# Supervised Hyperbolic Representation Learning for Real-World Networks

**Manan Shah**
manans@stanford.edu

**Sagar Maheshwari**
msagar@stanford.edu

## Abstract

Graphs are inherently complex structures, and learning suitable representations from these dynamic networks for downstream prediction tasks is a canonical problem with great practical implications. Indeed, the generation of node embeddings that accurately capture the structure and high dimensionality of a graph may prove incredibly useful in the development of models for traditional prediction and classification tasks across nodes and edges. In this work, we focus on two critical components of representation learning on graphs: the efficient and effective generation of node embeddings, and the use of such embeddings for node classification and link prediction. In particular, our work aims to leverage recent developments of representation learning in hyperbolic space to provide two significant contributions to the field:

1. The development and empirical evaluation of a *supervised* algorithm to embed graphs in hyperbolic space, providing significant improvements over recent unsupervised methods.
2. The development of a comprehensive multimodal framework to combine both Euclidean and hyperbolic embeddings for increased representational power on node classification and link prediction tasks, further improving results beyond current state-of-the-art performance.

We provide an open-source implementation of our findings, embedding frameworks, and visualization methods at `www.github.com/mananshah99/hyperbolic`.

## 1 Introduction

Graphs are powerful tools to encode relationships between diverse objects and process unstructured real-world data. However, while there are numerous benefits to such complexity, the diverse nature of graphs has resulted in difficulty analyzing networks with varying size, shape, and structure. Furthermore, conducting downstream prediction tasks (e.g. regression or hierarchy generation) from arbitrary nodes, edges, or substructures in a graph requires fixed-dimensional representations of these structures, a task which is made nontrivial due to the arbitrarily complex nature of networks. The generation of embeddings for nodes in a graph provides an effective and efficient way to approach the analysis of graphical structures while retaining the diversity encoded in the graph itself. Precisely, given graph $G(V, E)$, the canonical formalization of the node embedding task is to learn a function $f|_{v \in V}$ such that $f : v \to X^d$ for some metric space $(X, s)$ equipped with suitable metric $s$, and dimension $d$. A suitably learned embedding function $f$ has a desired property of mapping similar nodes close to one another in the embedding space $X^d$. The resulting fixed-dimensional embeddings for nodes can be readily used as input feature matrices for downstream tasks, emphasizing the importance of properly capturing graph structure in embedding generation.

Although numerous methods have been previously proposed to learn $f : v \to \mathbb{R}^d$ in both supervised and unsupervised manners, recent works ([11], [10]) suggest learning $f : v \to \mathcal{H}^d$ is more suitable for ensuring that hyperbolic structures are more accurately represented in the embedding vectors. In particular, as hyperbolic space can be viewed as a continuous generalization of discrete tree structures due to its exponential distance properties, hub and authority structures commonly found in real-world networks are more accurately represented in finite dimensions with hyperbolic space than Euclidean space. Our proposed work builds upon existing frameworks for embedding graphs in hyperbolic space and can be segmented into two stages, each with different inputs and outputs. The first stage takes a graph $G = (V, E)$ as input and produces node-level embeddings, for which we develop a novel supervised learning algorithm to embed nodes in hyperbolic space. The second stage uses the aforementioned learned embeddings to perform node classification and link prediction tasks, for which we develop the first-ever hybrid embeddings between hyperbolic and Euclidean spaces to achieve state-of-the-art results. This project is a joint work between CS 229 and CS 224W; while our work for CS 229 focused on the development of a supervised hyperbolic embedding algorithm and hybrid embeddings for node classification and link prediction, our work for

CS 224W leveraged graph hyperbolicity to improve random walks and graph convolutional networks. All results, graphs, and analysis presented in this paper are solely for CS 229, although code between both classes was shared to avoid duplication of node embedding pipelines from graphs.

## 2 Related Work

The problem of embedding nodes in graphs as high-dimensional feature vectors has been explored in varying directions, of which state of the art results have been observed by random walk based approaches DeepWalk [13] and node2vec [4]. However, recent work characterizing the hyperbolicity of graphs has yielded promising results on embedding nodes in hyperbolic space, with [11] and [10] demonstrating significant improvements of hyperbolic embeddings over Euclidean embeddings on limited, tree-like datasets. We discuss both methods in this section.

**Euclidean embeddings.** The first method to employ random walks for embedding nodes in graphs, DeepWalk generates node embeddings by treating random walks as sentences in linguistic models. Inspired by the widespread success of language models in learning latent distributed representations from words, DeepWalk relies on the intuition that random walks can be modeled as sentences in a particular language. Analogous to natural language modeling frameworks that estimate the probability that a word will appear in a sentence, DeepWalk estimates the probability that a vertex appears in a random walk by learning feature vectors $f_v \mid v \in V$. For scalability purposes, the authors use hierarchical softmax to approximate the conditional probabilities during gradient descent. Node2vec generalizes DeepWalk by utilizing both the current node and its predecessor to identify the next node in a random walk. By tuning two parameters $p, q \in \mathbb{R}$ where the former loosely controls depth-first transitions and the latter breadth-first behavior, node2vec is able to interpolate between different kinds of neighborhoods to better incorporate graph structure in embedding generation. node2vec conducts a similar optimization procedure as DeepWalk, sampling vertices $v \in V$, conducting random walks, and updating feature vectors in gradient descent. Due to its ability to smoothly characterize networks according to both the homophily and structural hypotheses for neighborhood formation, node2vec successfully improves performance over DeepWalk across varied benchmarks.

**Hyperbolic embeddings.** Hyperbolic space, which uses non-Euclidean geometries, is characterized by its constant negative curvature. Due to this curvature, distances increase exponentially away from the origin and give hyperbolic spaces the name of continuous analogues of trees. As a result, hyperbolic spaces are critically relevant to model hierarchical data, and distance in hyperbolic space can be used to infer both node hierarchy and similarity. Nickel et al [11] examines the task of learning hierarchical representations of data by embedding in hyperbolic space; more specifically, the $n$-dimensional Poincaré ball. Through empirical studies, the authors discover that Euclidean embeddings are limited in their ability to model intricate patterns in network structures due to the need for high dimensions. Motivated by this finding, the authors formulate a model for generating node embeddings in an $n$-dimensional Poincaré ball. Of the many hyperbolic space models, the authors use the Poincaré model due to its ease-of-use in gradient-based optimization: nodes are initially given random embeddings, which are then updated by minimizing a loss function using Riemannian Stochastic Gradient Descent. The authors use this procedure to generate node embeddings for the WORDNET dataset and a collaboration network to compare performance to Euclidean embeddings on the tasks of lexical entailment and link prediction, respectively. In both tasks, Poincaré embeddings were able to outperform Euclidean embeddings, even with significantly lower dimensions.

Hyperbolic embeddings excel in their encoding of both node similarity and node hierarchy, which allows for unsupervised identification of latent hierarchical structures. In spite of these benefits, however, the field of hyperbolic embeddings is relatively new and as such has significant room for improvement. In particular, all datasets used in [11] and [10] have explicit hierarchical structure, making for promising node embedding structures on the Poincaré disk. However, such embeddings have yet to be trained in a supervised manner to incorporate node label information, and no prior work has considered the combination of both Euclidean embeddings (as in [13] and [4]) with hyperbolic embeddings for classification and prediction tasks. We tackle both such tasks in our work.

## 3 Datasets

As the generation of hyperbolic embeddings is a relatively novel field in representation learning on graphs, a critical problem with recent papers on hyperbolic embeddings ([11] and [10]) was their evaluation of such embeddings on small datasets with clearly defined explicit hierarchies. To remedy this issue and utilize real-world datasets that allow for analysis of representational power, large networks that contain both explicit and latent hierarchies are required for careful analysis. We use these datasets to evaluate our supervised hyperbolic embedding framework and our hybrid embedding generation for node classification and link prediction.

**ChG-miner.** [9] represents a drug-target interaction network with information on which proteins encoded by genes are targeted by drugs on the United States market. Nodes represent drugs and genes, and edges correspond to a drug-target interaction. The network has 12,358 nodes and 15,424 edges, rendering it significantly larger than other networks employed to evaluate hyperbolic embeddings. For classification purposes, each node is classified as either

a drug or a gene. With no explicit hierarchy in drug-target interactions and no explicit hyperbolicity encoded in the graph, ChG-miner represents a dataset that does not have intuitively strong reasons to prefer hyperbolic embeddings over canonical Euclidean embeddings.

**email-Eu-core.** [8] represents an email interaction network between members of a large European research institution. Edges represent communications between institution members, and nodes represent the corresponding members. Additional information is provided regarding the departments of each institution member for ground-truth evaluation on node classification; in particular, each node is assigned an integer class from 1 to 42 indicating the role of the corresponding member in the organization. Furthermore, the presence of a hierarchy is explicitly encoded into the network due to the different titles and roles of individuals sending one another emails as well as the directed nature of the graph. Therefore, the assumptions made in the generation of hyperbolic embeddings are assuredly present in this network, providing intuitively strong reasons to prefer hyperbolic embeddings over Euclidean ones.

In aggregate, these two datasets represent graphs at either end of the spectrum of latent hierarchy presence: while ChG-miner has no encoded hierarchies, email-Eu-core explicitly encodes hierarchies in its construction. As both datasets have defined classes (2 classes of drugs and genes in ChG-miner and 42 classes corresponding to different organizational roles in email-Eu-core), node classification and link prediction can both be conducted on each graph.

# 4 Methods

## 4.1 Hyperbolic Representation Learning

**Supervised Hyperbolic Embeddings.** Leveraging recent groundbreaking work on the generation of unsupervised hyperbolic embeddings, we developed a supervised algorithm to generate node embeddings in hyperbolic space. In order to modify current generation of hyperbolic embeddings to incorporate node labels, we alter the sampling procedure described in [11] to only generate negative samples between differently labeled nodes. We retain the Riemannian SGD update rule, where $\nabla$ represents the Euclidean gradient of the loss function such that

$$\theta_{t+1} \leftarrow \text{proj}\left(\theta_t - \eta_t \frac{(1-\|\theta_t\|^2)^2}{4}\nabla_E\right)$$

In particular, our novel supervised loss function solely incorporates updates from vertices of different classes, so that

$$\mathcal{L}(\Theta) = \sum_{(u,v)\in E} \log \frac{e^{-d(u,v)}}{\sum_{v'\in\mathcal{N}(u)} e^{-d(u,v')}}$$

is defined with the set of negative examples for $u$ equal to $\mathcal{N}(u) = \{v \mid (u,v) \notin E, c(u) \neq c(v)\} \cup \{u\}$ where $c$ denotes a surjective mapping from node to class and $d(u,v)$ denotes the Poincaré distance function between nodes $u$ and $v$. As a result of this supervised modification, embeddings are trained to simultaneously ensure that nodes of similar labels are embedded near one another in hyperbolic space and that nodes connected by an edge are encoded near one another as well. Since the resulting embeddings are mapped to an $n$-dimensional Poincaré ball as in [11], our supervised model retains the conformal property of the Poincaré ball with respect to the hyperbolic space. In particular, this means that Euclidean and hyperbolic angles between vectors in the Poincaré ball are identical, preserving the notion of dot product similarity at inference time.

**Euclidean and Hyperbolic Embedding Fusion.** In addition to the supervised learning algorithm, we introduce the concept of hybrid embeddings that incorporate both Euclidean and hyperbolic embeddings of a graph. Though a viable form of fusion, methods that mathematically combine both types of embeddings (eg. Hadamard product and cosine similarity) are not used due to the scale differences between Euclidean and hyperbolic space. Instead, we simply concatenate Euclidean and hyperbolic embeddings to maximize utility gained from each during downstream prediction tasks without utilizing mathematically incorrect combinations of vectors from different spaces.

## 4.2 Node Classification and Link Prediction

To evaluate our generated embeddings, we measure performance for two major network learning tasks: node classification and link prediction. While logistic regression on generated node representations is employed for node classification, random forest classification on edge embeddings is utilized for link prediction.

**Node Classification.** In the node classification pipeline, note that the feature vector for each node is its fixed-dimension embedding. With feature vectors for each node in the graph, we implement logistic regression optimized using stochastic gradient descent. In logistic regression, our hypothesis has the form $h_\theta(x) = \sigma(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$.

| | Node Classification (Accuracy) | | Link Prediction (Average Precision) | |
|---|---|---|---|---|
| | email-EU-core<br>Train: 804<br>Test: 201 | ChG-miner<br>Train: 5872<br>Test: 1468 | email-EU-core<br>Train: 19278<br>Test: 9638 | ChG-miner<br>Train: 18170<br>Test: 9082 |
| DeepWalk | 0.741 | 0.669 | 0.55 / 0.55 | 0.55 / 0.55 |
| node2vec ($p = q = 0.25$) | 0.761 | 0.693 | 0.57 / 0.55 | 0.57 / 0.56 |
| Supervised Poincaré | 0.905 | 0.686 | 0.49 / 0.51 | 0.51 / 0.51 |
| Hybrid Poincaré + Euclidean | **0.915** | **0.710** | **0.57 / 0.56** | **0.58 / 0.57** |

Table 1: *Overall Embedding Performance.* This table depicts the performance of varying types of embedding schema on node classification and link prediction, with best results typeset in bold. Link prediction results are reported for train and test sets. Sizes of training and test sets for both tasks are additionally reported.

During stochastic gradient descent, we minimize the loss function

$$-\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))$$

from which we can take the gradient to derive the update rule

$$\theta_{j+1} \leftarrow \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$$

After iterating until convergence, we use optimized hypothesis function $h_\theta$ to classify nodes given input feature vectors of arbitrary dimension. Specifically, we split nodes of our graph into train and test sets for 5-fold cross validation, utilizing $h_\theta$ to predict across 42 classes for email-Eu-core and across 2 classes in ChG-miner.

**Link Prediction.** We leverage random forests in place of logistic regression for link prediction due to empirical performance improvements. A random forest $R(x)$ is a classifier consisting of a collection $D$ of decision tree classifiers where $|D| = B$ and each tree $d_b(x) \in D$, $b \in \{1, 2, \cdots, B\}$ casts a unit vote to determine the most popular class for a particular input observation $x$. Given a training set $X$ and corresponding class labels $Y$, we perform bagging $B$ times, each time selecting a random sample of fixed-size from $X$ and fitting a decision tree to the sample using a fixed-size $K$ random sample of the features. When creating decision trees, we aim to achieve maximum homogeneity in child nodes each time the data is divided into two parts, where $t_p, t_l, t_r$ are parent, left child, and right child nodes; $x_i$ is a feature variable; and $x_i^*$ is the best splitting value for that feature. Hence, we split at the optimal split value among all possible split values for all features. To maximize homogeneity of child nodes, we use impurity function $\iota(t)$. Since the impurity of parent node $t_p$ is constant for all possible splits, maximizing homogeneity of child nodes is equivalent to maximizing the change of impurity function $\Delta\iota(t) = \iota(t_p) - P_l\iota(t_l) - P_r\iota(t_r)$, where $P_l$ is the proportion of data distributed to the left node and $P_r$ the right node. Therefore, when partitioning the data at each node of a decision tree, we solve the optimization problem

$$\underset{x_i \leq x_i^*, i \in \{1,2,\cdots,K\}}{\arg\max} \quad \iota(t_p) - P_l\iota(t_l) - P_r\iota(t_r)$$

In particular, we use the impurity function $\iota(t) = \sum_{a \neq b} P(a|t)P(b|t)$, where $a, b$ are indexes of classes $1, 2, \ldots C$ and $P(a|t)$ is the conditional probability of class $a$ given we are in node $t$.

For the link prediction task, we generate $X$ by taking the Hadamard product of source and destination node embeddings to generate edge embeddings for each edge in the graph. We subsequently split our graph into training, validation, and test sets which can be viewed as versions of the graph different time intervals (where we wish to train on edges in time interval 0 and predict future edges to appear in time interval 1). We generate positive training examples by sampling from edges that exist in the graph, and negative examples from edges that are missing between nodes so that there is a perfect class balance between positive and negative examples. With $X$ and corresponding ground-truth edge labels $Y$, we utilize the random forest algorithm to predict existence of edges.

## 5 Results and Discussion

In order to evaluate our proposed supervised hybrid hyperbolic embeddings on the aforementioned datasets in Section 3, we performed 5-fold cross validation for node classification according to the accuracy metric and evaluated on test splits of edges for link prediction according to the average precision metric (Table 1). We further visualized node-level embeddings colored by their respective classes with PCA and t-SNE in 3 dimensions in order to qualitatively analyze the performance of our proposed model over canonical models in the literature (Figure 1).

Table 1 depicts the core results of our work, with DeepWalk and node2vec representing canonical state-of-the-art embedding baselines in Euclidean space. While supervised Poincaré embeddings are able to significantly improve

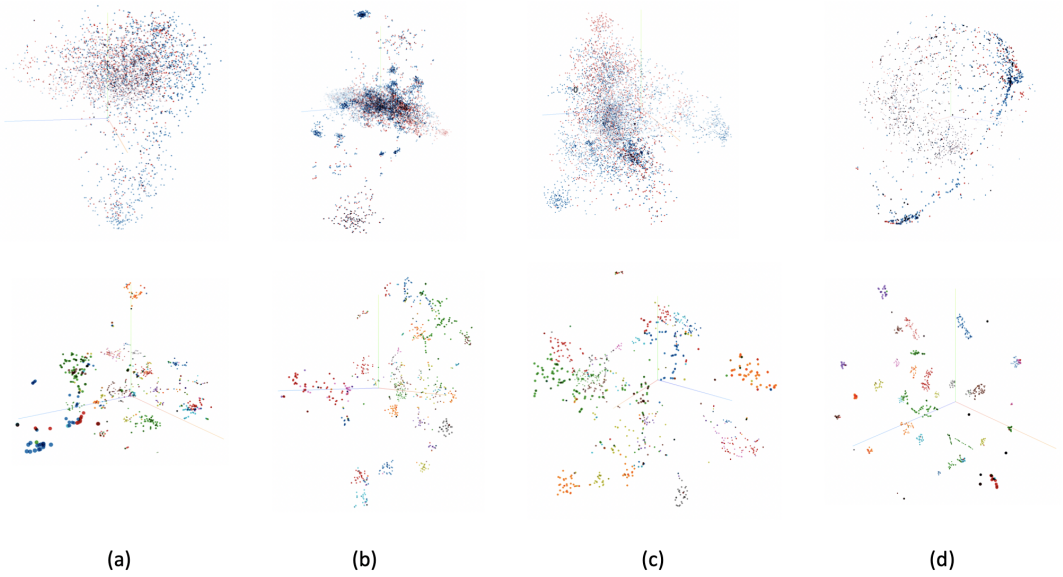


Figure 1: *Embedding Visualizations.* The top row depicts PCA visualizations for embeddings generated on ChG-miner, and the bottom row depicts t-SNE visualizations for embeddings generated on email-Eu-core (both colored by class). (a) Represents the canonical graph factorization baseline, (b) represents DeepWalk, (c) represents node2vec, and (d) represents our supervised Poincaré embeddings.

upon Euclidean methods on the email-Eu-core dataset due to its hierarchical nature, they perform worse than such baselines on the amorphous ChG-miner graph. These results indicate that supervised Poincaré embeddings excel when the structure of input graphs matches the hierarchical assumption under which the emebddings were trained, but that the embeddings lose signficant representational power when the hierarchical assumption is loosened. Further note that hybrid embeddings outperform both Euclidean and supervised Poincaré embedding frameworks, representing the power of utilizing representations in both spaces. The superior performance of hybrid embeddings over pure embeddings in both email-Eu-core and ChG-Miner indicates that incorporating information from embeddings in both Euclidean and hyperbolic space has significant utility regardless of graph hyperbolicity. This result demonstrates that hybrid embeddings can be utilized for a wide variety of prediction related tasks without regard for the individual structure of a graph as the fusion of both Euclidean and hyperbolic embeddings incorporates both structural and hierarchical information. Additionally, our empirical findings show promise for the bolstering of current state-of-the-art embedding frameworks with hybrid embeddings, independent of framework implementation.

We can further evaluate our embedding frameworks by comparing the quality of visualizations in Figure 1. Note first that the PCA plots for all four models are reasonably spread out with significant class mixing, aligning with the 60% observed accuracies on ChG-miner. However, the ball-like structure of the Poincaré embeddings in (d) conforms with our intuitions that Poincaré balls form spheres in Euclidean space. Furthermore, the stronger clustering of blue points in (d) compared to other methods indicates modest gains provided by supervised Poincaré embeddings over canonical Euclidean embeddings. The t-SNE plots on email-Eu-core further elucidate the profound improvements of supervised Poincaré embeddings over traditional models: while loose clusters are found in (a)-(c), the increase in cluster density obtained from t-SNE using Poincaré embeddings in (d) evinces the representational power of embedding graphs in hyperbolic space.

## 6 Conclusions and Future Work

Our work develops a supervised hybrid hyperbolic embedding framework to approach embedding tasks for arbitrarily complex graphs. We further generate models for node classification and link prediction provided node-level embeddings and evaluate our models on numerous real-world datasets. Our results indicate that our supervised hybrid hyperbolic embeddings vastly outperform traditional methods on both node classification and link prediction, indicating that leveraging graph hyperbolic structure alongside canonical Euclidean frameworks provides significant benefits for overall performance. Specifically, node classification results were bolstered by over 2 percent on both email-EU-core and ChG-miner datasets with the hybrid model, indicating that both of our developed methods effectively learned representations characterizing the structurally differing graphs. In the future, we hope to extend our work to more diverse and large datasets to further verify the benefits of hybrid embeddings on differing graphical structures. We further hope to identify more advanced methods of embedding fusion beyond concatenation.

## Contributions

Manan Shah developed baseline models, supervised Poincaré embeddings, the visualization pipeline, and conducted experiments. Sagar Maheshwari evaluated the developed models for both the node classification and link prediction tasks and conducted experiments.

## References


[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, 2011.

[3] Mikhael Gromov. Hyperbolic groups. In *Essays in group theory*, pages 75–263. Springer, 1987.

[4] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[5] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[6] Lucas Hu, Thomas Kipf, and Gökçen Eraslan. lucashu1/link-prediction: v0.1: FB and Twitter Networks, September 2018.

[7] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[8] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[9] Jure Leskovec and Andrej Krevl. {SNAP Datasets}:{Stanford} large network dataset collection. 2015.

[10] Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. *arXiv preprint arXiv:1806.03417*, 2018.

[11] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347, 2017.

[12] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[13] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[14] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.