule Network Performance

Each capsule learns to recognize an implicitly defined visual entity over a limited domain of viewing conditions, deformations and poses, and it outputs both the probability that the entity is present within its limited domain and a set of "instantiation parameters" that may include the precise pose, lighting and deformation of the visual entity relative to an implicitly defined canonical version of that entity. A Capsule is a group of neurons (short for "encapsulate"):

- Output vectors - Connections pass vectors (Tensors) rather than scalar values;
- Vector length: probability the object represented by the capsule is in the visual field;
- Vector orientation represents the object being viewed;
- Each connection's weight is a matrix, which becomes an affine transformation, trainable for feature positions;
- Matrix multiplication, scalar weighting and summation of input vectors;
- Vector-to-vector nonlinearity (squishing function); and,
- Routing-by-agreement: when multiple predictions agree, a higher level capsule becomes active. In turn, a lower level capsule will send its input to the higher level capsule that "agrees" with its input.

What we get is the predicted position of the higher level feature as a vector, which represents where the face should be according to the detected position of the eyes. This is a clever algorithm that addresses some of the shortcomings of Deep Learning methods. But with respect to what we are trying to do, there are important questions to be answered. Can it be extended to more complex, abstract kinds of relationships? It is still gradient descent and requires lots of training data and many, many training runs. So an important question is, is it scalable?

Sophie created an implementation of a Capsule network in TensorFlow and used on the CIFAR-10 data set. The training times were literally days. It did not increase accuracy and it was extremely compute intensive.

Capsule networks somewhat solve our problem, but results are subject to combinatorial explosion. And they are, for the most part, a localized representation. While having the Deep Learning community developing these ideas is encouraging, but after extensive simulation, we do not feel that this particular model helps us very much.

3.1.3 Numenta

The other important work is that done by Jeff Hawkin and others at his company, Numenta. They have been developing a number of theories and ideas and are starting to publish these results. There are several quality papers on their web page: https://numenta.com/papers/

The one that is most relevant to us is "A Theory of How Columns in the Neocortex Enable Learning the Structure of the World" [6].

Their cortical model is organized into a 2D sheet of columns. Each column has two layers, simplified from 6 layers to 2, a single 2D array forms a level. Each level has an input layer (similar to cortical layer IV) and an output layer (similar to cortical layers II/III). The prevalence of this two-layer connection motif suggests it plays an essential role in cortical processing. Inputs to a level are defined as Local Receptive Fields (LRF) for each column of a lower input level, also a 2D array. Columns have local neighborhood inhibitory connections to strengthen strong inputs and attenuate weak inputs, leading to Sparse Distributed Representations. There are also long range excitatory connections which cause columns to have some correlation with selected neighbors. Columns have sparse localized spatial pooling and multi-dendritic temporal pooling.

Each column integrates its changing input over time to learn predictive models of observed objects. Excitatory lateral connections across columns allow the network to more rapidly infer objects based on the partial knowledge of adjacent columns. Excitatory connections tend to be longer range (though how far is an open issue).

According to Numenta, because columns integrate input over time and space, the network learns models of complex objects that extend well beyond the receptive field of individual cells. A single cortical column can learn models of complete objects via a temporal presentation of objects.

An important development in their model is the presence in each column of a signal representing location. The location signal is "allocentric", meaning it is a location relative to the object being sensed. It is not clear how and where this information is computed. The Hippocampus has been shown to implement "place" cells. And parts of Entorhinal cortex implement a more abstract "grid" functionality [7]. However, for finding an object with multiple features such computations would have to be done quickly and more local to the area of cortext that is actually "recognizing" the object. Numenta claims to have solved this problem and promises a paper in the Fall with their proposed circuitry. One hypothesis appeared in a poster at COSYN 2018. The proposed model had a grid for the object, and another grid for the component of the object in question. The grids compute the physical transformation from the egocentric view to the allocentric view of the object. They then use both representations to compute the allocentric information. However, the approach implies significant computation and it is unlikely it is used as a part of every feature location in visual object recognition. This approach may yield something, but we are not optimistic and do not plan to leverage this current work, at least for now.

So each input layer has both: what feature it is sensing, and the position of the sensory feature is on the object being sensed. The output layer learns complete models of objects as a set of features at relative locations. This notion of a parallel "location" path is creative and looks to support the graph structures we are studying.

There are other cortical models, for example, the work being done in the DiCarlo lab at MIT [8]. DiCarlo, incidentally, is a Co-PI on the Purdue's C-BRIC JUMP center, which I am also a member of. But the Numenta model seems to be the closest to what we want.

We are studying Numenta's models closely and are thinking about building a similar cortical model (though simpler, by not initially having temporal pooling which adds quite a bit of complexity). Should we use Numenta's simulator, possibly augmenting it to fit our goals (https://github.com/numenta). OR, should we develop our own simulator (a complex job)? We are studying the issues before we make a decision.

HTM algorithms have significant complexity. This complexity mostly allows them to make extensive us of temporal information. Although Jeff feels that it is fundamental to everything Cortex does, we've been trying to avoid utilizing temporal information due to the extra complexity it adds.

HTM is intended to be a multi-level, hierarchical algorithm. However, as far as we know, the simulators that Numenta has developed are just for single levels – our interest is in building up a hierarchy of multiple (possibly simpler) levels. And as mentioned they do not, as yet, have an implementation of how allocentric location information is derived and encoded. And finally, they do not appear to have a "parallelized" GPU accelerated implementation of their models in general.

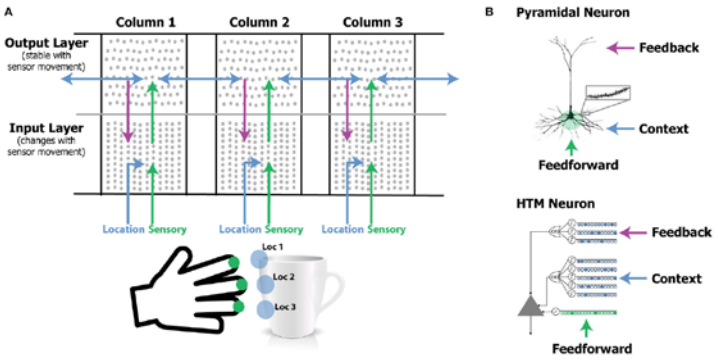

Figure 4 Numenta Cortical Model HTM/CLA

3.2 Results and Discussion

As part of our effort to determine what programming environment we should use, if any, Dan Petre did experiments of traditional deep learning algorithms on cortical like grids using Tensorflow[1]. Tensorflow allows us to take advantage of high performance, inexpensive GPU based cloud computing systems (For example, Amazon's EC2).

Sophie ran a number of experiments using Kohonen's SOM [9]. SOM implements a 2D array of nodes and is unsupervised. Typically the largest and 2nd largest variance is represented in the two dimensions of the array. It seemed promising at first, but eventually we realized it was not close enough to cortical like functionality to be useful to us. These data are not included here.

As discussed above, Sophie also implemented Hinton's Capsule networks using Tensorflow. The models were trained on the MINST character set with excellent results. She also tried CIFAR-10, but it was too compute intensive and the results were not good, so we did not pursue it further.

Here are the simulation results, Xifeng Guo's Keras implementation on Github:

[1] TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

- Batch size: 100, after 18 epochs:
- Validation accuracy – 99.58%
- Validation loss – 0.0115803
- It took 2 days to run 18 epochs.

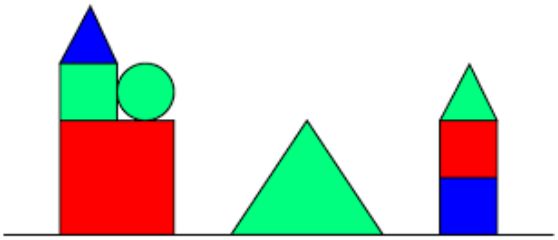

*Figure 5 SEO Figure * ARABIC*

Mark developed a "graph" training set tool development. It allows one to define a simple graph (or let it create one randomly for you), our plan is to use Block's world 2D (no depth) relationships, but initially develop the graphs ourselves. The tool creates training and test sets based on sparse distributed representations. The structure of those sets is selectable. For example, how many graph nodes are in each individual vector. It then generates test and training sets that consist of random subsets of connected nodes, where each node in the subset is connected to at least one other node in the subset. This moderately distributed representation which allows some parallel feature representation. As mentioned we realized that the first iteration did not meet all out needs and we are writing a second version.

Once the revised tool is operational, the next step is to apply these training and test vectors to the Capsule and grid networks. As discussed above, the tool ended up not being exactly what we need, so we are re-building it, based on what we learned the first time.

Aakanksha developed a technique for creating SDRs (Sparse Distributed Representations) from blocks world images [3]. In particular that they incorporate the composition of new objects based on the components.

We may use brute-force techniques for generating the graphs and not worry initially about automatic graph capture algorithms. The goal is to study representation issues first, then look at how to capture such representations. We still ended up struggling with how to deal with the computational complexity, creating a graph from data is NP-Hard, and it is also NP-Hard to do inference over a resulting graph – that is, finding best matching subgraphs. One insight then is that cortical algorithms do not need to build a complete graph, nor perform inference over such a graph. Instead they can operate on graphs "fragments", incomplete approximation to graph information.

It is much more likely that cortical structures build fragments of graphs and these fragments are intersections of features, where the edges in the graph are represented using coincidence detectors in dendritic trees. Graph like relations result when several elements are active at once and there is sufficient overlap. Simultaneity or co-activation, can, in essence, approximate conditional probabilities.

3.3 Conclusions

We made progress during the first year, but it was less than expected, however, given the difficulty of the problem we have focused on, this result is not surprising. The effort has resulted in a more realistic focus for possible additional work that we will propose.

4 RECOMMENDATIONS

We have concluded that for the follow on research we will need to pursue more cortical like models and look at how such models can find structure (still represented by graphical relationship). Such a direction will constitute the primary objectives for the second year of this effort. We also need to develop a simulator to experiment with these more complex models.

5 PERSONNEL

For this grant, Sophie Choe was supported. Sophie is a PhD student in the Electrical and Computer Engineering Department. As of July 1, Sophie moved to the JUMP / C-BRIC grant. Aakanksha Mathuria, will be put on the AFRL grant if it is renewed for a second year.

There are two part time PhD students, who are self-funded and who are a fundamental part of this effort: Ajit Lamaye (Wells Fargo), and Dan Petre (Intel). Likewise, Mark Musil is a senior in ECE and was selected to be part of the PSU Undergraduate Research and Mentoring Program (URMP) which provides some funding and goes for the entire academic year.

6 SYNERGY

FUNDED: JUMP Proposal: C-BRIC - Center For Brain-inspired Computing enabling autonomous intelligence. Purdue is the Prime, Kaushik Roy is PI. Member universities are: Univ. of Pennsylvania, MIT, Princeton, USC, UC Irvine, Georgia Tech, Univ. of Arizona, and Portland State. This program started last spring. There are many excellent collaboration opportunities. In addition, there is much interest in the participating industrial community, in particular, Intel and IBM.

DARPA has launched a major initiative in AI, https://www.darpa.mil/news-events/2018-07-20a, "DARPA is now interested in researching and developing 'third wave' AI theory and applications that address the limitations of first and second wave technologies by making it possible for machines to contextually adapt to changing situations." This focus is directly aligned with the work being done by our research. One advantage is that it is not likely that many groups are pursuing biologically inspired approaches to solving this problem.

We've had some discussion with Intel the possible use of their Liohi chip, "Loihi: A Neuromorphic Many core Processor with On-Chip Learning", Mike Davies, et al., Intel Labs, Intel Corporation [10]. Since we are not using spiking models initially, we have decided not to pursue this at this time.

7 REFERENCES

[1] P. W. Battalia and et al., "Relational inductive biases, deep learning, and graph networks," Cornell University Library - arXiv.org - https://arxiv.org/abs/1806.01261v2, 2018.

[2] R. Szeliski, Computer vision: algorithms and applications, Springer , 2010.

[3] A. Mathuria, "Application of Neuromorphic Computing in Object Recognition," IIIT - Allahabad, Allahabad, 2017.

[4] S. Ahmad and J. Hawkins, "Properties of Sparse Distributed Representations and their Application to Hierarchical Termporal Memory," Cornell University Library - arXiv.org - https://arxiv.org/abs/1503.07469.

[5] S. Sabour, N. Frosst and G. E. Hinton, "Dynamic Routing Between Capsules," Cornell University Library - arXiv.org - https://arxiv.org/abs/1710.09829, 2017.

[6] J. Hawkins and S. Ahmad, "Why Neurons Have Thousands of Synapses - Theory of Sequence Memory in Neocortex," *Frontiers in Neural Circuits,* vol. 10, no. 23, 2016.

[7] E. I. Moser and M.-B. Moser, "Grid Cells and Neural Coding in High-End Cortices," *Neuron Perspective,* vol. 80, no. 3, pp. 765-774, 2013.

[8] J. J. DiCarlo, D. Zoccolan and N. C. Rust, "How Does the Brain Solve Visual Object Recognition," *Neuron Perspective,* vol. 73, no. February 9, 2012, 2012.

[9] "Self-Organizing Map," [Online]. Available: https://en.wikipedia.org/wiki/Self-organizing_map.

[10] M. Davies and et al., "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro (Vol. 38, Issue. 1),* 16 January 2018.

8 ACRONYMS

HTM	Hierarchical Temporal Memory
DARPA	Defense Advanced Research Projects Agency
IoT	Internet of Things
NP-Hard	Non-deterministic Polynomial-time Hardness
NP-Complete	Belong to NP and NP-hard complexity classes
SDR	Sparse Distributed Representations
CNN	Convolutional Neural Networks
GPU	Graphical Processor Unit
LRF	Local Receptive Fields
CIFAR-10	Dataset consists of 60000 32x32 color images