Analysis and Optimization of Security Infrastructure with Deep Learning Methods

Franklin E. Diaz
fdiaz@paloaltonetworks.com[1,2]

[1]Palo Alto Networks
[2]Professional Services - Automation

November 28, 2021

Abstract

Our teams rely heavily on the Terraform software made by Hashicorp as a means of automating network infrastructure deployments in our labs and customer environments. Our teams also develop Open Source Terraform modules. These deployments and modules take the form of graph-structured data. This paper describes efforts to apply Graph neural networks (GNNs), a form of deep learning, to perform analysis of this infrastructure automation and determine the likelihood of computationally improving it.

1 Introduction

A cursory consideration of the technology we are using to define and deploy network infrastructure reveals a declarative use of graphing as a descriptive language and in a definitive manner. Given this inherent use of graphs, and a seemingly meteoric rise in the availability of Artificial Intelligence research and tooling, it feels like a logical next step to transform these designs into a computationally suitable format. The use of Python, open source modules, and the proliferation of scholarly papers and tutorials is a powerful combination indeed. As the common refrain goes, gathering and making data suitable for analysis may be the largest challenge facing the would-be data scientist.

With these ideas in mind, this paper describes attempts to realize improvements in security and efficiency related to automated deployment of network infrastructure. That is to say, capturing attempts to understand what's possible and find the edges of the problem space while learning new things is the true goal. Usable, useful or even interesting results are a happy accident

Where possible, code samples and screenshots of resultant execution are provided in an attempt to make the material accessible to a greater number of readers. Studying and working in what is considered a sub-field of artificial intelligence reveals a limited number of folks engaging in these methodologies.

2 A Very Brief Primer on Graphing and AI

While an in depth explanation of graph theory is beyond the scope of this paper, some background will be provided here to make the paper accessible to a wider audience. A collection of related references will be included at the end of this paper for those who may be interested.

2.1 About Graphs

Use of the term graph in this paper simply means a representation of information in a way that describes a group of things, which we will refer to as "nodes". A relationship between nodes is called an "edge". Graphs have many properties, most relevant of which to us include whether a graph has directionality, As with many things in computer science, we are affected by problems of scale. It often makes sense to constrain our scope to a given sub-graph or "neighborhood".

2.2 About Deep Learning with Graph Neural Networks

The first item of note is Consider figure 1.

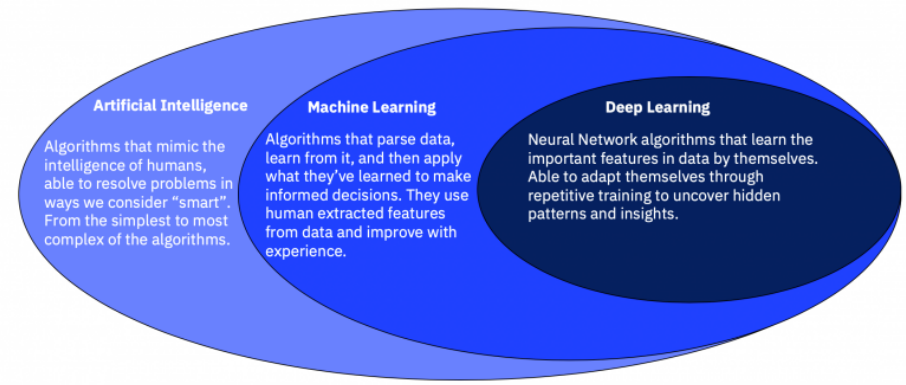

Figure 1: From AI, machine learning and deep learning: What's the difference? by Rodrigo Ceron.

3 Gathering Data

Once a Terraform code base has been initialized, the user has the option to generate a directed graph of the infrastructure. Consider the following declaration of a Cloud Function for Google Cloud.

3.1: Terraform Declaration Example

```
resource "google_project_service" "cloud_function" {
  project                    = var.project_id
  service                    = "cloudfunctions.googleapis.com"
  disable_on_destroy         = false
  disable_dependent_services = false
}
```

Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. DOT is a language used to describe graphs. Conversion of Terraform HCL to a directed graph representation is trivial thanks to Graphviz and Dot.

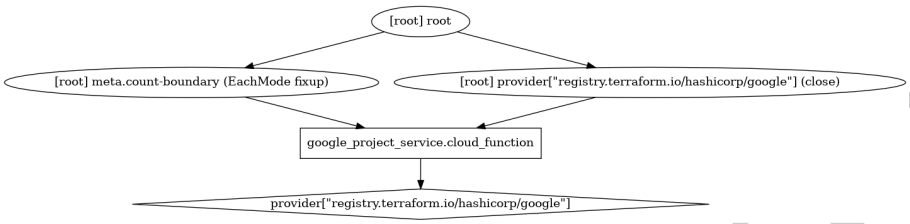

Figure 2: DOT output from Terraform Files as a digraph

Once our Terraform files have been output as a directed graph, we need to bring it in to Python so we can perform operations on it. To run Terraform commands in an automated fashion, the python-terraform module is used.

Directed graphs are ingested into Python as an object using the networkx module. Now we are ready to perform operations on the digraph, displayed in figure 3.

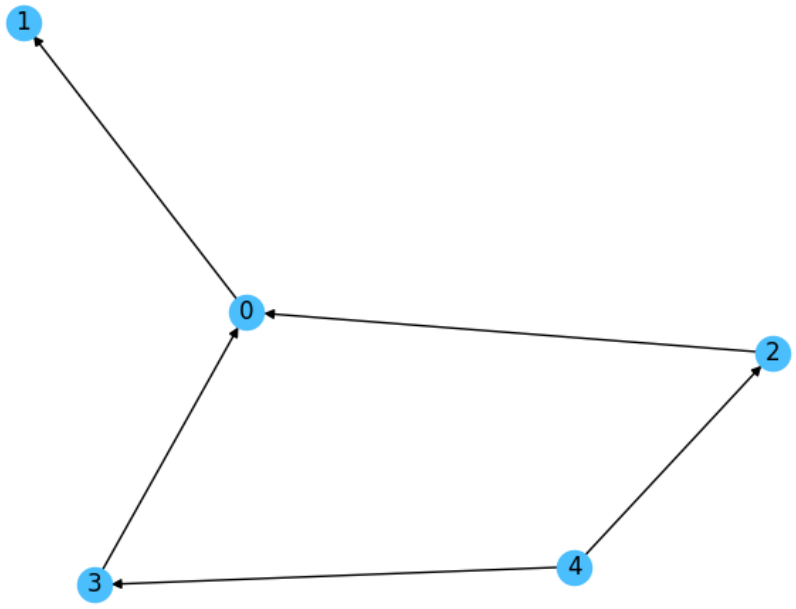

Figure 3: Directed graph as ingested by Networx

Note that new numerical labels are displayed in place of the original labels. These original labels are stored in the node objects in case we need them later. The root node of the adjacency matrix is labeled as the 0th element in the graph. Also of note is that the graphs displayed in figure 2 and figure 3 can be

described as isomorphic.

3.1 The Adjacency Matrix

The adjacency matrix is a mathematical representation of our graph. The nodes are represented in a columnar format known as a matrix. Continuing with our previous example, the representation of our

3.2 Node Embedding

Assigning weights and features to the edges of the graph.

Revision History

Revision	Date	Author(s)	Description
v0.1	Nov. 20th, 2021	Franklin Diaz	Initial Draft

References

[1] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *CoRR*, abs/1912.12693, 2019.

[2] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *CoRR*, abs/1709.05584, 2017.