Covariant Compositional Networks & GraphFlow Deep Learning Framework In C++/CUDA

Truong Son Hy
Advisor: Prof. Risi Kondor

The University of Chicago

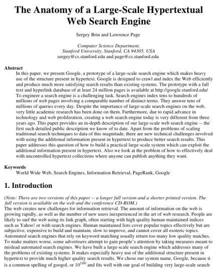
2018

Overview

Introduction

In the field of Machine Learning, standard objects such as vectors, matrices, tensors were carefully studied and successfully applied into various areas including Computer Vision, Natural Language Processing, Speech Recognition, etc. However, none of these standard objects are efficient in capturing the structures of molecules, social networks or the World Wide Web which are not fixed in size. This arises the need of graph representation and extensions of Support Vector Machine and Convolution Neural Network to graphs.

Message Passing - The Motivation



The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/
To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

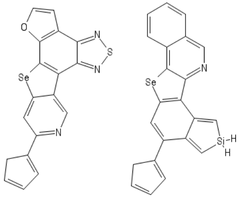
Keywords

World Wide Web, Search Engines, Information Retrieval, PageRank, Google

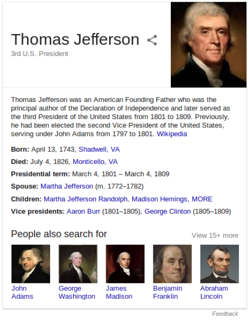
1. Introduction

(Note: There are two versions of this paper -- a longer full version and a shorter printed version. The full version is available on the web and the conference CD-ROM.)
The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as Yahoo! or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10^{100} and fits well with our goal of building very large-scale search

(a) Page Rank



(b) Physics/Chemistry



Thomas Jefferson

3rd U.S. President

Thomas Jefferson was an American Founding Father who was the principal author of the Declaration of Independence and later served as the third President of the United States from 1801 to 1809. Previously, he had been elected the second Vice President of the United States, serving under John Adams from 1797 to 1801. Wikipedia

Born: April 13, 1743, Shadwell, VA

Died: July 4, 1826, Monticello, VA

Presidential term: March 4, 1801 – March 4, 1809

Spouse: Martha Jefferson (m. 1772–1782)

Children: Martha Jefferson Randolph, Madison Hemings, MORE

Vice presidents: Aaron Burr (1801–1805), George Clinton (1805–1809)

People also search for — View 15+ more

John Adams · George Washington · James Madison · Benjamin Franklin · Abraham Lincoln

Feedback

(c) Knowledge graph

Message Passing - Label Propagation Algorithm

Given an input graph / network $G = (V, E)$:

1. Initially, each vertex v of the graph is associated with a feature representation l_v (label) or f_v^0. This feature representation can also be called as a *message*.
2. Iteratively, at iteration ℓ, each vertex collects / aggregates all messages of the previous iteration $\{f_{v_1}^{\ell-1}, .., f_{v_k}^{\ell-1}\}$ from other vertices in its neighborhood $\mathcal{N}(v) = \{v_1, .., v_k\}$, and then produces a new message f_v^ℓ via some *hashing function* $\Phi(.)$.
3. The graph representation $\phi(G)$ is obtained by aggregating all messages in the last iteration of every vertex. $\phi(G)$ is then used for downstream application.

Message Passing - Label Propagation Algorithm

```
 1: for $v \in V$ do
 2:     $f_v^0 \leftarrow l_v$
 3: end for
 4: for $\ell = 1 \rightarrow L$ do
 5:     for $v \in V$ do
 6:         $f_v^\ell \leftarrow \Phi\left(f_{v_1}^{\ell-1}, .., f_{v_k}^{\ell-1}\right)$ where $\mathcal{N}(v) = \{v_1, .., v_k\}$
 7:     end for
 8: end for
 9: $\phi(G) \leftarrow \Phi\left(f_1^L, .., f_{|V|}^L\right)$
10: Use $\phi(G)$ for downstream regression / classification tasks.
```

Message Passing - Graph Neural Networks

```
1:  for $v \in V$ do
2:      $f_v^0 \leftarrow l_v$
3:  end for
4:  for $\ell = 1 \rightarrow L$ do
5:      for $v \in V$ do
6:          $f_v^\ell \leftarrow \Phi\left(f_{v_1}^{\ell-1}, .., f_{v_k}^{\ell-1}; \{W_1^\ell, .., W_{n_\ell}^\ell\}\right)$ where $\mathcal{N}(v) = \{v_1, .., v_k\}$
7:      end for
8:  end for
9:  $\phi(G) \leftarrow \Phi\left(f_1^L, .., f_{|V|}^L; \{W_1, .., W_n\}\right)$
10: Use $\phi(G)$ for downstream regression / classification tasks.
```

The gradient with respect to W can be computed via Back-propagation.

Message Passing - What others do?

1. Weisfeiler-Lehman graph kernel (Shervashidze et al., 2011):
   - Extension of Weisfeiler-Lehman graph isomorphism test
   - Applicable with Support Vector Machine and kernel methods
2. Neural Graph Fingerprint (Duvenaud et al., 2015):
   - Vector vertex representation
   - Φ as summation
3. Learning Convolutional Neural Networks (Niepert et al., 2016):
   - Flatten the graph into a long vector of vertices
   - Apply traditional convolution on top
4. Gated Graph Neural Networks (Li et al., 2017):
   - Similar to Neural Graph Fingerprint
   - Embeding LSTM/GRU
5. Message Passing Neural Networks (Gilmer et al., 2017):
   - Summary of the field

THE UNIVERSITY OF CHICAGO

Message Passing - What others miss?

We will describe how our compositional architecture is a generalization of previous works with an extension to higher-order representations, which can retain this structural information.

Recent works on graph neural networks can all be seen as instances of *zeroth order message passing* where each vertex representation is a vector (1st order tensor) of c channels in which each channel is represented by a scalar (zeroth order P-tensor). This results in the loss of certain structural information during the message aggregation, and the network loses the ability to learn topological information of the graph's multiscale structure.

Covariant Compositional Networks - Scheme

Definition

Let $\mathcal{G}$ be an object with n elementary parts (atoms) $\mathcal{E} = \{e_1, .., e_n\}$. A **compositional scheme** for $\mathcal{G}$ is a directed acyclic graph (DAG) $\mathcal{M}$ in which each node v is associated with some subset $\mathcal{P}_v$ of $\mathcal{E}$ (these subsets are called **parts** of $\mathcal{G}$) in such a way that:

1. In the bottom level, there are exactly n leaf nodes in which each leaf node v is associated with an elementary atom e. Then $\mathcal{P}_v$ contains a single atom e.
2. $\mathcal{M}$ has a unique root node v_r that corresponds to the entire set $\{e_1, .., e_n\}$.
3. For any two nodes v and v', if v is a descendant of v', then $\mathcal{P}_v$ is a subset of $\mathcal{P}_{v'}$.

Covariant Compositional Networks - Definition

The compositional network $\mathcal{N}$ is constructed as follows:

1. In layer $\ell = 0$, each leaf node v_i^0 represents the single vertex $\mathcal{P}_i^0 = \{i\}$ for $i \in V$. The corresponding feature tensor f_i^0 is initialized by the vertex label l_i.
2. In layers $\ell = 1, 2, .., L$, node v_i^ℓ is connected to all nodes from the previous level that are neighbors of i in G. The children of v_i^ℓ are $\{v_j^{\ell-1} | j : (i,j) \in E\}$. Thus, $\mathcal{P}_i^\ell = \bigcup_{j:(i,j)\in E} \mathcal{P}_j^{\ell-1}$. The feature tensor f_i^ℓ is computed as an aggregation of feature tensors in the previous layer:
$$f_i^\ell = \Phi(\{f_j^{\ell-1} | j \in \mathcal{P}_i^\ell\})$$
where Φ is some aggregation function.
3. In layer $\ell = L+1$, we have a single node v_r that represents the entire graph and collects information from all nodes at level $\ell = L$:
$$\mathcal{P}_r \equiv V$$
$$f_r = \Phi(\{f_i^L | i \in \mathcal{P}_r\})$$

Covariant Compositional Networks - Covariance

Definition

For a graph G with the comp-net $\mathcal{N}$, and an isomorphic graph G' with comp-net $\mathcal{N}'$, let v be any neuron of $\mathcal{N}$ and v' be the corresponding neuron of $\mathcal{N}'$. Assume that $\mathscr{P}_v = (e_{p_1}, .., e_{p_m})$ while $\mathscr{P}_{v'} = (e_{q_1}, .., e_{q_m})$, and let $\pi \in \mathbb{S}_m$ be the permutation that aligns the orderings of the two receptive fields, i.e., for which $e_{q_{\pi(a)}} = e_{p_a}$. We say that $\mathcal{N}$ is **covariant to permutations** if for any π, there is a corresponding function R_π such that $f_{v'} = R_\pi(f_v)$.

Covariant Compositional Networks - First order

We propose **first order message passing** by representing each vertex v by a matrix: $f_v^\ell \in \mathbb{R}^{|\mathscr{P}_v^\ell| \times c}$, each row of this feature matrix corresponds to a vertex in the neighborhood of v.

Definition

We say that v is a **first order covariant node** in a comp-net if under the permutation of its receptive field $\mathscr{P}_v$ by any $\pi \in \mathbb{S}_{|\mathscr{P}_v|}$, its activation transforms as $f_v \mapsto P_\pi f_v$, where P_π is the permutation matrix:

$$[P_\pi]_{i,j} \triangleq \begin{cases} 1, & \pi(j) = i \\ 0, & otherwise \end{cases} \tag{1}$$

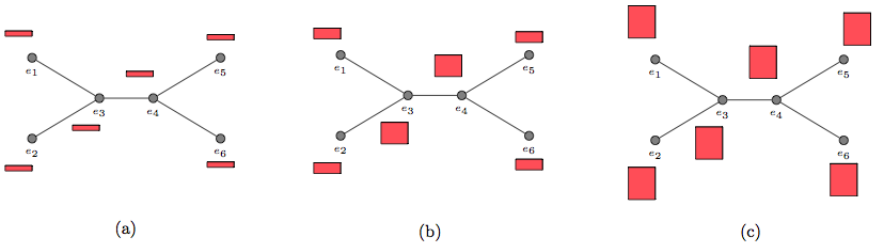
The transformed activation $f_{v'}$ will be:

$$[f_{v'}]_{a,s} = [f_v]_{\pi^{-1}(a),s}$$

where s is the channel index.

Covariant Compositional Networks - First order

Figure: CCN 1D on C_2H_4 molecular graph



Covariant Compositional Networks - Second order

Instead of representing a vertex with a feature matrix as done in first order message passing, we can represent it by a 3rd order tensor $f_v^\ell \in \mathbb{R}^{|\mathcal{P}_v^\ell| \times |\mathcal{P}_v^\ell| \times c}$ and require these feature tensors to transform covariantly in a similar manner:

Definition

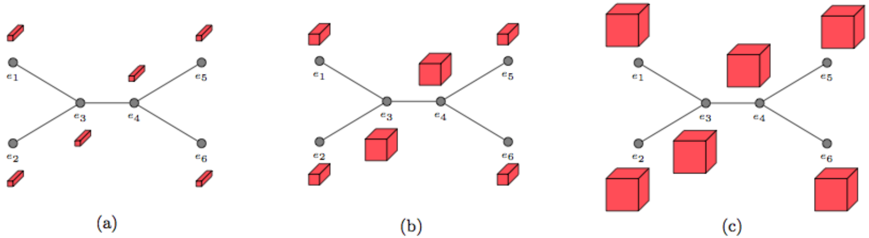
We say that v is a **second order covariant node** in a comp-net if under the permutation of its receptive field $\mathcal{P}_v$ by an $\pi \in \mathbb{S}_{|\mathcal{P}_v|}$, its activation transforms as $f_v \mapsto P_\pi f_v P_\pi^T$. The transformed activation $f_{v'}$ will be:

$$[f_{v'}]_{a,b,s} = [f_v]_{\pi^{-1}(a),\pi^{-1}(b),s}$$

where s is the channel index.

Covariant Compositional Networks - Second order

Figure: CCN 2D on C_2H_4 molecular graph



Covariant Compositional Networks - Second order

$$\mathcal{T}_{i_1,i_2,i_3,i_4,i_5,i_6} = (\mathcal{F}_{i_1})_{i_2,i_3,i_6} \cdot \mathcal{A}_{i_4,i_5} \tag{2}$$

1. The **1+1+1** case contracts $\mathcal{T}$ in the form $\mathcal{T}_{i_1,i_2,i_3,i_4,i_5}\delta^{i_{a_1}}\delta^{i_{a_2}}\delta^{i_{a_3}}$, i.e., it projects $\mathcal{T}$ down along 3 of its 5 dimensions. This can be done in $\binom{5}{3} = 10$ ways.
2. The **1+2** case contracts $\mathcal{T}$ in the form $\mathcal{T}_{i_1,i_2,i_3,i_4,i_5}\delta^{i_{a_1}}\delta^{i_{a_2},i_{a_3}}$, i.e., it projects $\mathcal{T}$ along one dimension, and contracts it along two others. This can be done in $3\binom{5}{3} = 30$ ways.
3. The **3** case is a single 3-fold contraction $\mathcal{T}_{i_1,i_2,i_3,i_4,i_5}\delta^{i_{a_1},i_{a_2},i_{a_3}}$. This can be done in $\binom{5}{3} = 10$ ways.

Totally, we have 50 different contractions that result in 50 times more channels. In practice, we only implement 18 contractions for efficiency.

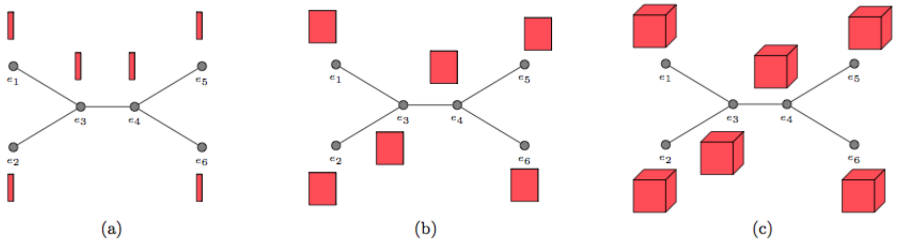
THE UNIVERSITY OF CHICAGO

Covariant Compositional Networks - Second order

```
1: Input: $G$, $l_v$, $L$.
2: Parameters: Matrices $W_0 \in \mathbb{R}^{c \times c}$, $W_1, .., W_L \in \mathbb{R}^{(18c) \times c}$ and biases $b_0, .., b_L$.
3: $F_v^0 \leftarrow \Upsilon(W_0 l_v + b_0 \mathbb{1})$ $(\forall v \in V)$
4: Reshape $F_v^0$ to $1 \times 1 \times c$ $(\forall v \in V)$
5: for $\ell = 1, .., L$ do
6:     for $v \in V$ do
7:         $F_{w \to v}^\ell \leftarrow \chi \times F_w^{\ell-1} \times \chi^T$ where $\chi = \chi_{w \to v}^\ell$ $(\forall w \in \mathscr{P}_v^\ell)$
8:         Apply tensor contractions with inputs $\{F_{w \to v}^\ell | w \in \mathscr{P}_v^\ell\}$ and the re-
           stricted adjacency matrix $A\downarrow_{\mathscr{P}_v^\ell}$ to compute $\overline{F}_v^\ell \in \mathbb{R}^{|\mathscr{P}_v^\ell| \times |\mathscr{P}_v^\ell| \times (18c)}$.
9:         $F_v^\ell \leftarrow \Upsilon(\overline{F}_v^\ell \times W_\ell + b_\ell \mathbb{1})$
10:    end for
11: end for
12: $F^\ell \leftarrow \sum_{v \in V} \Theta(F_v^\ell)$ $(\forall \ell)$
13: Graph feature $F \leftarrow \bigoplus_{\ell=0}^{L} F^\ell \in \mathbb{R}^{(L+1)c}$
14: Use $F$ for downstream tasks.
```

Covariant Compositional Networks - 0th, 1st, 2nd orders

Figure: Zeroth, first and second order message passing



GraphFlow Deep Learning Framework - TensorFlow

1. Static computation graph
2. Able to have object abtraction, but unable to differentiate in back-propagation
3. How Google Brain trains graph neural networks (Gilmer et al., 2017):
   - Concatenate all graphs (each graph as an example) into a huge graph in which each connected component is an individual graph
   - Write 0th Message Passing in matrix form, and apply to the huge graph

GraphFlow Deep Learning Framework - PyTorch

1. Dynamic computation graph
2. Unable to have object abtraction
   - Proof: `Variable` **inherits** `at::Tensor`.
3. Only efficient with graph neural networks that can be expressed in matrix forms

GraphFlow Deep Learning Framework - Philosophy

1. C++
2. Flexibility
3. Little engine in your machine to build your own experiments
4. Assumption that users build **non-traditional** networks

GraphFlow Deep Learning Framework - Overview

GraphFlow is designed with the philosophy of Object Oriented Programming (OOP). There are several classes divided into the following groups:

1. **Data structures**: `Entity`, `Vector`, `Matrix`, `Tensor`, etc. Each of these components contain two arrays of floating-point numbers: `value` for storing the actual values, `gradient` for storing the gradients for the purpose of automatic differentiation. Also, in each class, there are two functions: `forward()` and `backward()` in which `foward()` to evaluate the network values and `backward()` to compute the gradients.

GraphFlow Deep Learning Framework - Overview

Power of polymorphism

Based on the OOP philosophy, `Vector` inherits from `Entity`, and both `Matrix` and `Tensor` inherit from `Vector`, etc. It is essentially important because polymorphism allows us to construct the computation graph of the neural network as a **Directed Acyclic Graph** (DAG) of `Entity` such that `forward()` and `backward()` functions of different classes can be called with object casting.

GraphFlow Deep Learning Framework - Overview

2 **Operators**: Matrix Multiplication, Tensor Contraction, Convolution, etc.

For example, the matrix multiplication class `MatMul` inherits from `Matrix` class, and has 2 constructor parameters in `Matrix` type. Suppose that we have an object `A` of type `MatMul` that has 2 `Matrix` inputs `B` and `C`. In the `forward()` pass, `A` computes its value as `A = B * C` and stores it into `value` array. In the `backward()` pass, `A` got the gradients into `gradient` (as flowing from the loss function) and increase the gradients of `B` and `C`.

GraphFlow Deep Learning Framework - Topological order

Topological order

It is important to note that our computation graph is DAG and we find the topological order to evaluate `value` and `gradient` in the correct order. That means `A -> forward()` is called after both `B -> forward()` and `C -> forward()`, and `A -> backward()` is called before both `B -> backward()` and `C -> backward()`.

GraphFlow Deep Learning Framework - Topological order

Topological order

It is important to note that our computation graph is DAG and we find the topological order to evaluate `value` and `gradient` in the correct order. That means `A -> forward()` is called after both `B -> forward()` and `C -> forward()`, and `A -> backward()` is called before both `B -> backward()` and `C -> backward()`.

GraphFlow vs TensorFlow & PyTorch

GraphFlow allows programmers to define their own topological order (depending on each individual example). In contrast, TensorFlow & PyTorch are **not** able to.

GraphFlow Deep Learning Framework - Overview

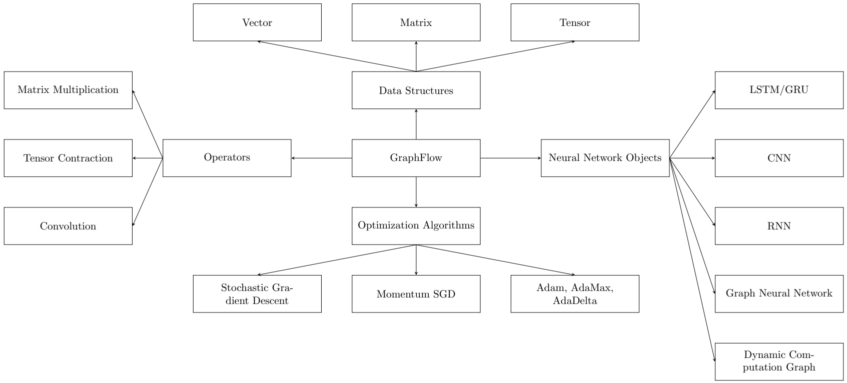
3 **Optimization algorithms**: Stochastic Gradient Descent (SGD), SGD with Momentum, Adam, AdaGrad, AdaMax, AdaDelta, etc. These algorithms are implemented into separate drivers: these drivers get the values and gradients of learnable parameters computed by the computation graph and then optimize the values of learnable parameters algorithmically.

GraphFlow Deep Learning Framework - Overview

4 **Neural Networks objects**: These are classes of neural network architectures implemented based on the core of GraphFlow including graph neural networks (for example, CCN, NGF and LCNN), convolutional neural networks, recurrent neural networks (for example, GRU and LSTM), multi-layer perceptron, etc. Each class has multiple supporting functions: load the trained learnable parameters from files, save them into files, learning with mini-batch or without mini-batch, using multi-threading or not, etc.

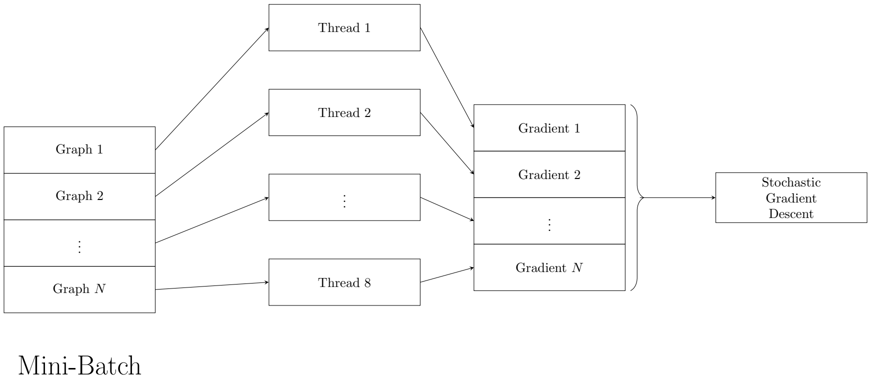
GraphFlow Deep Learning Framework - Overview

Figure: GraphFlow overview



GraphFlow Deep Learning Framework - Multi-threading

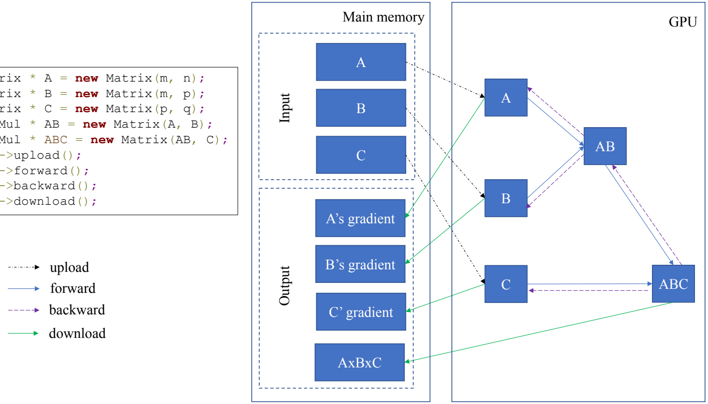
Figure: CPU multi-threading for gradient computation



GraphFlow Deep Learning Framework - GPU

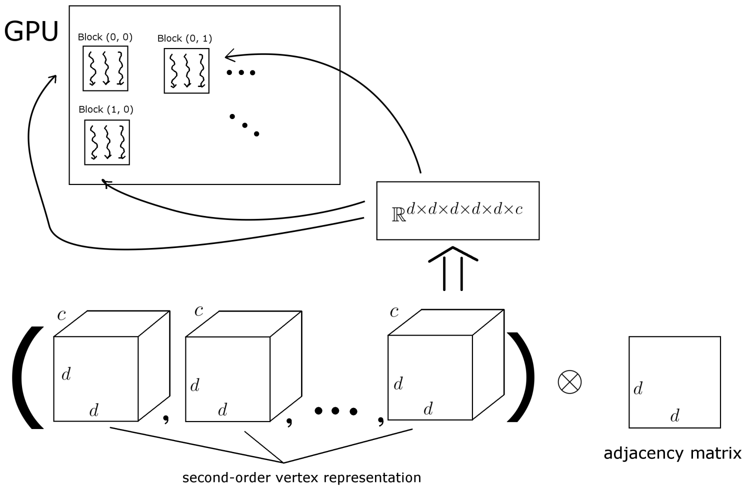
Figure: Example of data flow between GPU and main memory

```
Matrix * A = new Matrix(m, n);
Matrix * B = new Matrix(m, p);
Matrix * C = new Matrix(p, q);
MatMul * AB = new Matrix(A, B);
MalMul * ABC = new Matrix(AB, C);
ABC->upload();
ABC->forward();
ABC->backward();
ABC->download();
```



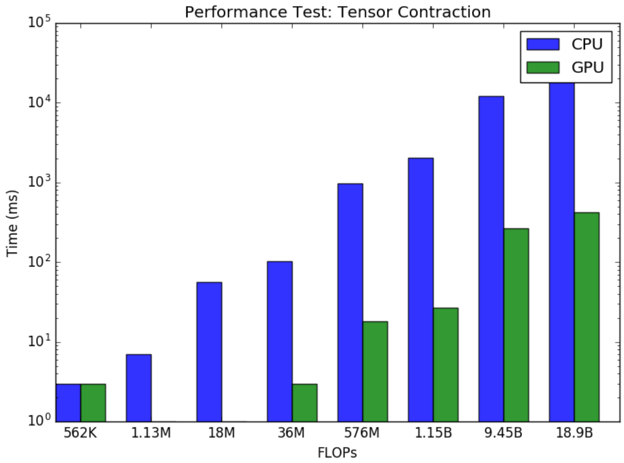
GraphFlow Deep Learning Framework - GPU

Figure: Tensor contraction with GPU


GPU
Block (0, 0)
Block (0, 1)
Block (1, 0)
$\mathbb{R}^{d\times d\times d\times d\times c}$
c
d
d
$\otimes$
adjacency matrix
second-order vertex representation

GraphFlow Deep Learning Framework - GPU

Figure: Tensor contraction with CPU vs GPU



GraphFlow Deep Learning Framework - Open Source

The source code of GraphFlow Deep Learning framework can be found at:

`https://github.com/HyTruongSon/GraphFlow`

Experiments - Objective

Objective

Density Functional Theory (DFT) is the workhorse of the molecular chemistry community, given its favorable tradeoff between accuracy and computational power. Still, it is too costly for tasks such as drug discovery or materials engineering, which may require searching through millions of candidate molecules. An accurate prediction of molecular properties would significantly aid in such tasks.

Experiments - Datasets

We compare our CCN framework to several standard graph learning algorithms on two datasets that contain the result of a large number of Density Functional Theory (DFT) calculations:

1. **The Harvard Clean Energy Project (HCEP)**, consisting of 2.3 million organic compounds that are candidates for use in solar cells.
2. **QM9**, a dataset of ~134k organic molecules with up to nine heavy atoms (C, O, N and F) out of the GDB-17 universe of molecules. Each molecule contains data including 13 target chemical properties, along with the spatial position of every constituent atom.

Experiments - Learned molecular representation

Figure: 2D PCA projections of Weisfeiler-Lehman features in HCEP

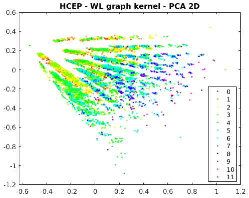
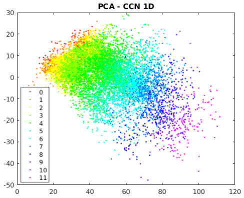
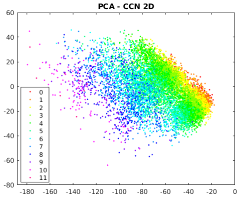


Figure: 2D PCA projections of CCNs graph representations in HCEP





Experiments - Learned molecular representation

Figure: 2D t-SNE projections of Weisfeiler-Lehman features in HCEP

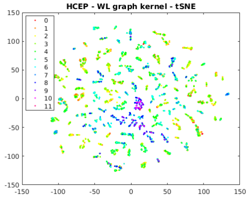
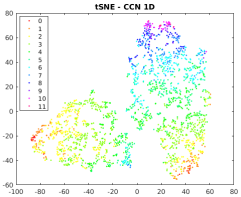
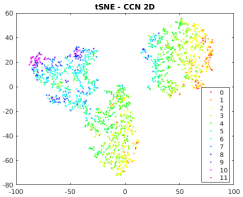


Figure: 2D t-SNE projections of CCNs graph representations in HCEP





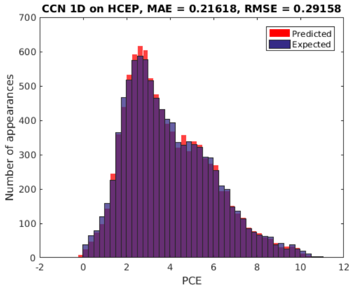
Experiments - HCEP

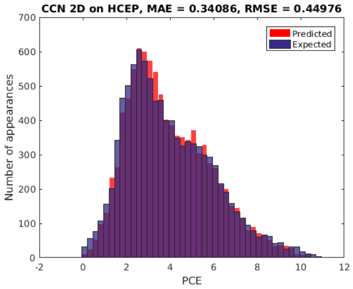
Table: HCEP regression results

	Test MAE	Test RMSE
Lasso	0.867	1.437
Ridge regression	0.854	1.376
Random forest	1.004	1.799
Gradient boosted trees	0.704	1.005
WL graph kernel	0.805	1.096
Neural graph fingerprints	0.851	1.177
PSCN	0.718	0.973
CCN 1D	**0.216**	**0.291**
CCN 2D	**0.340**	**0.449**

Experiments - HCEP

Figure: Distributions of ground-truth and prediction of CCN 1D & 2D in HCEP





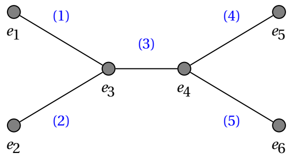
Experiments - QM9 (graph features)

Target	WLGK	NGF	PSCN	CCN 2D
alpha	0.46	0.43	0.20	**0.16**
Cv	0.59	0.47	0.27	**0.23**
G	0.51	0.46	0.33	**0.29**
gap	0.72	0.67	0.60	**0.54**
H	0.52	0.47	0.34	**0.30**
HOMO	0.64	0.58	0.51	**0.39**
LUMO	0.70	0.65	0.59	**0.53**
mu	0.69	0.63	0.54	**0.48**
omega1	0.72	0.63	0.57	**0.45**
R2	0.55	0.49	0.22	**0.19**
U	0.52	0.47	0.34	**0.29**
U0	0.52	0.47	0.34	**0.29**
ZPVE	0.57	0.51	0.43	**0.39**

THE UNIVERSITY OF CHICAGO

Experiments - QM9 (physical features)

To include the edge features into our model along with the vertex features, we used the concept of a **line graph** from graph theory. We constructed the line graph for each molecular graph in such a way that: an edge of the molecular graph corresponds to a vertex in its line graph, and if two edges in the molecular graph share a common vertex then there is an edge between the two corresponding vertices in the line graph.



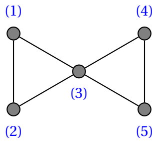


Figure: Molecular graph of C_2H_4 (left) and its corresponding line graph (right).

Experiments - QM9 (physical features)

Target	CCNs	DFT error	Physical unit
alpha	**0.19**	0.4	Bohr^3
Cv	**0.06**	0.34	cal/mol/K
G	**0.05**	0.1	eV
gap	**0.11**	1.2	eV
H	**0.05**	0.1	eV
HOMO	**0.08**	2.0	eV
LUMO	**0.07**	2.6	eV
mu	0.43	**0.1**	Debye
omega1	**2.54**	28	cm^{-1}
R2	5.03	-	Bohr^2
U	**0.06**	0.1	eV
U0	**0.05**	0.1	eV
ZPVE	**0.0043**	0.0097	eV

Experiments - Discussion

On the subsampled HCEP (50K examples) dataset, CCN outperforms all other methods by a very large margin. In the QM9 (graph features) experiment, CCN obtains better results than three other graph learning algorithms for all 13 learning targets. In the QM9 (physical features) experiment, our method gets smaller errors comparing to the DFT calculation in 11 out of 12 learning targets (we do not have the DFT error for R2).

Conclusion

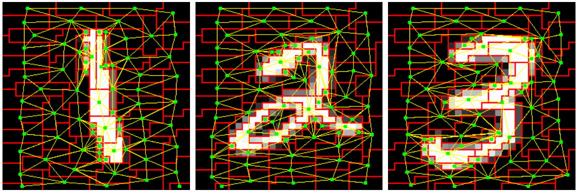
We extended Message Passing Neural Networks and generalized convolution operation for Covariant Compositional Networks by higher-order representations in order to approximate Density Functional Theory. We obtained very promising results and outperformed other state-of-ther-art graph neural networks on Harvard Clean Energy Project and QM9 datasets.

We are developing our custom Deep Learning framework named **GraphFlow** which supports automatic and symbolic differentitation, dynamic computation graph as well as complex tensor/matrix operations with GPU computation acceleration.

Future Work - Computer Vision

The next step would be to find applications of CCNs in different areas of computer science.

Example 1: Understanding visual scenerio – segment the image (e.g., by Minimum Spanning Tree) to construct the representation graph in which each vertex corresponds to a connected region of the image, the vertex input features can be extracted by SIFT or HoG. The model is **rotational invariant**.

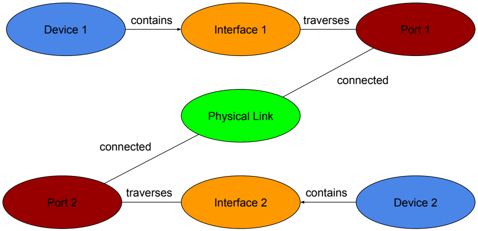


(a) One (b) Two (c) Three

Future Work - Networking

Example 2 (from my Google internship): Applying graph neural networks on large-scale data center's network topology and monitoring timeseries data to detect and find the root causes of network failures.

Figure: Unified Network Model (2-device model)



Acknowledgment

I would like to express my deepest gratitude to my advisor, **Professor Risi Kondor** – who has supported me, helped me, guided me, and conveyed continually and convincingly to all of us, his students, a spirit of discovery in regard to research and scholarship. I would like to thank my committee, **Professor Sanjay Krishnan** and **Professor Michael Maire**, for their valuable feedback. Finally, a very special thanks to the entire **UChicago Machine Learning group** and **Department of Computer Science**.