Deep Learning for Network Analysis: Problems, Approaches and Challenges

Siddharth Pal*, Yuxiao Dong†, Bishal Thapa*, Nitesh V. Chawla†, Ananthram Swami‡, Ram Ramanathan*
*Raytheon BBN Technologies, †University of Notre Dame, ‡Army Research Lab

***Abstract*—The analysis of social, communication and information networks for identifying patterns, evolutionary characteristics and anomalies is a key problem for the military, for instance in the Intelligence community. Current techniques do not have the ability to discern unusual features or patterns that are not a priori known.**

We investigate the use of deep learning for network analysis. Over the last few years, deep learning has had unprecedented success in areas such as image classification, speech recognition, etc. However, research on the use of deep learning to network or graph analysis is limited. We present three preliminary techniques that we have developed as part of the ARL Network Science CTA program: (a) unsupervised classification using a very highly trained image recognizer, namely Caffe; (b) supervised classification using a variant of convolutional neural networks on node features such as degree and assortativity; and (c) a framework called *node2vec* for learning representations of nodes in a network using a mapping to natural language processing.

I. Introduction

Massive amounts of data are being collected for military intelligence purposes, much of it on social and information networks. Current techniques for mining and analyzing such data range from manual inspection to the use of statistical features or standard machine learning techniques such as Support Vector Machines. These techniques are limited in their ability to identify subtle or unusual features that are beyond the set of features the analyst is explicitly looking for. For example, do terrorist networks such as the ones shown in Figure 1 have common features that reliably distinguish it from non-terrorist ones? More generally, suppose we have a large graph representing an unknown social network, and would like to know if a terrorist subnetwork is hidden within it – in other words, does this graph contain any signatures for adversarial activity? Conventional techniques require the design of a feature extractor with known features, and manual analysis does not scale.

We investigate the application of deep learning (DL) to the broad area of network analysis. Deep learning is a burgeoning field that has shown promise in building detectors for discriminative features that are nearly impossible to build by hand or other learning techniques. It typically consists of several layers of neural networks, each layer taking as input the output from the previous layer. This enables representation of data at multiple levels of abstraction, and enables the recognition of intricate features. Deep learning has recently garnered several successes, chiefly in the areas of image recognition and classification [2], speech processing, natural language understanding [3], etc. However, there exists little research on using DL for network analysis.

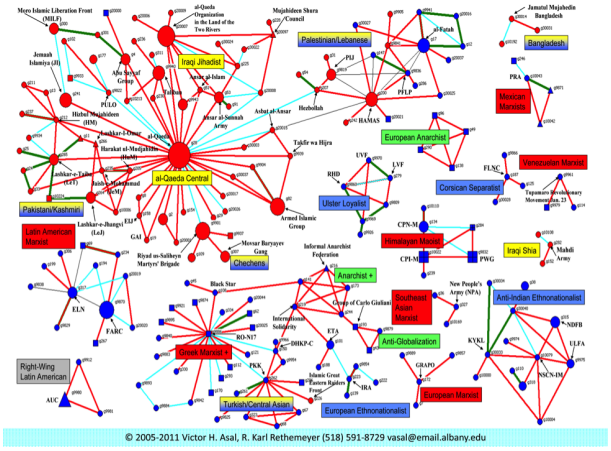

Figure 1: Structure of terrorist networks [1].

A key reason for the recent successes of deep learning in the aforementioned domains is due to the statistical properties of the data in the domains, namely the stationarity and compositionality [4]. In other words, the most successful of the techniques such as deep convolutional neural networks leverage the "grid structured" nature of data [5]. However, when the data in question is graph-structured, such as for network analysis, these techniques do not work well, and we need new techniques.

In this paper, we explore three approaches for analyzing graph-structured data, each targeting a different problem in the area of network analysis. We begin by asking the question: can we map our problem to one already solved? We explore the simplest possible approach, which is to feed the actual *network image* produced by a consistent a priori chosen planar embedding, to a mature image classification engine, namely Caffe [6], and use its output to cluster the input data. Although this has obvious limitations, it is able to classify reasonably

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 (the ARL Network Science CTA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

978-1-5090-3781-0/16/$31.00 ©2016 IEEE

between different random graph models, at least for the same size, and its discriminative power piggybacks on the growing capabilities of Caffe. We describe our approach and results in section IV.

The next approach uses the actual graph data and employs a technique called *search convolution* [7]. We replace the node attributes of the previous work by features of the graph, including degree, local clustering coefficient, and assortativity. Employing search convolution with a 2-layer network, we are able to obtain perfect classification for moderately-sized graphs (200 and 500 nodes). We describe our approach and results in section V.

Our final approach seeks to learn latent representations that capture internal relations in graph-structured data that can be applied to a variety of domains such as node classification, link prediction, community detection, etc. Specifically, we build upon the *word2vec* framework proposed by Mikolov et al[8], [9], to obtain a framework called *node2vec* that is capable of learning desirable node representations in networks. We present a preliminary version of the framework that is able to learn the latent representations of conference venues and authors in a heterogeneous network, and predict the similarities between nodes (authors/venues). The approach and results are described in section VI.

The general problem of network analysis and classification is challenging. Our three approaches represent an initial exploratory foray into this space, intended more as a way of understanding the potential of these approaches in a "breadth first" manner. Our initial results are encouraging, but also point to the need for fundamental new innovations that can pave the way for applying deep neural networks to graph-structured data.

II. Related Work

The advantages of deep learning over conventional techniques, e.g., support vector machines, have been recognized in several domains of classification and learning [10]. Prior work on graph classification has largely employed a priori designated statistical features [11], or kernel-based classification methods which adopt predefined similarity measures between graphs [12]. These methods by definition can only detect high-level features that are combinations of the a priori defined low-level ones, and are therefore limited in their ability to detect discriminative or unusual spatial patterns and temporal interactions within adversarial network graphs. Recently, deep learning techniques have been proposed for dealing with graph-structured data [4], [7]. While Henaff et al. [4] defines a graph convolution operation for applying deep learning on classification tasks, Atwood and Towsley [7] also extends the notion of convolution, albeit in a different manner using graph search, and employ deep learning to solve node classification problems with pointers for handling graph classification tasks.

In natural language processing, deep representation learning aims to learn the embeddings of words that can be used to infer their context—nearby words in a sentence or document [13], [9]. In particular, Mikolov et al. first incorporated the Skip-gram architecture into neural networks for word representation learning [9]. Recently developed network representation learning methods have been largely inspired by this framework [14], [15]. In this work, we propose to enhance the Skip-gram based neural network model so as to handle heterogeneous networks, wherein unique challenges arise.

III. Background

Deep learning is a branch of machine learning that attempts to model high-level abstractions in data by using multiple processing layers, composed of non-linear transformations [3]. It belongs in a broader family of machine learning methods based on learning representations of data, allowing us to replace handcrafted features with efficient algorithms for extracting features in a hierarchical fashion. In other words, deep learning builds complex concepts out of simpler concepts or representations.

The representative example of a deep learning model is the feedforward neural network or the multilayer perceptron. A multilayer perceptron is a mathematical model consisting of building blocks called neurons which simply map an input vector to an output value by weighting the inputs linearly, and applying a nonlinearity (activation function) on the weighted sum. In particular, a multilayer perceptron consists of a visible layer where the input is presented, one or more hidden layers which extract increasingly abstract features from the data, and an output layer which gives the classification result.

A particular kind of feedforward neural network is the convolutional network which is used primarily for processing data that has a known grid-like topology. Such networks consists of multiple convolutional layers, where each neuron performs a convolution operation on small regions of the input. This drastically reduces the number of free parameters and improves performance. Convolutional neural networks have become popular in the past several decades due to success in handwriting recognition, image recognition and natural language processing [2], [16].

We use random graph models to evaluate our approaches, namely Erdös-Rényi (ER) and Barábasi-Albert (BA). An ER graph $\mathbb{G}(n;p)$ is a graph on n nodes, where connections between nodes occur in an i.i.d. fashion, with probability p. On the other hand, the BA model in its simplest form is a growth model that starts with an initial graph G_0, and progressively grows by adding a new node and m edges between the new node and the existing nodes using the preferential attachment mechanism, i.e., the new node connects with an existing node with probability proportional to its degree.

Popular DL implementation frameworks are Caffe [6], Theano [17], Torch [18], Tensor Flow [19], etc. Caffe is a popular library for convolutional neural networks primarily geared for image classification tasks. Theano is used by researchers for handling a more diverse set of problems. A comparison of various implementation frameworks, with a focus on run time is given in [20].

IV. Mapping to Image Classification

A. Problem

We are given graphs drawn from various random or regular models (e.g. Barabási-Albert, Erdös-Rényi, Line graph etc.). We need to group them into various classes using some application of Deep Learning such that graphs drawn from the same model are in the same class.

B. Approach

As mentioned earlier, the approach that we use in this section is simple – we feed an image of the graph to a well-trained image recognizing DL engine. Under this effort, we use an unsupervised DL framework called Caffe as our deep graph learning tool [6]. It is primarily designed for image classification, but we find it to be a perfect fit for our application due to its modular, expressive, and very capable (in terms of speed and efficiency) implementation of deep learning toolboxes.

Caffe takes images as input and runs it against its pre-trained Convolutional Neural Network (CNN) model. It outputs two types of scores: a maximally specific score and a maximally accurate score. We use the maximally accurate scores to classify the input of network graph images. Table I shows a sample classification for different families of network graphs. We use NetworkX to draw input graph images and feed into the Caffe [21]. Each row in the output represents the similarity metric between the input image and a labelled image from the training dataset in the order of their similarity rank. Note that Caffe's pre-trained model (and corresponding training images) do not contain any network graphs. Therefore, the absolute classification is not accurate, however, we will see in Figure 2 that the relative classification is surprisingly accurate.

C. Results

We first describe the two metrics we use on top of Caffe's maximally accurate score to provide a tradeoff between granularity in classification vs. accuracy.

Graph	Caffe CNN Classification
ER (40, 0.1, 26) # of nodes: $n = 40$ Edge probability: $p = 0.1$ Seed for random number generator: 26	[('binder', '0.10054'), ('Windsor tie', '0.07407'), ('chain mail', '0.06515'), ('honeycomb', '0.04726'), ('space heater', '0.03602')]
ER (40, 0.2, 55) # of nodes: $n = 40$ Edge probability: $p = 0.2$ Seed for random number generator: 55	[('binder', '0.08910'), ('Windsor tie', '0.07250'), ('envelope', '0.04664'), ('chain mail', '0.03803'), ('web site', '0.03381')]
BA (40, 1, 55) # of nodes: $n = 40$ # of edges from new node: $m = 1$ Seed for random number generator: 55	[('chain mail', '0.10623'), ('Windsor tie', '0.08268'), ('binder', '0.07917'), ('honeycomb', '0.04329'), ('window screen', '0.02943')]

Table I: Caffe classification of various Erdös-Rényi (ER) and Barabási-Albert (BA) Graphs

Hamming distance based Metric (HAM): This is a hamming distance inspired classification metric where we ignore the specific scores per image classification, and rather use the rank of output list of images that are most similar to the input image to differentiate one classification from the other. The difference in the rank at each level (for instance, 1 to 10) is denoted by a 1 and equality by a 0. The HAM metric is then calculated as the decimal value of the binary string generated starting from the first rank to the last. For example, if two input random graphs were classified as follows:

- BA(40,1,26): [harvestman, walking stick, long-horned beetle, bee eater, dragonfly, black and gold garden spider, bald eagle, kite, pole, ant]
- BA(40,1,5): [harvestman, walking stick, bald eagle, bee eater, wolf spider, vulture, kite, umbrella, long-horned beetle, black and gold garden spider]

Then, HAM(BA(40,1,26), BA(40,1,5)) = [0010111111] = 191. If one of the graphs was a clique and the other was a Barabási-Albert graph with no similarity at any of the ranks, then the HAM score would be equal to 1027. A larger HAM score implies that the input image types are farther apart from each other.

Cumulative HAM (CuHAM): CuHAM loosens the granularity of the HAM metric further for more accuracy by looking at the list of output images as a set rather than an ordered list. So, CuHAM does not care about the order of the rank. The metric is calculated as the size of the output list (set) minus the size of the intersection set between the two corresponding output lists (sets) of any two image inputs. For the above BA graphs, CuHAM(BA(40,1,26), BA(40,1,5))) = 3. If the output sets of any two input images only differ in the rank of the output image list and not in the content, then the CuHAM metric score is 0, which would be the same CuHAM score for two copies of the same input images. However, if the set of images in the output list for two input images had no element in their intersection set, then their CuHAM score would be 10. A larger CuHAM score implies that the input image types are farther apart from each other.

Based on these metric, we draw a clustering graph by denoting each input image with a node in the graph, and their edge with a weight equal to the HAM/CuHAM classification score. Figure 2 shows one such clustering graph based on CuHAM metric for a set of BA, ER, Clique, and Star Graphs. As one can see in Figure 2, Caffe does a pretty good job of clustering each type of graphs into its own cluster corresponding to a classification group. Moreoever, it even has the capability of separating 40-node ER graphs (middle of the graph cluster) from 20-node ER graphs (top-most cluster). This shows a promising potential for Caffe to be used as a deep graph learning tool.

D. Limitations

Because Caffe takes network graph images as input, if any two graph inputs are very dense due to their network size or network (in/out)-degree, Caffe fails at classifying the graphs properly. Caffe is really good at discriminating between different contours and shape of the objects in an image,

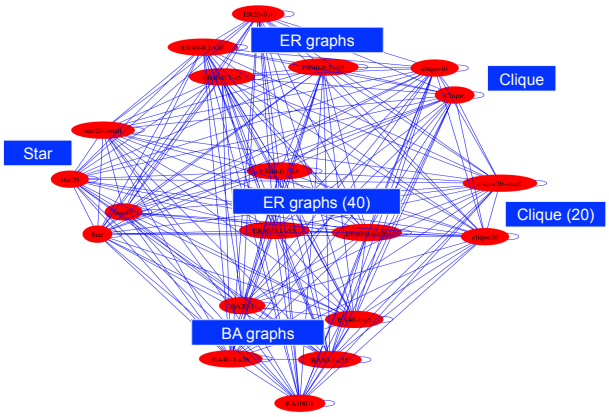

Figure 2: CuHAM-based Caffe Classification of different network graphs.

but they must be sparse enough. With increasing graph size, the ability to discriminate the image reduces, limiting the feasibility of our approach. Sparse representation of dense graphs and their structure-preserving embedding is a topic of our future work under this effort. Also, our approach requires that the exact same embedding code is used for all graphs.

V. Graph features with SCNN

In this section, we demonstrate graph classification using the Search Convolutional Neural Network (SCNN) framework developed by Atwood and Towsley [7]. The SCNNs have a novel architecture which extends convolutional neural networks to general graph-structured data. Features defined through local graph traversals like breadth-first search and random walks are used to learn latent representations that capture local behavior in graph-structured data. Furthermore, the parameters of the model are independent of the size of the input graph.

A. Problem

We demonstrate that the SCNNs are capable of distinguishing graphs due to various generative models, in particular ER and BA graphs [22]. We formalize the problem as follows.

Problem definition: Given an input graph $G = (V, A)$, where V is an ordered set of n vertices and A is an adjacency matrix, it needs to be classified as a realization of a generative model from a list of predetermined models (here, ER and BA).

We report results on graphs of different sizes and to make the classification harder, we choose the parameters such that the expected number of edges in both graph models is the same. More precisely, we set the parameter m of the BA model to be 1, and the parameter p of the ER model to $\frac{2}{n}$.

B. Approach

The methodology involves exploring neighborhood of nodes, using a breadth first search or random search, and classifying them. This leads to a classification of nodes between various generative models. We further classify graphs by choosing the class which obtains majority among the node classes.

The input to the SCNN is the graph adjacency matrix A of dimension $n \times n$, and a matrix of node attributes F of dimension $n \times f$, where f is the number of attributes for a particular node. For our purpose, the node attributes are local network properties such as degree, local clustering coefficient [22], etc.

The SCNN consists of an input layer that feeds in the graph data (A, F), and the hidden search-convolutional layer which performs a graph search and weights the attributes of neighboring nodes in accordance to the hop distance and the feature values. We denote the maximum hops of any search as h. Next, we have a fully connected layer from the hidden search-convolutional layer to the output layer which gives the node classes. The class of the graph is obtained by choosing the majority among the node classes.

Consider a node classification task where a label is predicted for each input node in a graph. Let A^{pow} be an $h \times n \times n$ tensor containing the power series of A, i.e., $A^{pow}_{ijl} = 1$, if there exists a path of i hops between nodes j and l. The convolutional activation z_{ijk} for node i, hop j and feature k is given by

$$z_{ijk} = g\left(W_{jk} \cdot \sum_{l=1}^{n} A^{pow}_{jil} F_{lk}\right),$$

where g is the nonlinear activation function. The model only consists of $O(h \times f)$ parameters, making the search-convolutional representation independent of the size of the input graph, thus enabling transfer learning between graphs.

C. Results

The performance of classification is measured through certain metrics described below. Since, we deal with binary classification in this study, we can assume that we have two hypotheses H_0 and H_1, where we set hypotheses $H_0 = \{\text{ER graph}\}$ and $H_1 = \{\text{BA graph}\}$. Let $N_{i|j}$ be the number of times hypothesis i is reported when hypothesis j is true.

The false alarm rate (F.A.R.) is the proportion of times H_1 is reported when H_0 is true. Therefore,

$$\text{F.A.R.} = \frac{N_{1|0}}{N_{1|0} + N_{0|0}}.$$

Missed detection rate (M.D.R.) is the proportion of times H_0 is reported when H_1 is true, i.e.,

$$\text{M.D.R.} = \frac{N_{0|1}}{N_{0|1} + N_{1|1}}.$$

Furthermore, we use the precision and recall metrics. Precision is the number of correct H_1 results divided by the number of total H_1 results, while recall is the number of correct H_1 results divided by the number of H_1 results that should have been returned. F-score is defined as the harmonic mean of precision and recall.

The dataset consisted of 10 ER graphs and 10 BA graphs with $n = 40, 80, 100, 200,$ and 500 nodes. Two-thirds of the dataset is used for training the SCNN while the rest is used for testing. The node attributes are the degrees and the local clustering coefficients. The classification results averaged over 50 runs are reported in Table II. We observe that the SCNN

No. of Nodes	F.A.R.	M.D.R.	F-Score
n=40	0.0	0.026	0.98
n=80	0.0	0.29	0.82
n=100	0.0	0.31	0.81
n=200	0.0	0.0	1
n=500	0.0	0.0	1

Table II: Results on classification of Erdös-Rényi (ER) and Barabási-Albert (BA) graphs without degree assortativity

performs well for small graphs, deteriorates as the size of the graph increases but again improves and gives perfect classification results for large graphs.

Next, we include a local measure of degree assortativity as node attribute. For every node, we define the local degree assortativity to be the assortativity of the subgraph consisting of edges in its 1-hop neighborhood. On including local degree assortativity as a node attribute, we obtain perfect graph classification for all graph sizes $(n = 40, 80, 100, 200, 500)$.

VI. Representation Learning for Networks

In this section, we investigate how the recent advances of representation learning in natural language processing can be leveraged to further network analysis and graph mining tasks. One natural way is to map the way that people choose friends and maintain connections to a kind of "social language". The objective is to learn latent representations that capture the internal relations from rich, complex network data.

A. Problem

While homogeneous networks or networks with singular types of nodes or edges have been considered in the literature, we posit that heterogeneous networks present novel challenges. To that end, we consider a heterogeneous network as the input to the task. Similar to the definition in [23], we define a heterogeneous network as a graph in which each node v_i and edge e_{ij} are also associated with their mapping functions $\phi(v_i)$ and $\varphi(e_{ij})$, respectively, indicating their object and relation types.

For example, one can represent the academic network with either papers as nodes, where edges indicate the citation relationships, or venues as nodes, where edges denote the overlapping among authors. Another example of a heterogeneous network is a set of nodes of the same type that are associated with different types of relationships — two authors can share a relationship on the virtue of their collaboration on a paper or attending a conference together or even being at the same institution. Thus, in this case there are three possible edge types between nodes (authors).

By considering the heterogeneous network as the input, we formalize the problem of network representation learning as follows.

Problem definition: Given a heterogeneous network the task of deep representation learning is to learn the d-dimension latent representations $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ of the node set V that are able to capture the structural relations among them.

The output of the problem is the low-dimension matrix $\mathbf{X}$, with the i^{th} row—a d-dimension vector—corresponding to the representations of the node v_i. Notice that, although there are different types of objects in V, their representations are mapped into the same latent space. In doing so, the latent representations can be used to solve network analysis tasks involving different types of objects. For example, for the link prediction task in a heterogeneous academic network G [24], [23], in which authors, venues, publications, and terms comprise the node set V, we are able to estimate the probability that a link exists between not only nodes with the same type, such as two authors, but also nodes with different types, such as one author and one conference.

With the setting of the problem, the goal is to design a neural network based framework that learns the deep representations of objects in a heterogeneous network, from which other network analysis tasks would benefit. For example, the learned vector of each node can be used as the feature (latent) input of the node classification [14], community detection [25], link prediction [26], and similarity search [23] models in heterogeneous networks.

B. The node2vec Framework

Given a text corpus, Mikolov et al. proposed the *word2vec* framework to learn the distributed representations of words in the corpus [8], [9]. Specifically, the core of *word2vec* is the Skip-gram model, which learns the representation of a word that facilitates the prediction of its surrounding words (context) in a sentence. The underlying assumption lies in the fact that geographically close words—a word and its context—in a sentence or document exhibit interrelations in human natural language.

More recently, a number of Skip-gram based network representation learning methods were proposed. Essentially, these methods aim to map the word-context concept in a corpus into a network. Perozzi et al. first proposed to use the paths generated by random walks in the network to simulate the sentences in natural language [14]. Tang et al. further decomposed a node's context into first- (friends) and second-order (friends' friends) proximity [15].

Inspired by *word2vec* and other methods, we propose a general framework, *node2vec*, that is capable of learning desirable node representations in a heterogeneous network. The input to this framework will be the heterogeneous network and the output is the latent-space vectors for its nodes that are learned by the model. Similar to *word2vec*, the objective of the *node2vec* method is to maximize the network probability [9], [27].

C. Preliminary Evaluation

We use the DBIS dataset that was used in PathSim [23]. It covers publications from the database and information system area, which was extracted from the DBLP dataset in Nov. 2009. From this publication set, we construct a heterogeneous network, in which there are there types of nodes – authors, papers, and venues. The edges represent a type of relationship among these set of nodes — such as collaboration on a paper or presence at a conference.

Table III: **Case study of conference similarity search.**

Rank	KDD	SIGMOD	SIGIR	WWW
0	KDD	SIGMOD	SIGIR	WWW
1	SDM	PVLDB	TREC	CIKM
2	ICDM	ICDE	CIKM	SIGIR
3	DMKD	TODS	Inf. Proc. Mag.	KDD
4	KDD Exp.	VLDBJ	Inf. Retr.	ICDE
5	PKDD	PODS	ECIR	TKDE
6	PAKDD	EDBT	TOIS	VLDB
7	TKDE	CIDR	WWW	TOTT
8	CIKM	TKDE	JASIST	SIGMOD
9	ICDE	ICDT	JASIS	WebDB

Table IV: **Case study of author similarity search.**

Rank	J. Han	M. Franklin	C. Faloutsos	M. Stonebraker
0	J. Han	M. Franklin	C. Faloutsos	M. Stonebraker
1	J. Pei	S. Madden	C. Aggarwal	D. Dewitt
2	H. Wang	D. Dewitt	P. Yu	S. Madden
3	C. Aggarwal	D. Vaskevitch	R. Agrawal	M. Carey
4	P. Yu	J. Berg	J. Pei	H. Wong
5	C. Faloutsos	N. Bruno	J. Han	M. Franklin

With the DBIS heterogeneous network as the input, the *node2vec* framework learns the latent representations of authors and venues. They can be leveraged to solve other tasks in network analysis and graph mining. Here we show two case studies of similarity search in the DBIS network. The similarity between two nodes is simply computed by the cosine similarity of their latent-space vectors learned by *node2vec*.

Tables III and IV list the top results for querying several well-known authors and conferences. One can observe that for query "KDD", *node2vec* returns venues with the same focus at the top positions, i.e., data mining, such as SDM (1), ICDM (2), DMKD (3), KDD Explorations (4), PKDD (5), PAKDD (6), and those with similar topics, such as TKDE (7), CIKM (8), and ICDE (9). Similar performance can be also achieved when querying other conferences and authors, such as SIGMOD, SIGIR, and WWW.

This is work in progress as we continue to develop the model and evaluate on different use-cases for predictive modeling, similarity, and classification in heterogeneous networks.

VII. Conclusion

Deep neural networks have recently provided significant lift to recognition and classification of images, speech, video, etc. However, their application to military network analysis, for instance for the detection of adversarial activity, is still in its infancy. In this position paper, we have presented three approaches that explore the efficacy of deep learning to network analysis. While still very preliminary, we believe that our early results show that deep learning has the potential for representation learning in networks.

Future directions include experimenting with increased number of layers, using graph "motifs" in the search convolution, and training with a larger corpus of data, including synthetic and real-world data. Fundamentally new neural network architectures that are explicitly designed for or leverage graph-structured data is another exciting new direction.

References

[1] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *An MIT Press book in preparation. Draft chapters available at http://www. iro. umontreal. ca/ bengioy/dlbook*, 2015.

[4] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv:1506.05163*, 2015.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv:1312.6203*, 2013.

[6] "Caffe." [Online]. Available: http://caffe.berkeleyvision.org/

[7] J. Atwood and D. Towsley, "Search-convolutional neural networks," *arXiv:1511.02136*, 2015.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv:1301.3781*, 2013.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[11] G. Li, M. Semerci, B. Yener, and M. J. Zaki, "Effective graph classification based on topological and label attributes," *Statistical Analysis and Data Mining*, vol. 5, no. 4, pp. 265–283, 2012.

[12] T. Horváth, T. Gärtner, and S. Wrobel, "Cyclic pattern kernels for predictive graph mining," in *Proc. ACM SIGKDD*, 2004, pp. 158–167.

[13] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[14] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[15] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding." in *WWW*. ACM, 2015.

[16] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," *International conference on document analysis and recognition*, vol. 3, pp. 958–962, 2003.

[17] "Theano." [Online]. Available: http://deeplearning.net/software/theano/

[18] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," IDIAP, Tech. Rep., 2002.

[19] "Tensorflow." [Online]. Available: https://www.tensorflow.org/

[20] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of Caffe, Neon, Theano, and Torch for deep learning," *arXiv:1511.06435*, 2015.

[21] "Networkx." [Online]. Available: https://networkx.github.io/

[22] M. Newman, *Networks: an introduction*. OUP Oxford, 2010.

[23] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "PathSim: Meta path-based top-k similarity search in heterogeneous information networks," in *PVLDB '11*, 2011, pp. 992–1003.

[24] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *KDD '08*, 2008, pp. 990–998.

[25] Y. Sun, B. Norick, J. Han, X. Yan, P. S. Yu, and X. Yu, "Integrating meta-path selection with user-guided object clustering in heterogeneous information networks," in *Proc. ACM SIGKDD '12*, 2012, pp. 1348–1356.

[26] Y. Dong, J. Zhang, J. Tang, N. V. Chawla, and B. Wang, "CoupledLP: Link prediction in coupled networks," in *Proc. ACM SIGKDD*, 2015, pp. 199–208.

[27] Y. Goldberg and O. Levy, "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method," *CoRR*, vol. abs/1402.3722, 2014.