Neural Structured Learning: Training Neural Networks with Structured Signals

Arjun Gopalan
arjung@google.com
Google Research
USA

Chun-Sung Ferng
csferng@google.com
Google Research
USA

Philip Pham
phillypham@google.com
Google Research
USA

Da-Cheng Juan
dacheng@google.com
Google Research
USA

Allan Heydon
aheydon@google.com
Google Research
USA

George Yu
georgeyu@google.com
Google Research
USA

Yueqi Wang
yueqi@google.com
Google Research
USA

Cesar Ilharco Magalhaes
magalhaesc@google.com
Google Research
Switzerland

Chun-Ta Lu
chunta@google.com
Google Research
USA

Yicheng Fan
yichengfan@google.com
Google Research
USA

ABSTRACT

We present Neural Structured Learning (NSL) in TensorFlow [1], a new learning paradigm to train neural networks by leveraging structured signals in addition to feature inputs. Structure can be explicit as represented by a graph, or implicit, either induced by adversarial perturbation or inferred using techniques like embedding learning. NSL is open-sourced as part of the TensorFlow [2] ecosystem and is widely used in Google across many products and services. In this tutorial, we provide an overview of the NSL framework including various libraries, tools, and APIs as well as demonstrate the practical use of NSL in different applications. The NSL website is hosted at www.tensorflow.org/neural_structured_learning, which includes details about the theoretical foundations of the technology, extensive API documentation, and hands-on tutorials.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks; Regularization; Semi-supervised learning settings.**

KEYWORDS

Neural networks; Structured signals; Graph learning; Adversarial learning; Regularization; TensorFlow

ACM Reference Format:

Arjun Gopalan, Da-Cheng Juan, Cesar Ilharco Magalhaes, Chun-Sung Ferng, Allan Heydon, Chun-Ta Lu, Philip Pham, George Yu, Yicheng Fan, and Yueqi Wang. 2021. Neural Structured Learning: Training Neural Networks with Structured Signals. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel*. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3437963.3441666>

1 INTRODUCTION

Neural Structured Learning (NSL) [1] is a new learning paradigm to train neural networks by leveraging structured signals in addition to feature inputs. Structured signals are commonly used to represent relations or similarity among samples that may be labeled or unlabeled. So, leveraging these signals during training harnesses both labeled and unlabeled data, which can improve model accuracy, particularly when the amount of labeled data is relatively small. In NSL, structure can be explicit as represented by a graph, or implicit, either induced by adversarial perturbation or inferred using techniques like embedding learning. Models trained with samples that are generated by adding adversarial perturbation have been shown to be robust against malicious attacks, which are designed to mislead a model's prediction or classification. NSL generalizes to both Neural Graph Learning [5] as well as Adversarial Learning [7, 11].

NSL is part of the TensorFlow [2] ecosystem and is open-sourced at <https://github.com/tensorflow/neural-structured-learning>. The NSL framework in TensorFlow provides the following easy-to-use APIs and tools to train models with structured signals:

- Keras and Estimator APIs to enable training with graphs (explicit structure) and adversarial perturbations (implicit structure)

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

WSDM '21, March 8–12, 2021, Virtual Event, Israel

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8297-7/21/03.

<https://doi.org/10.1145/3437963.3441666>

- TensorFlow operations and functions to enable training with structured signals when using lower-level TensorFlow APIs
- Tools to build graphs and prepare graph inputs for training

In this tutorial, we will cover several aspects of NSL with an emphasis on two techniques: graph regularization and adversarial regularization. In addition to using interactive hands-on tutorials that demonstrate the NSL framework and APIs in TensorFlow, we also plan to have short presentations to provide additional context and motivation. Finally, we will discuss recent research that is closely related to Neural Structured Learning but not yet part of the framework in TensorFlow. Topics here will include using graphs for learning embeddings [3, 8] and other advanced models of graph neural networks [14, 17].

2 TUTORIAL OUTLINE

In this section, we describe the outline for the 3-hour long tutorial. Section 3 gives an overview of the Neural Structured Learning framework. Section 4 broadly covers *Graph Learning* including data preprocessing for graph regularization (section 4.1), regularization with natural graphs (section 4.2), regularization with synthesized graphs (section 4.3), scalable graph learning using historical embeddings (section 4.4), graph neural networks (4.5), and neural clustering processes (section 4.6). Section 5 covers *Adversarial Learning*, another key facet of Neural Structured Learning. Section 6 covers how NSL models can be deployed in production using *TensorFlow Extended* (TFX) pipelines. Section 7 covers recent research in NSL as well as future directions, and finally, section 8 concludes.

3 NSL OVERVIEW

We will begin the tutorial with an overview of the Neural Structured Learning framework and motivate the benefits of training neural networks with structured signals. These structured signals are used to regularize the training of a neural network, forcing the model to learn accurate predictions as well as maintain the structural similarity that exists in the input. This technique is generic and can be applied on arbitrary neural network architectures (such as feed-forward neural networks, convolutional neural networks and recurrent neural networks). Using NSL offers several advantages including but not limited to:

- **Higher accuracy:** The structured signals among samples can provide information that is not always available with typical feature inputs. The joint training approach with both structured signals and features has been shown to outperform many existing methods (that rely on training with features only) on a wide range of tasks such as document classification and semantic intent classification [5, 9].
- **Model Robustness:** Models trained with adversarial examples have been shown to be robust against adversarial perturbations that are designed to mislead a model's prediction [7, 11]. When the number of training samples is small, training with adversarial examples also helps improve model accuracy [15].
- **Training with limited labels:** NSL enables neural networks to leverage both labeled and unlabeled data, which extends the learning paradigm to semi-supervised learning. Specifically, NSL allows the network to train using labeled

data as in the supervised learning setting, and at the same time drives the network to learn similar hidden representations for the *neighboring samples* that may or may not have labels.

4 GRAPH LEARNING

This section will cover several topics related to graph learning with Neural Structured Learning.

4.1 Data Preprocessing

Doing graph-based regularization in NSL can be divided into two parts: preparing the training data based on a similarity graph and training a model with it subsequently. This section of the tutorial focuses on the data preprocessing tools required to accomplish the first part of the process.

There are two inputs to this process: 1) training instances represented by `tensorflow.Examples` (some of which are labeled) and 2) a *similarity graph* $G = (V, E)$ defined on the training examples V . The edges of the graph E connect pairs of similar training instances. If a natural organic similarity graph — based, for example, on natural links like connections in a social network or citations among papers — already exists, it can be encoded in a simple tab-separated value (TSV) format. Alternatively, NSL's *graph builder* tool can be used to construct a graph based on dense or sparse embeddings associated with the data instances.

The brute-force approach to constructing a similarity graph from instance embeddings is $O(n^2)$, which does not scale well to large training sets. To mitigate that problem, our open-source graph builder uses *locality-sensitive hashing* (LSH) [6] to partition the training examples into smaller buckets of similar instances, then performs the all-pairs similarity computation among the instances in each bucket. The tutorial describes this process as well as how repeated applications of it using randomized LSH functions can be used to increase edge recall. We describe the graph builder tool and its parameters, and present an example demonstrating the effects of various parameters on the time required for graph construction.

Once a similarity graph has been specified or constructed, the next step in NSL data preprocessing is to associate with each labeled input instance one or more of its neighbors in the similarity graph. NSL includes a *pack neighbors* tool for accomplishing this. We describe the tool and its parameters.

Both the *graph builder* and *pack neighbors* tools can be invoked as Python libraries or run as stand-alone binaries. Finally, we also describe the NSL libraries for reading and writing graphs in TSV format, the internal graph representation, and utility functions for low-level graph operations.

4.2 Regularizing with Natural Graphs

Graph regularization [5] allows neural networks to learn similar predictions or representations for entities connected to each other in a similarity graph. *Natural graphs* or *organic graphs* are sets of data points that have an inherent relationship between each other. We will demonstrate via a practical tutorial, the use of natural graphs for graph regularization and modern natural language processing models to classify the veracity of public social media posts.

4.3 Regularizing with Synthesized Graphs

Input data may not always be represented as a graph. However, one can infer similarity relationships between input instances and use that to subsequently build a similarity graph. We refer to such graphs as *synthesized graphs*. While graphs can be built in many ways, we will make use of embeddings to build a graph in this tutorial.

As mentioned in Section 4.1, the NSL framework offers a graph building tool that can be used to build graphs using embedding similarity. We will consider a text classification task, and demonstrate graph building and subsequent graph regularization using a practical tutorial. This tutorial will involve the following steps:

1. Converting text to embeddings using a publicly available pre-trained text embedding model
2. Using the text embeddings to build a similarity graph using NSL's graph builder
3. Using the similarity graph and the text input to create a graph-regularized model for the text classification task.

4.4 Scalable Graph Learning with Historical Embeddings

There has been a surge in graph learning techniques recently including but not limited to graph regularized neural networks, graph convolution networks and graph attention networks. However, they often don't scale very well to large data sets, especially for deep neural networks. A major limiting factor for scalability is the linear dependence on the size of the neighborhoods used in training.

In this part of the tutorial, we will introduce a new framework that enables efficient graph operations for large-scale graph learning. It addresses the linear dependence on the neighborhood size by using *historical embeddings*, i.e., embeddings from a previous time step in training. Embeddings are stored in a remote storage system and their freshness is maintained by dedicated processes, which load the latest model checkpoint from training, and use it to compute and update embeddings in a distributed manner. Neighborhood information and historical embeddings are retrieved from the remote storage system as part of input preprocessing, rather than recomputed repeatedly across multiple training batches.

This framework allows us to train large-scale graph-based deep neural networks where the training time is mostly invariant to the neighborhood size. It also enables the use of dynamic graphs or neighborhoods in training.

4.5 Graph Neural Networks with NSL

Graph Neural Networks (GNNs) [9, 16, 18] have become popular methods to train neural networks leveraging graph-structured signals. NSL provides libraries and examples demonstrating how to build commonly-used GNN models such as Graph Convolutional Networks [9], Graph Attention Networks [16], and Graph Isomorphism Networks [18]. We may highlight some of these during our tutorial.

4.6 Neural Clustering Processes

Clustering has traditionally been posed as an unsupervised learning problem. However, unsupervised clustering methods often rely on

strong assumptions about data geometry, and the clusters produced might not agree with human labels. Recently, supervised formulations of clustering have gained increasing interest, with novel objective functions and neural network architectures that directly learn complex cluster objectives from labeled data. In this tutorial, we will demonstrate supervised neural clustering methods [12, 13], such as *Neural Clustering Process (NCP)*, that can be trained end-to-end to perform probabilistic clustering on both generic data points and graph-structured data without assuming a fixed or maximum number of clusters.

5 ADVERSARIAL LEARNING

When similarity relationships between input instances are not available or hard to define, one can generate synthetic neighbor instances that are close to the input instances and use them in addition during training. One approach of incorporating such synthetic instances is adversarial learning, which involves construction of similar but difficult instances for models to predict. These constructed instances are intended to mislead the model into making incorrect predictions. So, by training with these instances, models learn to generalize better and become robust to slight perturbations in the input.

Adversarial learning has been shown to be effective in improving the accuracy of a model as well as its robustness to adversarial attacks. We will demonstrate adversarial learning techniques [7, 10, 11] like *Fast Gradient Sign Method (FGSM)* and *Projected Gradient Descent (PGD)* for image classification using a practical tutorial.

6 NSL IN TENSORFLOW EXTENDED (TFX)

The preceding sections discuss how NSL can be used with native TensorFlow. While this may serve the needs of many users, organizations will often want to create production workflows for their machine learning pipelines. TensorFlow Extended (TFX) [4] is a TensorFlow-based platform for automating, managing, and auditing machine learning pipelines for production. In this part of the tutorial, we will discuss how Neural Structured Learning can be integrated with TFX pipelines.

7 RESEARCH AND FUTURE DIRECTIONS

Neural Structured Learning can be viewed as a broad learning paradigm that combines the use of neural networks and structured signals in input data. This session will include a presentation to discuss recent research in areas related to NSL as well as future directions in research and development. Topics here may include:

- Using graphs to learn embeddings [3, 8]
- Building advanced models of graph neural networks [14, 17]

We will also share details on academic and industrial collaboration opportunities for NSL. Open source contributions to NSL are highly welcome.

8 CONCLUSION

In this tutorial, we provide an overview of Neural Structured Learning (NSL) in TensorFlow, a new learning paradigm to train neural networks by leveraging structured signals in addition to feature inputs. We also describe the outline of the tutorial session that will include: (a) an overview of the framework, (b) data preprocessing

in NSL which includes graph building, (c) regularization with natural and synthesized graphs, (d) scalable graph learning, (e) graph neural networks, (f) neural clustering processes, (g) adversarial regularization, (e) NSL in TFX, and (f) research and future directions. We look forward to advancing research and development in NSL with the broader machine learning community.

ACKNOWLEDGMENTS

We would like to thank Andrew Tomkins, Yun Zeng, Daniel ‘Wolff’ Dobson, Karmel Allison, Mark Daoust, Billy Lamberta, Yash Katariya, Robert Crowe, and many members from the TensorFlow community for their support and contributions.

REFERENCES

- [1] 2019. *Neural Structured Learning in TensorFlow*. https://www.tensorflow.org/neural_structured_learning
- [2] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 265–283.
- [3] Trapti Bansal, Da-Cheng Juan, Sujith Ravi, and Andrew McCallum. 2019. A2N: Attending to Neighbors for Knowledge Graph Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
- [4] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, Chiu Yuen Koo, Lukasz Lew, Clemens Mewald, Akshay Naresh Modi, Neoklis Polyzotis, Sukriti Ramesh, Sudip Roy, Steven Euijong Whang, Martin Wicke, Jarek Wilkiewicz, Xin Zhang, and Martin Zinkevich. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1387–1395.
- [5] Thang D. Bui, Sujith Ravi, and Vivek Ramavajjala. 2018. Neural Graph Learning: Training Neural Networks Using Graphs. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 64–71.
- [6] Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proc. of 34th STOC*. ACM, 380–388.
- [7] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.
- [8] Da-Cheng Juan, Chun-Ta Lu, Zhen Li, Futang Peng, Aleksei Timofeev, Yi-Ting Chen, Yaxi Gao, Tom Duerig, Andrew Tomkins, and Sujith Ravi. 2020. Ultra Fine-Grained Image Semantic Embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 277–285.
- [9] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net. <https://openreview.net/forum?id=SJU4ayYgl>
- [10] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
- [11] Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), 1979–1993.
- [12] Ari Pakman, Yueqi Wang, Yoonho Lee, Pallab Basu, Juho Lee, Yee Whye Teh, and Liam Paninski. 2020. Attentive Clustering Processes. *arXiv:2010.15727* [stat.ML]
- [13] Ari Pakman, Yueqi Wang, Catalin Mitelut, Jinhyung Lee, and Liam Paninski. 2020. Neural Clustering Processes. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, Virtual, 7455–7465. <http://proceedings.mlr.press/v119/pakman20a.html>
- [14] Otilia Stretcu, Krishnamurthy Viswanathan, Dana Movshovitz-Attias, Emmanouil Platanios, Sujith Ravi, and Andrew Tomkins. 2019. Graph Agreement Models for Semi-Supervised Learning. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8713–8723.
- [15] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness May Be at Odds with Accuracy. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*. OpenReview.net. <https://openreview.net/forum?id=SyxAb30cY7>
- [16] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [17] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020), 1–21.
- [18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).