Deep Classifiers For Fine Structure Need Segmentation

Arvind Rathnashyam
RPI Math and CS, rathna@rpi.edu

Radoslav Ivanov
RPI CS, ivanor@rpi.edu

Malik Magdon-Ismail
RPI CS, magdon@cs.rpi.edu

October 25, 2023

Abstract

We study fine-structure classification using trees as our application domain and deep networks as our primary classifier. Specifically, we investigate the role of segmentation in fine structure classification. Within a controlled synthetic setting, we demonstrate that deep networks don't learn the fine structure. Instead the network learns the background which results in low test accuracy for trees with no background or different backgrounds. We show that fine structure can be learned if the background is subtracted first. Background subtraction entails segmentation which is challenging when the foreground has essential fine structure. Hence, we pursue methods based on approximate coarse segmentation to extract the relevant topological properties of the fine structure, without the need for detailed fine structure segmentation. We propose a class of novel methods based on an intrinsic Markov Chain Model of the Fine Structure Graph. We use the Markov Chain stationary distribution to distinguish between different topologies. The Markov Chain approach gives superior performance over several neural network models in both synthetic and real data.

1 Introduction

Fine Structure Classification (FSC) can be considered as the problem of re-identification when the objects exhibit graph-like structure, e.g., trees (De Souza and Falcão, 2020), animals with specific markings (Nepovinnykh et al., 2020), or eye retinas (Shin et al., 2019). FSC is known to be challenging for purely data-driven techniques such as deep learning, as it requires large amounts of clean and segmented data for each class. In this paper, we present a new FSC dataset, namely a tree identification image dataset, along with exhaustive synthetic and real analysis aimed to demonstrate that neural networks (NNs) trained on FSC tasks tend to learn the background and generalize poorly unless segmentation is performed first.

Re-identification has been studied extensively in the problem of classifying Saimma ringed seals given images of seals and utilizing their permanent ringing patterns (Nepovinnykh et al., 2022). This dataset has been explored by various works (Chehrsimin et al., 2018; Nepovinnykh et al., 2018). These works, such as (Chelak et al., 2021), utilize a deep learning approach based on feature extraction from manually generated features. (Immonen et al., 2023) extends Chelak et al. (2021) and uses the geometry of the patterns extracted from the seals to increase the accuracy of the model. In ecology, the problem of re-identification allows researchers to track individual animals or plants over time, (Hu et al., 2022; Kirk et al., 2021; Ravoor and Sudarshan, 2020). This finds importance in studying migration patterns and population dynamics for specific species. Re-identification also aids in converservation efforts to monitor the movements of endangered species (Chelak et al., 2021; Nepovinnykh et al., 2023). Re-identification has also been studied in the context of specific ship identitification (Shi et al., 2018).

Current approaches for fine-structure classification utilize complex and large NNs to uncover features in the data and make decisions. In medical-imaging, the U-net (Ronneberger et al., 2015) is a state-of-the-art architecture for segmentation, i.e. pixel-by-pixel classification. Segmentation-based algorithms require a large amount of fine segmentation for classification. Convolutional Neural Networks (CNNs) Gu et al. (2018) have also emerged as a popular method for fine-structure classification in the past decade with the utilization of GPUs for fast compute.

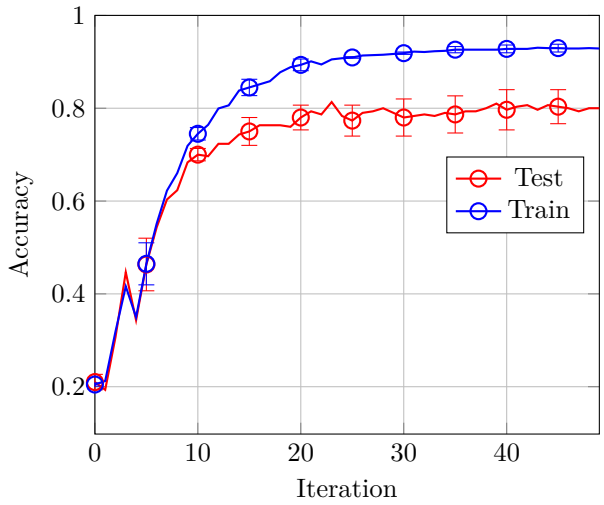


Figure 1: The dataset is 750 trees in 5 classes of our real dataset of trees pictured in various rotations, lighting conditions, and distances. Naïvely applying a CNN obtains good test accuracy.

We consider the problem of fine-structure classification for Trees. Tree identification is especially important when it comes to identifying hazard and danger trees, e.g., when protecting utility lines Townsend Corporation (2023). Tree identification is an especially difficult problem due to the high amount noise in the branches, viewpoint rotations, illumination differences, and occlusions (e.g. leaves, sunlight). To illustrate the challenges involved in this task, we collected a dataset of 55 trees, taking multiple photos daily of each tree over the course of 60 days, from spring until early-summer. In this paper, we will focus on a subset of 5 trees with a total of 750 tree images to make comparisons with our synthetic dataset in § 2.4. [1]

If we directly train a CNN on the raw image data, we observe an interesting pattern. From Figure 1, one might think it is a good idea to use CNNs for fine-structure classification for trees, as the NN achieves fairly high test accuracy as can be seen in Figure 1 as approximately 80%. However, one of the contributions of our paper is showing that this good accuracy is likely a result of learning the *background* features and therefore cannot generalize when the the background changes.

We instead propose to use the inherent graph structure of the trees and develop novel methods based on the Markov Chain for the classification of tree-graph structures. Assuming braches are not hidden, rotating a tree in the $3d$ space the tree exists in (i.e. moving around a tree and taking a picture), will give the same tree graph modulo the left to right placement of the edges. Furthermore, if we consider different lighting environments, if the branches are still visible the graph is still visible and invariant. We then compare our tree-graph approach against the state of the art methods in graph classification. We show in the small-data setting our class of models is up to 25.8 percent more accurate in our synthetic dataset, and 32.9 percent more accurate in our real dataset than state of the art methods in graph classification. Our algorithms use the topology of the tree (e.g. node neighborhoods, edge distance to root) and the geometry of the tree (edge distances) to learn a probability distribution over the tree space and make inferences. We summarize our contributions below.

Main Contributions

1. We show even though NNs achieve high accuracy in tree classification, with synthetic datasets, if the NNs are trained on fine-structure data with random objects in the background, the NNs are not robust to shifts in the background.

2. We propose a tree classification algorithm that utilizes the intrinsic Markov Chain Representation of the tree structure based on the ideas of *Expected Hitting Distance*.

3. We perform experiments to demonstrate the effectiveness our Markov Chain based algorithms compared to existing deep classification approaches.

[1] All data will be released upon acceptance of the paper.

Related work. Learning Tree Structures has been explored previously in the context of biomedical imaging (Feragen et al., 2013a, 2011b) and Natural Language Processing (NLP) (Zhang et al., 2010; Bai et al., 2021; Tai et al., 2015). A family of Geometric Tree Kernels is presented in Feragen et al. (2013b) for the classification of Geometric Trees. The distance between two trees root-paths by landmarking a set of points on each path and comparing the difference, either with a linear dot product or with the the gaussian kernel function. The spaces of Tree-Like shapes has been analyzed in (Feragen et al., 2011a). Using the space of tree-like shapes in Principal Component Analysis (PCA) has been studied in (Nye, 2011).
Tree classification is a subset of the graph classification problem. For graph classification, Graph Neural Networks (GNNs) have gained considerable attention due to their high expressivity and scalability in large datasets. The Graph Convolutional Network (Kipf and Welling, 2017) is a semi-supervised approach for Graph Classification based upon the principles of CNNs used typically in images. The convolutional structure is similar to CNNs for images where in lower layers local features are learned and in later layers more hollistic features about the graph are learned. Graph Isomorphism Networks,(Xu et al., 2019) are a powerful GNN which has provably maximal discrimative power among GNNs.

2 On the Limitations of Deep Learning for Fine Structure Classification

In this section, we discuss in detail the limitations of deep learning for FSC by performing extensive experiments on our real and synthetic datasets.

2.1 High-Level Approach

At a high-level, the goal of this paper is to illustrate the challenges of directly using deep learning for FSC. Our systematic approach consists of the following steps: 1) collect a real dataset of trees by varying the pose and background, as described below; 2) show that deep learning performs well on the raw classification task; 3) illustrate that the performance in 2 is deceptive since accuracy drastically drops when the classifier is applied to segmented (synthetic) images; 4) illustrate the benefits of a graph-based learning method given a coarse graph segmentation of each tree.

2.2 Dataset Description

In order to analyze the difficulty of FSC, we collect a high-definition dataset of tree images. In Figure 2 we give two sample images in our dataset. Our dataset was collected by taking pictures of trees everyday for one and a half months from April to May, when the trees go from no leaves to fully bloomed. We take the pictures from random positions around the tree at various times of day and weather conditions. Since we are not moving the trees, there do exist objects or buildings in the background of the same tree in many of the images. The two challenges of our dataset are given as follows: (i) learning a robust representation of the trees despite occlusions, rotations, and illumination changes, and (ii) not overfitting on the background objects.

2.3 Limitations of Training with Non-Segmented Data

From Figure 1 it is clear the train and test accuracy is high, however this does not show the full story. We first refer to Figure 2. In the background of images within the same class, there tend to be many similar objects in the background. Without segmentation, the NN can learn the aforementioned background objects. The rotation in the image plane in Figure 2 makes the tree structure look different, however the lamp post common in both images looks the same and is therefore easier to classify. These limitations lead to decreased generalization performance, e.g. when animals migrate, or the climate changes, the representation learned by the NN will not be robust to such changes.

2.4 Synthetic Data

(a) (b)

Figure 2: An example of a common background object in the real dataset.

(a) (b)

Figure 3: Example of Synthetic Data without background generated from the $3d$ rendering software (Community, 2018) with the Mtree Addon (Herpin, 2016).

(a) SAM and Real Image (b) SAM and Real Image

Figure 4: Segmentation Results from Segment-Anything-Model (Kirillov et al., 2023). SAM is good at distinguishing between structures such as buildings, however is unable to segment the fine structure of the tree such as the branches.

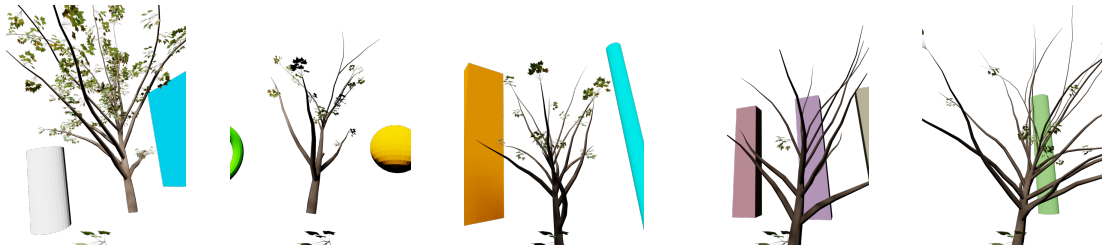
Through the use of a carefully curated synthetic dataset, we will show when given unsegmented data, NNs will learn the features in the background and do not generalize to data with different background. In Figure 3, we give two sample images of the synthetic dataset we have created in the 3d rendering software, Blender Community (2018) with Mtree Addon (Herpin, 2016). In Blender we are able to replicate the real-world dataset, by adding leaves, changing the rotation, lighting, and distance from the camera to the tree.

To replicate the real-world data, we place random objects unique to each class of trees in the background. We show examples of the images in Figure 15.

We create two datasets, $\mathcal{D}_{\text{BG}}$ and $\mathcal{D}_{\cancel{\text{BG}}}$ which are exact copies of each other, i.e same lighting, rotation, occlusions, with the exception that the trees in $\mathcal{D}_{\text{BG}}$ have random objects in the background unique to each class as in Figure 15 whereas the trees in $\mathcal{D}_{\cancel{\text{BG}}}$ have had the background objects subtracted. We are thus able to create, train and test splits which are exact copies of each other modulo the random objects. We then train our neural network model on all the training/test combinations of datasets. We refer to the subsets of $\mathcal{D}_{\text{BG}}$ as $\mathcal{D}_{\text{BG}}^{750}$ and $\mathcal{D}_{\text{BG}}^{5000}$ with 750 and 5000 total samples, respectively. We

Train	Test	Test Accuracy
$\mathcal{D}_{\cancel{\text{BG}}}^{750}$	$\mathcal{D}_{\cancel{\text{BG}}}^{750}$	$71.3_{(1.89)}$
$\mathcal{D}_{\cancel{\text{BG}}}^{750}$	$\mathcal{D}_{\text{BG}}^{750}$	$39.7_{(2.36)}$
$\mathcal{D}_{\text{BG}}^{750}$	$\mathcal{D}_{\text{BG}}^{750}$	$32.7_{(5.66)}$
$\mathcal{D}_{\text{BG}}^{750}$	$\mathcal{D}_{\text{BG}}^{750}$	$82.3_{(8.96)}$
$\mathcal{D}_{\cancel{\text{BG}}}^{5000}$	$\mathcal{D}_{\cancel{\text{BG}}}^{5000}$	$83.4_{(4.80)}$
$\mathcal{D}_{\cancel{\text{BG}}}^{5000}$	$\mathcal{D}_{\text{BG}}^{5000}$	$56.1_{(4.38)}$
$\mathcal{D}_{\text{BG}}^{5000}$	$\mathcal{D}_{\cancel{\text{BG}}}^{5000}$	$33.9_{(0.57)}$
$\mathcal{D}_{\text{BG}}^{5000}$	$\mathcal{D}_{\text{BG}}^{5000}$	$98.5_{(1.01)}$

Table 1: Testing Accuracies for our Synthetic Datasets. We train on 80% of the training dataset and test on 20% of the testing dataset.



(a) Class 1 (b) Class 2 (c) Class 3 (d) Class 4 (e) Class 5

Figure 5: Synthetic Tree Images with Random Objects in the Background that are unique to each class of trees. Each class represents the same tree with images taken from different angles and lighting conditions.

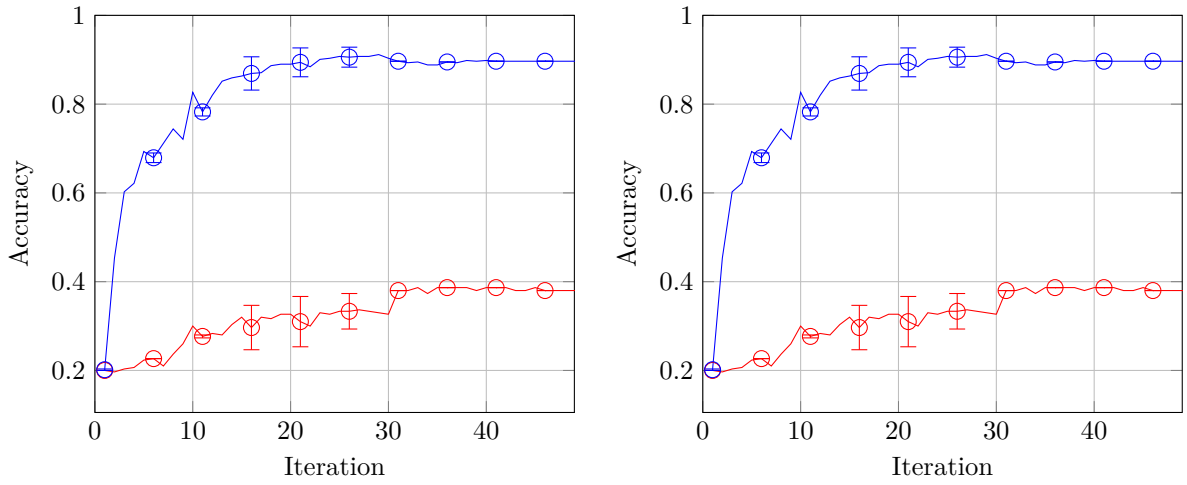


Figure 6: Train Accuracy is given in blue and Test Accuracy is given in red. We train on data with random objects in the background and test on images with no background. In *(Left)*, we train on $\mathcal{D}_{\text{BG}}^{750}$ and test on $\mathcal{D}_{\not{\text{BG}}}^{750}$. In *(Right)*, we train on $\mathcal{D}_{\text{BG}}^{5000}$ and test on $\mathcal{D}_{\not{\text{BG}}}^{5000}$. This Figure shows the increase in data does not overcome the distributional shift of removing background objects.

similarly have $\mathcal{D}_{\not{\text{BG}}}^{750}$ and $\mathcal{D}_{\not{\text{BG}}}^{5000}$. The $\mathcal{D}_{\not{\text{BG}}}$ datasets have what we consider to be Fine Structure Segmentation, i.e. background is subtracted. Our results are in Table 1. Therefore, with our carefully curated synthetic dataset, it is clear learning of the intrinsic features is required as the neural network will learn features that are easy to distinguish, but not necessarily of the fine structure. Therefore, when presented in a new environment, e.g. when Saimma ringed seals migrate, the neural network will not classify based on the features of the fine-structure itself.

From our experiments in Table 1, we make four observations. (I) From rows two and six in Table 2, also graphed in Figure 6, a neural trained with the background objects does not learn a good representation of the fine tree structure and thus can not generalize to trees without background objects. (II) From rows three and seven in Table 2, a neural network that is trained on the the graph structure alone is not robust to the addition of random, class-specific objects in the background. (III) From observing the difference in rows two and six and rows three and seven, we observe that increasing the dataset size does not significantly help in overcoming the generalization gap. (IV) When given more segmented data, the accuracy of the NN increases. It is clear from our four observations, to develop an algorithm that is robust to changes in the background and the scene in general, we must either develop an algorithm to learn the intrinsic features of the fine-structure or develop a method for pixel-level fine structure segmentation.

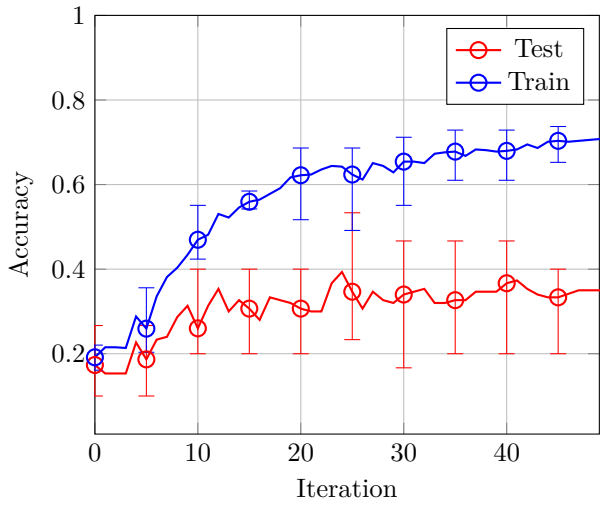


Figure 7: The dataset is 150 trees in 5 classes with Tree Segmentations from SAM.

2.5 An attempt at Fine-Structure Segmentation

In § 2.4 we show a NN can learn a good representation of the fine-structure when given a pixel-level segmentation of the object. With the advent of foundation models for segmentation such as the Segment-Anything-Model (SAM) (Kirillov et al., 2023), object segmentation has now become available across various different objects, including Trees. Sample images of SAM Segmentation can be seen in Figure 4. Although there is some level of segmentation from SAM, there is no good generalization from the CNN as can be seen in Figure 7.

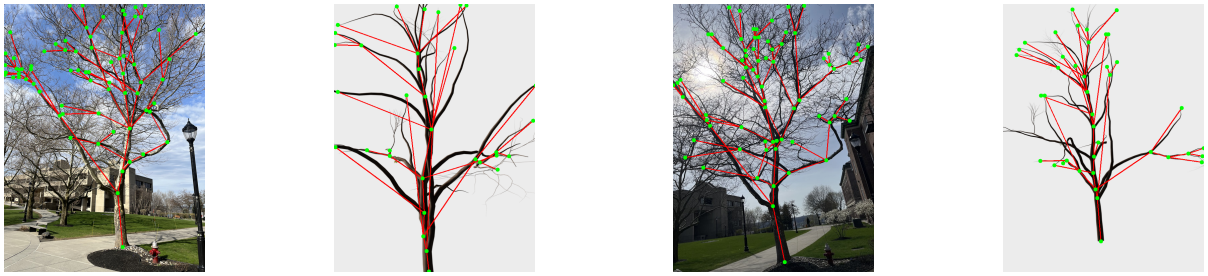
3 Fine Structure Markov Chains

We will use the inherent graph structure in the tree for our algorithms, as even for simple objects, pixel-level segmentation is time-consuming Lee et al. (2022) and requires extensive human labor. In Figure 8, we overlay the graph structure on the trees, which take approximately $3-5$ minutes to generate with our Geometric Tree Extraction Interface [2]. The literature on learning algorithms for geometrical trees is quite sparse, as it is a small subfield within graph classification. Existing methods in graph classification are tailored to utilizing the node-level information and propagating it towards its neighbors. Although information such as node degree and edge distance can be stored as node-level information, these approaches tend to not work well in our dataset where the information is topological and geometric, as we demonstrate in our experimental section. Furthermore, extracting the fine-structure graph can be quite costly in settings such as biomedical imaging (Feragen et al., 2011b) where the structure of the fine-structure graph is intricate. Therefore, we require an algorithm that is not only accurate, but also data-efficient.
In this section we will discuss the background for Markov Chain Modeling and then present our algorithms.

3.1 Notation

We are a given a dataset $\mathcal{D} = \{(T_i, c_i)\}_{i=1}^n$ where $T_i = (\mathcal{V}_i, \mathcal{E}_i)$ and $c_i \triangleq \mathcal{C}(T_i)$ is a class in the set $\mathcal{C}$. We then learn a function $f : \mathcal{T} \to \mathcal{C}$ for tree classification. We let $\pi(v_1, v_2)$ represent a path between two vertices, $v_1, v_2 \in \mathcal{V}$. Let $\rho : \mathcal{V} \to \mathbb{Z}_+$ be defined as $\rho(v) = \min_{\pi(v,v_0)} |\pi(v, v_0)|$, be the distance to the root of the graph. We define $D \triangleq \max_{\mathcal{T}_i \in \mathcal{D}} \max_{v \in \mathcal{V}_i} \Gamma(v)$, i.e. the largest vertex degree in the dataset and $L \triangleq \max_{\mathcal{T}_i \in \mathcal{D}} \max_{v \in \mathcal{V}_i} \rho(v)$, i.e. the longest distance from the root in terms of edges in $\mathcal{D}$. Then we define states in a Markov chain as $s = (\alpha, \beta)$ where $\alpha \triangleq \rho(\cdot)$ and $\beta \triangleq \Gamma(\cdot)$ and the mapping $\phi : \mathcal{V} \to \mathcal{S}$ as $\phi(v) = (\rho(v), \Gamma(v))$. It is a two-tuple which represents a state by its distance from the root and the number

[2] Code for the interface will be released upon acceptance of the paper



(a) Real (b) Synthetic (c) Real (d) Synthetic

Figure 8: Geometric Tree Graphs overlaying the Tree Images. The graphs are generated with our geometric graph generation interface described in the Appendix.

of neighbors. Next, we define $[\mathbf{M}]_{ij} \triangleq \mathrm{M}_{ij} = \mathbb{P}[s_i \to s_j]$ for any $s_i, s_j \in \mathcal{S}$. Furthermore, we represent $\mathcal{B}(T)$ as all branches of T, where all paths of the form $\phi(\pi(r \to v)) \in \mathcal{B}(T)$, where r is the root of T, v is any node such that $\Gamma(v) = 0$, and ϕ is applied to each vertex in the path. The branch is then the ordered list of Markov Chain States for this path from the root to a leaf.

Algorithm 1: Feature Vector to Markov Vector

Input: $\boldsymbol{x} \triangleq \{(v_i, \mu_i)\}_{v_i \in \mathcal{V}(T)} \in \mathbb{R}^{|\mathcal{V}(T)| \times 2}$

Output: $\Phi(\boldsymbol{x}) \in \mathbb{R}^{|S|}$

1: **for** $i \in [\mathcal{V}(T)]$ **do**

2: $\quad \Phi_{\phi(v_i)}(\boldsymbol{x}) \leftarrow \Phi_{\phi(v_i)}(\boldsymbol{x}) + \mu_i$

3: **end**

3.2 Markov Chain Preliminaries

A Markov Chain can be represented by a 3-tuple $(\mathcal{S}, \mathcal{P}, s_0)$, Norris (1997). $\mathcal{S}$ represents the state space of the Markov Chain, $\{s_0, s_1, \ldots, s_n\}$. $\mathcal{P}$ represents the probabilistic transition matrix. Each entry $\mathcal{P}_{ij}$ represents the probability of moving to s_j from s_i. The initial state distribution is given as s_0, which we often refer to as a start state. The Markov Property states for all $s_n \in \mathcal{S}$, it follows $\mathbb{P}\{s_n | s_{n-1}, s_{n-2}, \ldots, s_0\} = \mathbb{P}\{s_n | s_{n-1}\}$.

Definition 1. Let a Markov Chain be defined by $(\mathcal{S}, \mathcal{P}, s_0)$, then a Markov Chain is **irreducible** if for all $s_1 \in \mathcal{S}$, there exists a path to any $s_2 \in \mathcal{S}$.

Definition 2. Let a Markov Chain be defined by $(\mathcal{S}, \mathcal{P}, s_0)$. Let $\mathbb{P}[s_i \to s_i | t]$ be the return probability after t steps. If there exists t s.t. $\mathbb{P}[s_i \to s_i | nt] = 0$ for all $n \in \mathbb{N}$, then the Markov Chain is Periodic. If no such t exists, then the Markov Chain is **aperiodic**.

3.3 Topological Features

In this section we give our Markov Chain Approaches which utilize the Topological Information of the Tree Graph.

3.3.1 Markov Chain Likelihood Approach

In our first proposed method, we use the Maximum Likelihood Estimation (MLE) over the Markov Chain Representation of the Tree.

Theorem 3. *Let the Markov Chain be defined as* $(\mathcal{S}, \mathcal{P}, s_0)$ *and the classification problem be defined as follows:*

$$c = \arg\max_{c \in \mathcal{C}} \mathbb{P}\{C = c | T\} \tag{1}$$

under the condition

$$\sum_{k=1}^{|\mathcal{C}|} \mathbb{P}\left\{\mathcal{C}(T) = c|T\right\} = 1 \;\; \forall T \in \mathcal{T} \tag{2}$$

Let $\mathcal{B}(T)$ represent the set of branches in tree T, then the negative log-likelihood of a tree T belonging to class k is given as

$$\begin{aligned} &\arg\max_{c\in\mathcal{C}} \mathbb{P}\left[\mathcal{C}(T) = c|T\right] = \\ &\arg\min_{c\in\mathcal{C}} \left\{ \sum_{b\in\mathcal{B}(T)} \sum_{s_i\in b} \log\left(\mathcal{P}^{(k)}_{s_{i-1},s_i}\right) + \log\left(\mathbb{P}\left\{\mathcal{C}(T) = c\right\}\right) \right\} \end{aligned} \tag{3}$$

This represents the conditional probability a geometric tree T, is in class c of the $|\mathcal{C}|$ total classes. Similar to Feragen et al. (2013b), we identify a tree by it's root-leaf paths.

3.3.2 Calculating $\mathcal{P}$

Let $\mathcal{T}^{(k)}$ represent the set of trees in the training data set from class k, and let $\mathcal{N}^{(k)}(\cdot)$ represent the number of occurences of reaching $\cdot$ within $\mathcal{T}^{(k)}$.

$$\mathcal{P}^{(k)}_{s_{i-1},s_i} = \frac{\mathcal{N}^{(k)}\left(s_{i-1} \to s_i\right)}{\mathcal{N}^{(k)}\left(s_{i-1} \to \cdot\right)} \tag{4}$$

3.4 Stationary Distribution Approach

In our second approach, we make use of the stationary distribution $\boldsymbol{\pi}$, a vector that satisfies the following conditions

$$\sum_{i=1}^{|\mathcal{S}|} \pi_i = 1 \tag{5}$$

$$\boldsymbol{\pi}\mathcal{P} = \boldsymbol{\pi} \tag{6}$$

Let us note the probabilistic transition matrix of a geometric tree is always irreducible, however, this is a periodic Markov Chain. For example, if we start at the root, we can only return to the root in steps $2, 4, 6, \ldots$ due to all self-transition probabilities being equal to 0. The problem of aperiodicity is as follows, let the eigenvalues of $\mathbf{P}$ be $|\lambda_1| \geq |\lambda_2| \geq \ldots |\lambda_n|$. Since $\mathcal{P}$ has periodicity of 2, we end up with $|\lambda_1| \approx |\lambda_2|$. It thus follows in the power method

$$\boldsymbol{x}\mathcal{P}^k = \alpha_1\lambda_1^k\boldsymbol{v}_1 + \alpha_2\lambda_2^k\boldsymbol{v}_2 \tag{7}$$

Therefore the resultant stationary distribution will not be unique. To make the stationary distribution unique, we create an aperiodic Probabilistic Matrix with the following perturbation for some $\varepsilon \in (0, 1)$

$$\tilde{\mathbf{P}} = (1-\varepsilon)\mathbf{P} + \varepsilon\mathbf{R}, \quad \mathbf{R} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \ldots & 0 \end{pmatrix} \tag{8}$$

We refer to ε as the *return probability.* At each state in the graph, we have a probability of ε of returning to the root. Therefore, a higher ε will mean the vertices far from the root will be less explored, which can be beneficial as the vertices far from the root tend to carry the most amount of noise.

Lemma 4. *The probabilistic matrix $\tilde{\mathbf{P}} \triangleq (1-\varepsilon)\mathbf{P} + \varepsilon\mathbf{R}$ where $\mathbf{P}$ is a probabilistic matrix with columns summing up to 1 s.t. $\mathrm{P}_{i,j} \geq 0$ for all $i \in [n]$ and $j \in [n]$ representing the Markov Chain represnetation of the fine-structure graph, then $\tilde{\mathbf{P}}$ is a aperiodic matrix.*

We can now find the stationary distribution of $\tilde{\mathcal{P}}$ by calculating the eigenvector that is associated with $\lambda = 1$. Then consider the probabilities of the stationary distribution as μ_i, we then use Algorithm 3 to obtain $\Phi(\boldsymbol{x})$. Our Markov Chain representation has the nice propertie of being robust to different sizes of the original graphs as as the stationary distribution for any tree is mapped to the same size vector of dimension $|\mathcal{S}|$, which is then used as the feature representation of T in a SVM with the RBF kernel. Our results can be seen in Table 2. In Figure 9, we see at low return probabilities, the accuracy is maximized.

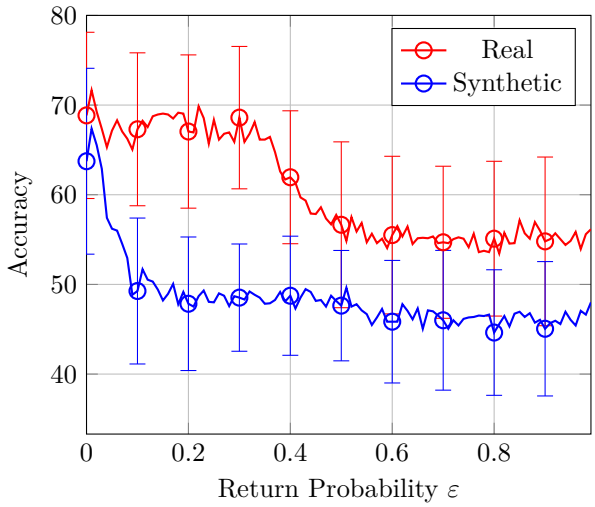


Figure 9: The effect of the return probability on the accuracy for our two datasets.

Table 2: Accuracy of our Markov based tree learning methods and various state of the art algorithms for our Synthetic Dataset and Real Dataset

Method	Synthetic Dataset (↑)	Real Dataset (↑)
Markov Chain Maximum Likelihood *(ours)*	$\mathbf{70.33}_{(\mathbf{9.304})}$	$58.77_{(1.026)}$
Stationary ('rbf', $p = 0.01$) *(ours)*	$\underline{69.65}_{(9.145)}$	$\mathbf{68.84}_{(\mathbf{9.667})}$
Geometric Expected Hitting Distance ('rbf') *(ours)*	$56.00_{(7.461)}$	$\underline{62.81}_{(8.26)}$
Roothpath, Gaussian (Feragen et al., 2013b)	$15.5_{(0.212)}$	$13.68_{(5.367)}$
Graph Convolutional Network (Kipf and Welling, 2017)	$44.5_{(11.056)}$	$34.39_{(10.677)}$
Graph Isomorphism Network (Xu et al., 2019)	$24.67_{(8.055)}$	$26.49_{(9.064)}$
Vertex Histogram Kernel	$42.67_{(10.71)}$	$35.96_{(10.141)}$
Spectral Baseline Method (de Lara and Pineau, 2018)	$20.07_{(3.392)}$	$21.05_{(4.611)}$

3.4.1 Perturbation Analysis

Obtaining the graph structure of an image will inevitably incur noise between the annotator, the viewpoint changes, and general visibility of the tree. We thus dedicate some analysis to the perturbation of the tree on the Markov stationary distribution. By the formation of the Tree Edit Distance (Bille, 2005; Zhang and Shasha, 1989), we can set the distance between trees T_1 and T_2 as a series of deletions and additions of leaf vertices.

Theorem 5. *Consider two trees, T_1 and T_2, where the difference is a sequence of R leaf additions and leaf deletions. Let $\mathbf{M}$ be the Markov matrix associated with T_1 and $\tilde{\mathbf{M}}$ be the Markov Matrix associated with T_2 and $\boldsymbol{\pi}$ be the dominant eigenvector for $\mathbf{M}$ and $\tilde{\boldsymbol{\pi}}$ be the dominant eigenvector for $\tilde{\mathbf{M}}$, we then have*

$$\|\boldsymbol{\pi} - \tilde{\boldsymbol{\pi}}\| \leq \frac{8R\sqrt{2}}{1 - \sigma_2(\mathbf{M})} \tag{9}$$

From Theorem 5, we see small perturbations to the structure of the tree cause small changes to the stationary distribution in theory. Between tree classes, the lower-level structure is likely significantly different giving a larger upper bound on the distance between the stationary distributions.

3.5 Geometric Features

Inspired by using the geometry of the features as in (Immonen et al., 2023), in our third proposed method, we utilize the locations of the nodes in a metric space $(\mathcal{X}, d)$ in conjunction with the Markov Chains for classification.

Definition 6. Let $A \subset S$. The **Hitting Distance** of a Geometric Markov Chain is defined as:

$$D_A = \inf\{n \in \mathbb{R}_{++} : d(\pi(s_{\text{start}}, s)), s \in A\} \tag{10}$$

where d represents a metric in the $\mathbb{R}^2$ space and calculates the distances of all edges in the path from s_{start} and s.

Definition 7. Let $A \subset \mathcal{S}$. Then the **Expected Hitting Distance** of a Geometric Markov Chain defined by the tuple $(\mathcal{S}, \mathcal{P}, s_0)$ is defined as follows:

$$\mathbb{E}\left[D_{iA}\right] = \mathbb{E}\left[D_A | s_{\text{start}} = s_i\right] \tag{11}$$

The Expected Hitting Distance defined in equation 11 can be solved with the following set of equations:

$$\mathbb{E}\left[D_{iA}\right] = \begin{cases} \sum_{j \in \mathcal{S}} \mathcal{P}_{ij}\left(d(i,j) + \mathbb{E}\left[D_{jA}\right]\right) & \text{if } i \notin A \\ 0 & \text{if } i \in A \end{cases} \tag{12}$$

This system of equations grows exponentially with the number of nodes. Thus we propose a novel approach to find the expected hitting distance of each node from the root by forming a Markov Chain based on the edges of the tree.

Theorem 8. *Consider the Adjacency matrix between the edges of the tree and the start state, $\mathcal{E}(T) \cup s_{\text{start}}$. Let Q represent the transient states of the adjacency matrix associated with the edges and s_{start}. The expected hitting distance from s_{start} to all other transient vertices can be solved by*

$$\boldsymbol{\xi} = (\mathbf{I} - \mathbf{Q})^{-1}\boldsymbol{\ell} \tag{13}$$

where $\boldsymbol{\ell} = \begin{bmatrix} 0 & \ell(e_1) & \ell(e_2) & \cdots & \ell(e_n) \end{bmatrix}^\top$ and $\ell(e)$ represents the length of edge e and the entries of $\boldsymbol{\xi}$ represent the expected distance from s_{start} to all the nodes in Q.

With Theorem 8, for each tree we calculate $\boldsymbol{\xi}$, then similar to our approach in § 3.4. We use Algorithm 3 to obtain $\Phi(\boldsymbol{\xi})$ and use it as data for a SVM with RBF Kernel.

4 Experiments

In this section, we will compare our Markov Chain methods to state of the art methods in graph and tree classification. In Table 3, we list the different statistics of each dataset. We see the `Real` dataset has trees with more nodes and edges on average, and therefore is likely to have significantly more noise. We split both

Table 3: Statistics of the Synthetic and Real Dataset

Dataset	Graphs	Classes	Avg. Nodes	Avg. Edges
`Synthetic`	100	5	38.29	39.29
`Real`	100	5	55.84	56.84

datasets with a 80/20 train-test split. We train each method on the train data and report the classification accuracy on the test set. We repeat all experiments 30 times and report the mean and standard deviation[3]. In Table 2, we find our class of Markov Chain based tree learning algorithms significantly outperform the state-of-the-art tree and graph algorithms on both our synthetic and real tree datasets.
To gain insight on the effectiveness of the stationary distribution approach in § 3.4, we use t-distributed Stochastic Neighbor Embedding (t-SNE), (Van Der Maaten, 2014), to visualize the stationary distributions in the 2-dimensional space in Figure 10.

[3]Code to reproduce all results will be released upon acceptance of the paper

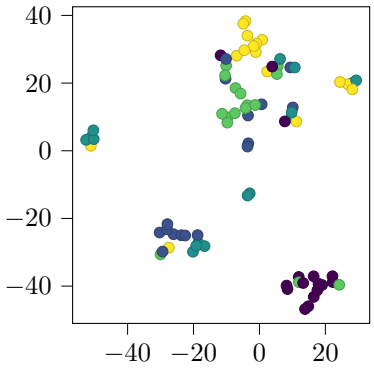


Figure 10: t-SNE on the stationary distributions of the Markov Chain representation of the trees described in 3.4. Even in 2-dimensions, we can see clearly there exist a good separation of the data across at least four of the five classes.

5 Discussion

In this paper, we study the limitations of Deep Classifiers for Fine Structure. Although Deep Classifiers are able to get good accuracy on our real dataset, with our synthetic dataset we show these NNs will not generalize when the background changes. We create our synthetic dataset in a 3d rendering software and show NNs tend to learn *background* features. We do find, however, that NNs are able to classify Fine Structure when given pixel-level segmentation. Pixel-level fine structure segmentation is difficult to obtain. We therefore use the intrinsic graph structure of the Tree. We use the Markov Chain Model for Geometric Tree Graphs to utilize the topological and geometric features of the trees. We experimentally demonstrate the effectiveness of our algorithms compared to state-of-the-art algorithms in tree and graph classification.

References

Abu-Mostafa, Y. S., Magdon-Ismail, M., and Lin, H.-T. (2012). *Learning from data*, volume 4. AMLBook New York.

Bai, J., Wang, Y., Chen, Y., Yang, Y., Bai, J., Yu, J., and Tong, Y. (2021). Syntax-bert: Improving pre-trained transformers with syntax trees. *arXiv preprint arXiv:2103.04350*.

BAYES (1958). An essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3-4):296–315.

Bille, P. (2005). A survey on tree edit distance and related problems. *Theoretical computer science*, 337(1-3):217–239.

Chehrsimin, T., Eerola, T., Koivuniemi, M., Auttila, M., Levänen, R., Niemi, M., Kunnasranta, M., and Kälviäinen, H. (2018). Automatic individual identification of saimaa ringed seals. *IET Computer Vision*, 12(2):146–152.

Chelak, I., Nepovinnykh, E., Eerola, T., Kälviäinen, H., and Belykh, I. (2021). Eden: deep feature distribution pooling for saimaa ringed seals pattern matching. In *International Conference Cyber-Physical Systems and Control*, pages 141–150. Springer.

Community, B. O. (2018). *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam.

de Lara, N. and Pineau, E. (2018). A simple baseline algorithm for graph classification.

De Souza, I. E. and Falcão, A. X. (2020). Learning cnn filters from user-drawn image markers for coconut-tree image classification. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5.

Feragen, A., Hauberg, S., Nielsen, M., and Lauze, F. (2011a). Means in spaces of tree-like shapes. In *2011 International Conference on Computer Vision*, pages 736–746. IEEE.

Feragen, A., Owen, M., Petersen, J., Wille, M. M., Thomsen, L. H., Dirksen, A., and de Bruijne, M. (2013a). Tree-space statistics and approximations for large-scale analysis of anatomical trees. In *Information Processing in Medical Imaging: 23rd International Conference, IPMI 2013, Asilomar, CA, USA, June 28–July 3, 2013. Proceedings 23*, pages 74–85. Springer.

Feragen, A., Pechin, L., Gorbunova, V., Nielsen, M., Dirksen, A., Reinhardt, J. M., Lauze, F., and De Bruijne, M. (2011b). An airway tree-shape model for geodesic airway branch labeling. In *Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy-Geometrical and Statistical Methods for Modelling Biological Shape Variability*, pages 123–134.

Feragen, A., Petersen, J., Grimm, D., Dirksen, A., Pedersen, J. H., Borgwardt, K., and de Bruijne, M. (2013b). Geometric tree kernels: Classification of copd from airway tree geometry. In Gee, J. C., Joshi, S., Pohl, K. M., Wells, W. M., and Zöllei, L., editors, *Information Processing in Medical Imaging*, pages 171–183, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., et al. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377.

Haigh, J. (2013). *Probability Models*. Springer London.

Herpin, M. (2016). Modular tree addon.

Hu, N., Wang, S., Wang, X., Cai, Y., Su, D., Nyamsuren, P., Qiao, Y., Jiang, Y., Hai, B., and Wei, H. (2022). Lettucemot: A dataset of lettuce detection and tracking with re-identification of re-occurred plants for agricultural robots. *Frontiers in Plant Science*, 13:1047356.

Immonen, V., Nepovinnykh, E., Eerola, T., Stewart, C. V., and Kälviäinen, H. (2023). Combining feature aggregation and geometric similarity for re-identification of patterned animals. *arXiv preprint arXiv:2308.06335*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al. (2023). Segment anything. *arXiv preprint arXiv:2304.02643*.

Kirk, R., Mangan, M., and Cielniak, G. (2021). Robust counting of soft fruit through occlusions with re-identification. In *International Conference on Computer Vision Systems*, pages 211–222. Springer.

Lee, J., Ilyas, T., Jin, H., Lee, J., Won, O., Kim, H., and Lee, S. J. (2022). A pixel-level coarse-to-fine image segmentation labelling algorithm. *Scientific Reports*, 12(1):8672.

Nepovinnykh, E., Eerola, T., Biard, V., Mutka, P., Niemi, M., Kunnasranta, M., and Kälviäinen, H. (2022). Sealid: Saimaa ringed seal re-identification dataset. *Sensors*, 22(19):7602.

Nepovinnykh, E., Eerola, T., and Kalviainen, H. (2020). Siamese network based pelage pattern matching for ringed seal re-identification. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision workshops*, pages 25–34.

Nepovinnykh, E., Eerola, T., Kälviäinen, H., and Radchenko, G. (2018). Identification of saimaa ringed seal individuals using transfer learning. In *Advanced Concepts for Intelligent Vision Systems: 19th International Conference, ACIVS 2018, Poitiers, France, September 24–27, 2018, Proceedings 19*, pages 211–222. Springer.

Nepovinnykh, E., Vilkman, A., Eerola, T., and Kälviäinen, H. (2023). Re-identification of saimaa ringed seals from image sequences. In *Scandinavian Conference on Image Analysis*, pages 111–125. Springer.

Norris, J. R. (1997). *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Nye, T. M. (2011). Principal components analysis in the space of phylogenetic trees. *The Annals of Statistics*, pages 2716–2739.

O'Rourke, S., Vu, V., and Wang, K. (2018). Random perturbation of low rank matrices: Improving classical bounds. *Linear Algebra and its Applications*, 540:26–59.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Ravoor, P. C. and Sudarshan, T. (2020). Deep learning methods for multi-species animal re-identification and tracking–a survey. *Computer Science Review*, 38:100289.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.

Shi, Q., Li, W., Zhang, F., Hu, W., Sun, X., and Gao, L. (2018). Deep cnn with multi-scale rotation invariance features for ship classification. *Ieee Access*, 6:38656–38668.

Shin, S. Y., Lee, S., Yun, I. D., and Lee, K. M. (2019). Deep vessel segmentation by learning graphical connectivity. *Medical image analysis*, 58:101556.

Tai, K. S., Socher, R., and Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Townsend Corporation (2023). Identifying Hazard & Danger Trees Around Utility Lines. https://www.townsendcorporation.com/services/vegetation-management/danger-trees-around-power-lines.

Van Der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *The journal of machine learning research*, 15(1):3221–3245.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations*.

Zhang, K. and Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

Zhang, W., Li, P., and Zhu, Q. (2010). Sentiment classification based on syntax tree pruning and tree kernel. In *2010 Seventh Web Information Systems and Applications Conference*, pages 101–105. IEEE.

A Theory

We will first give the necessary notation, and then proceed to the technical results.

A.1 Notation

Define $\mathcal{S}$ as the state space, $\mathcal{P}$ be all the one step transition probabilities. Let $\mathbb{P}[s_i \to s_j] = \mathcal{P}_{s_i,s_j}$, where $s_i, s_j \in \mathcal{S}$, be the transition probability between states s_i and s_j. Let $\pi(s_i, s_j)$ be a path from $s_i \to s_j$. The number of the states in a path between $s_i, s_j \in \mathcal{S}$ is given as $|\pi(s_i, s_j)|$. Using the above definitions, the notation $\mathbb{P}[s_0 \to s_j||\pi(s_0, s_j)| = k]$ represents the probability of a random walker going from state $s_0 \in \mathcal{S}$ to another state $s_j \in \mathcal{S}$ in a path with k vertices. Branches are can be considered as leaf-root paths and are denoted by the set $\mathcal{B}$. They are the set of states from the root to a vertex with degree 1.

A.2 Proof of Theorem 3

Proof. We will determine the Tree class in $\mathcal{C}$ which maximizes the likelihood of Tree T being in class c. Let C represent the class of T.

$$\arg\max_{c\in\mathcal{C}} \mathbb{P}\{C = c|T\} \overset{\zeta_1}{=} \arg\max_{c\in\mathcal{C}} \left\{ \frac{\mathbb{P}\{T|C = c\}\,\mathbb{P}\{C = c\}}{\mathbb{P}\{T\}} \right\} \tag{14}$$

$$\overset{\zeta_2}{=} \arg\max_{c\in\mathcal{C}} \left\{ \frac{\mathbb{P}\{T|C = c\}\,\mathbb{P}\{C = c\}}{\mathbb{P}\{T|C = c\}\,\mathbb{P}\{C = c\} + \mathbb{P}\{T|C \neq c\}\,\mathbb{P}\{C \neq c\}} \right\} \tag{15}$$

$$= \arg\max_{c\in\mathcal{C}} \left\{ \frac{\mathbb{P}\{T|C = c\}\,\mathbb{P}\{C = c\}}{\mathbb{P}\{T|C = c\}\,\mathbb{P}\{C = c\} + \sum_{j=1}^{m} \mathbb{I}\{j \neq c\}\,\mathbb{P}\{T|C = j\}\,\mathbb{P}\{C = j\}} \right\} \tag{16}$$

$$= \arg\max_{c\in\mathcal{C}} \left\{ \frac{\mathbb{P}\{T|C = c\}\,\mathbb{P}\{C = c\}}{\sum_{j=1}^{m} \mathbb{P}\{T|C = j\}\,\mathbb{P}\{C = j\}} \right\} \tag{17}$$

(ζ_1) follows from an application of Bayes' Theorem, (BAYES, 1958). (ζ_2) follows from the law of total probability, (Haigh, 2013). Let $\mathcal{B}(T)$ represent set of branches in tree T

$$= \arg\max_{c\in\mathcal{C}} \left\{ \frac{\left(\prod_{b\in\mathcal{B}(T)} \mathbb{P}\{b|C = c\}\right)\mathbb{P}\{C = c\}}{\sum_{j=1}^{|\mathcal{C}|}\left(\left(\prod_{b\in\mathcal{B}(T)} \mathbb{P}\{b|C = j\}\right)\mathbb{P}\{C = j\}\right)} \right\} \tag{18}$$

Let us here note that a branch is a sequence of states from the root to the branch of the markov chain

$$= \arg\max_{c\in\mathcal{C}} \left\{ \frac{\left(\prod_{b\in\mathcal{B}(T)} \prod_{s_i\in b} \mathcal{P}^{(c)}_{s_{i-1},s_i}\right)\mathbb{P}\{C = c\}}{\sum_{j=1}^{|\mathcal{C}|}\left(\left(\prod_{b\in\mathcal{B}(T)} \prod_{s_i\in b} \mathcal{P}^{(c)}_{s_{i-1},s_i}\right)\mathbb{P}\{C = j\}\right)} \right\} \tag{19}$$

$\mathcal{P}^{(c)}_{s_{i-1},s_i}$ represents the transition probability from state s_{i-1} to s_i in the Markov Chain for the trees in class (c). Then from the Principle of Maximum Likelihood Abu-Mostafa et al. (2012), we note the function $-\ln(1/\cdot)$ is a strictly increasing function.

$$= \arg\min_{c\in\mathcal{C}} \left\{ -\log\left(\frac{\sum_{j=1}^{|\mathcal{C}|}\left(\left(\prod_{b\in\mathcal{B}(T)} \prod_{s_i\in b} \mathcal{P}^{(c)}_{s_{i-1},s_i}\right)\mathbb{P}\{C = j\}\right)}{\left(\prod_{b\in\mathcal{B}(T)} \prod_{s_i\in b} \mathcal{P}^{(c)}_{s_{i-1},s_i}\right)\mathbb{P}\{C = c\}} \right) \right\} \tag{20}$$

$$= \arg\min_{c\in\mathcal{C}} \left\{ \log\left(\left(\prod_{b\in\mathcal{B}(T)} \prod_{s_i\in b} \mathcal{P}^{(c)}_{s_{i-1},s_i} \right)\mathbb{P}\{C = c\} \right) - \log\left(\sum_{j=1}^{|\mathcal{C}|} \left(\left(\prod_{b\in\mathcal{B}(T)} \prod_{s_i\in b} \mathcal{P}^{(c)}_{s_{i-1},s_i} \right)\mathbb{P}\{C = j\} \right) \right) \right\} \tag{21}$$

$$= \arg\min_{c \in \mathcal{C}} \left\{ \left(\sum_{b \in \mathcal{B}(T)} \sum_{s_i \in b} \log\left(\mathcal{P}^{(c)}_{s_{i-1}, s_i} \right) + \log\left(\mathbb{P}\left\{ C = c \right\} \right) \right) \right\} \tag{22}$$

Note the second term in equation 21 is constant for all $c \in \mathcal{C}$, therefore we do not need to consider when minimizing. This completes the proof. ■

A.3 Proof of Lemma 4

Proof. Recall $\widetilde{\mathbf{P}} \triangleq (1-\varepsilon)\mathbf{P} + \varepsilon\mathbf{R}$. We will first prove $\widetilde{\mathbf{P}}$ is an irreducible matrix, i.e. we will prove for all $i \in [n], j \in [n]$, there exists $k \in \mathbb{N}$, s.t. $\widetilde{\mathbf{P}}^k_{i,j} > 0$. After adding $\mathbf{R}$, we know have $\mathbb{P}\left[s_i \to s_0 | 1 \right] = \varepsilon$. Thus we have,

$$\mathbb{P}\left[s_i \to s_j || \pi(s_i, s_j)| = n \right] \geq \mathbb{P}\left[s_i \to s_0 || \pi(s_i, s_0)| = 1 \right] \cdot \mathbb{P}\left[s_0 \to s_j || \pi(s_0, s_{j-1})| = n-1 \right] = \varepsilon \mathcal{P}_{0,j} > 0 \tag{23}$$

when we choose n s.t. the distance from the root to s_j is equal to $n-1$. We have therefore proved $\widetilde{\mathbf{P}}$ is irreducible. We will now prove aperiodicity. It suffices to prove $\mathbb{P}[s_i \to s_j | n+1] > 0$.

$$\mathbb{P}\left[s_i \to s_j || \pi(s_i, s_j)| = n+1 \right] \tag{24}$$

$$\geq \mathbb{P}\left[s_i \to s_0 || \pi(s_i, s_0)| = 1 \right] \cdot \mathbb{P}\left[s_0 \to s_i | 1 \right] \cdot \mathbb{P}\left[s_0 \to s_0 || \pi(s_0, s_0)| = 1 \right] = \varepsilon^2 \mathcal{P}_{0,j} > 0 \tag{25}$$

Thus we have proved $\mathbb{P}\left[s_i \to s_j | n \right] > 0$ and $\mathbb{P}\left[s_i \to s_j | n+1 \right] > 0$, we also have the greatest common divisor for n and $n+1$ is equal to 1. Therefore there exists no t s.t. $\mathbb{P}\left[s_i \to s_j | tn \right] = 0$. Thus $\widetilde{\mathbf{P}}$ is aperiodic. This completes the proof. ■

A.4 Proof of Theorem 5

Before we prove Theorem 5, we will introduce two necessary results.

Lemma 9. *Let $\boldsymbol{v}$ and $\widetilde{\boldsymbol{v}}$ be vectors. s.t. $\|\boldsymbol{v}\| = \|\widetilde{\boldsymbol{v}}\| = 1$ and $\widetilde{\boldsymbol{v}}^\top \boldsymbol{v} \geq 0$, then*

$$\|\boldsymbol{v} - \widetilde{\boldsymbol{v}}\| \leq \sqrt{2} \sin\Theta\left(\boldsymbol{v}, \widetilde{\boldsymbol{v}} \right) \tag{26}$$

Proof.

$$\sin^2\Theta\left(\boldsymbol{v}, \widetilde{\boldsymbol{v}} \right) = 1 - \left(\boldsymbol{v}^\top \widetilde{\boldsymbol{v}} \right)^2 \overset{\zeta_1}{\geq} 1 - \boldsymbol{v}^\top \widetilde{\boldsymbol{v}} = 1 + \frac{1}{2}\|\widetilde{\boldsymbol{v}} - \boldsymbol{v}\|^2 - \frac{1}{2}\|\boldsymbol{v}\|^2 - \frac{1}{2}\|\widetilde{\boldsymbol{v}}\|^2 = \frac{1}{2}\|\boldsymbol{v} - \widetilde{\boldsymbol{v}}\|^2 \tag{27}$$

(ζ_1) follows from $0 \leq \boldsymbol{v}^\top \widetilde{\boldsymbol{v}} \leq 1$, therefore $\boldsymbol{v}^\top \widetilde{\boldsymbol{v}} \geq (\boldsymbol{v}^\top \widetilde{\boldsymbol{v}})^2$. Plugging this back into the first inequality and taking the square root gives us the desired result. ■

Theorem 10 (Modified Wedin's Theorem, O'Rourke et al. (2018))**.** *Let $\widetilde{\boldsymbol{\pi}}$ be the dominant eigenvector for $\widetilde{\mathbf{M}}$ and let $\boldsymbol{\pi}$ be the dominant eigenvector for $\mathbf{M}$. It then follows for probabilistic matrices $\mathbf{M}$ and $\widetilde{\mathbf{M}}$,*

$$\sin\Theta\left(\boldsymbol{\pi}, \widetilde{\boldsymbol{\pi}} \right) \leq 2 \left\| \mathbf{M} - \widetilde{\mathbf{M}} \right\|_{\mathrm{F}} \tag{28}$$

Now we are ready to prove Theorem 5.

Proof of Theorem 5. In our algorithm, we find the stationary distribution of the tree, and from there do a sum-aggregation in to the Markov Chain Representation. It is clear this is equivalent to first converting all the nodes in the tree to the their Markov Chain State and then finding the steady state distribution since we do a sum-aggregation. Therefore, we will give an upper bound for $\left\| \widetilde{\mathbf{M}} - \mathbf{M} \right\|_{\mathrm{F}}$ in the case of leaf insertion or leaf deletion, where $\mathbf{M}$ represents the Markov Chain Probability matrix of T and $\widetilde{\mathbf{M}}$ represents the Markov Chain Probability Matrix of $\widetilde{T}$. We first note, that by the Markov Chain Representation, a leaf insertion/deletion has the same difference in the Frobenius Norm. There will only be changes in the distribution from two levels underneath the deletion/insertion. WLOG let us consider a node deletion (as

this same as insertion of a node into the graph after deleting a node). Let us delete a leaf v, with parent p, and grandparent $\bar{p}$. Then we have,

$$\begin{aligned}
\left\|\mathbf{M}-\widetilde{\mathbf{M}}\right\|_{\mathrm{F}} = \Bigg(&\underbrace{\sum_{k=0}^{D}\left(\mathbb{P}\left\{(\rho(p),\Gamma(p))\to(\rho(p)+1,k)\right\}-\widetilde{\mathbb{P}}\left\{(\rho(p),\Gamma(p))\to(\rho(p)+1,k)\right\}\right)^2}_{\textcircled{\mathrm{T}_1}} \\
&+\underbrace{\sum_{k=0}^{D}\left(\mathbb{P}\left\{(\rho(p),\Gamma(p)-1)\to(\rho(p)+1,k)\right\}-\widetilde{\mathbb{P}}\left\{(\rho(p),\Gamma(p)-1)\to(\rho(p)+1,k)\right\}\right)^2}_{\textcircled{\mathrm{T}_2}} \\
&+\underbrace{\sum_{k=0}^{D}\left(\mathbb{P}\left\{(\rho(p)-1,\Gamma(\bar{p}))\to(\rho(p),k)\right\}-\widetilde{\mathbb{P}}\left\{(\rho(p)-1,\Gamma(\bar{p}))\to(\rho(p),k)\right\}\right)^2}_{\textcircled{\mathrm{T}_3}}\Bigg)^{1/2} \qquad (29)\\
&\overset{\zeta_1}{\leq}\left(2\sum_{k=0}^{D}\left(\mathbb{P}\left\{(\rho(p),\Gamma(p))\to(\rho(p)+1,k)\right\}\right)^2+\frac{2}{\mathcal{N}\left((\rho(\bar{p},\Gamma(\bar{p}))\to\cdot\right)^2}\right)^{1/2} \qquad (30)\\
&\overset{\text{Cauchy}-\text{Schwarz}}{\leq}\sqrt{2}\left(\left(\sum_{k=0}^{D}\left(\mathbb{P}\left\{(\rho(p),\Gamma(p))\to(\rho(p)+1,k)\right\}\right)\right)+\frac{1}{\mathcal{N}\left((\rho(\bar{p},\Gamma(\bar{p}))\to\cdot\right)}\right) \qquad (31)\\
&=\sqrt{2}\left(1+\frac{1}{\mathcal{N}\left((\rho(\bar{p},\Gamma(\bar{p}))\to\cdot\right)}\right) \qquad (32)\\
&\leq 2\sqrt{2} \qquad (33)
\end{aligned}$$

(ζ_1) follows as we are essentially transfering branches from $(\rho(p),\Gamma(p)$ to $(\rho(p),\Gamma(p)-1)$ because of the deletion of the node. Therefore maximally, if there were no existing branches in $(\rho(p),\Gamma(p)-1)$, then all the branches get transferred over. In T_1, we consider the difference in the structure of the parent. Let $s_p=(\rho(p,\Gamma(p)$ be the Markov State associated with p. In T_2, we consider the difference in the structure of the new parent. In T_3, we consider the difference in the structure of grandparent. The probability changes in the Markov Chain are unaffected past this point, as we are only considering the deletion of a leaf. If we are to consider the transition from matrix $\mathbf{M}$ to $\widetilde{\mathbf{M}}$ as a sequence of matrices, i.e. $\mathbf{M}_1$ represents one leaf insertion/deletion and $\mathbf{M}_j$ represents the matrix $\mathbf{M}$ after j such insertion/deletions. Then we can write a transition sequence as $\mathbf{M}\triangleq\mathbf{M}_0\to\mathbf{M}_1\to\mathbf{M}_2\to\cdots\to\mathbf{M}_R\triangleq\widetilde{\mathbf{M}}$. Then it follows,

$$\begin{aligned}
\|\mathbf{M}_0-\mathbf{M}_R\|_{\mathrm{F}} &= \|\mathbf{M}_0-\mathbf{M}_1+\mathbf{M}_1-\mathbf{M}_R\|_{\mathrm{F}} && (34)\\
&\leq \|\mathbf{M}_0-\mathbf{M}_1\|_{\mathrm{F}}+\|\mathbf{M}_1-\mathbf{M}_R\|_{\mathrm{F}} && (35)\\
&\overset{\zeta_1}{\leq}\sum_{i=1}^{R}\|\mathbf{M}_i-\mathbf{M}_{i-1}\|_{\mathrm{F}} && (36)\\
&\overset{\text{eq. }33}{\leq} 2R\sqrt{2} && (37)
\end{aligned}$$

In (ζ_1) we simply extend the idea from equation 35 to the R transitions. The bound in equation 37 is worst-case. Now we can bound $\|\boldsymbol{\pi}-\widetilde{\boldsymbol{\pi}}\|$,

$$\|\boldsymbol{\pi}-\widetilde{\boldsymbol{\pi}}\| \overset{\text{lem. }9}{\leq} \sqrt{2}\sin\Theta\left(\boldsymbol{\pi},\widetilde{\boldsymbol{\pi}}\right) \overset{\text{thm. }10}{\leq} 2\sqrt{2}\left\|\mathbf{M}-\widetilde{\mathbf{M}}\right\| \leq \frac{4R\sqrt{2}}{1-\sigma_2\left(\mathbf{M}\right)} \qquad (38)$$

By the aperiodicity we proved in lemma 4, we know $\sigma_2(\mathbf{M})<1$. This concludes the proof. ■

A.5 Proof of Theorem 8

Proof. From the definition of the Expected Hitting Distance given in Definition 7, we have

$$\mathbb{E}\left[D_{iA}\right] = \mathbb{E}\left[D_A | \text{initial state is } s_i\right] \tag{39}$$

$$= \sum_{j=1}^{|\mathcal{S}|} \mathcal{P}_{s_i,s_j} \left(d(s_i,s_j) + \mathbb{E}\left[D_{s_j,A}\right]\right) \tag{40}$$

$$= \sum_{j=1}^{|\mathcal{S}|} \mathcal{P}_{s_i,s_j} \left(d(s_i,s_j) + \left(\sum_{k=1}^{|\mathcal{S}|} \mathcal{P}_{s_j,s_k} \left(d(s_j,s_k) + \cdots\right)\right)\right) \tag{41}$$

$$= \sum_{j=1}^{|\mathcal{S}|} d(s_i,s_j)\mathcal{P}_{s_i,s_j} + \sum_{j=1}^{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} \mathcal{P}_{s_i,s_j,s_k} d(s_j,s_k) + \cdots \tag{42}$$

$$\overset{\zeta_1}{=} \sum_{j=1}^{|\mathcal{S}|}\sum_{k=1}^{|\mathcal{S}|} d(s_j,s_k) \sum_{r=1}^{\infty} \mathbb{P}\left\{s_j \to s_k\right\} \mathbb{P}\left\{s_i \to s_j || \pi\left(s_0,s_i\right)| = r-1\right\} \tag{43}$$

$$\overset{\zeta_2}{=} \sum_{(j,k)\in\mathcal{E}(T)} \ell\left(e_{j,k}\right) \sum_{r=1}^{\infty} \mathbb{P}\left\{e_{i,\cdot} \to e_{j,k} || \pi\left(e_{i,\cdot}, e_{j,k}\right)| = r\right\} \tag{44}$$

$$\overset{\zeta_3}{=} \sum_{(j,k)\in\mathcal{E}(T)} \ell(e_{j,k}) \sum_{r=1}^{\infty} \left(\mathbf{Q}^k\right)_{(i,\cdot)\to(j,k)} \overset{\zeta_4}{=} \sum_{(j,k)\in\mathcal{E}(T)} \ell(e_{j,k}) \left(\left(\mathbf{I}-\mathbf{Q}\right)^{-1}\right)_{(i,\cdot)\to(j,k)} \overset{\zeta_5}{=} \left(\left(\mathbf{I}-\mathbf{Q}\right)^{-1}\boldsymbol{\ell}\right)_{(i,\cdot)} \tag{45}$$

In (ζ_1), we expand out all the inner summations as we note all the terms are equivalent to the distance multiplied by the probability of the occurrence of a path, we make the observation this path starts at s_i and ends at s_j. Furthermore, we also make the note this covers all possible paths as we are summing over all possible states. In (ζ_2), we use the equivalent distribution of the edges of the tree to simplify the analysis, we also let $e_{\cdot,i}$ represent all edges s.t. the second node of the edge is s_i. In (ζ_3), if we let $\mathbf{Q}$ be defined as in the original theorem statement, then we have $\left(\mathbf{Q}^r\right)_{(i,\cdot)\to(j,k)}$ is the probability after r steps, a random walker in the edge with states s_i and one of it's neighbors will be in the edge connecting s_j and s_k. In (ζ_4), since $\mathbf{Q}$ is a probabilistic matrix, therefore $\|\mathbf{Q}\| = 1$, thus we have

$$\sum_{k=1}^{\infty} \mathbf{Q}^k = \left(\mathbf{I}-\mathbf{Q}\right)^{-1} \tag{46}$$

In (ζ_5), we define $\boldsymbol{\ell}$ as the vector of edge lengths given in the Theorem 8 statement. This completes the proof. ■

B Illustrative Example

In this section, we will give an illustrative example of our stationary distribution and expected distance Markov Chain methods on a small tree graph. The matrices for the toy example in Figure 11 are as follows:

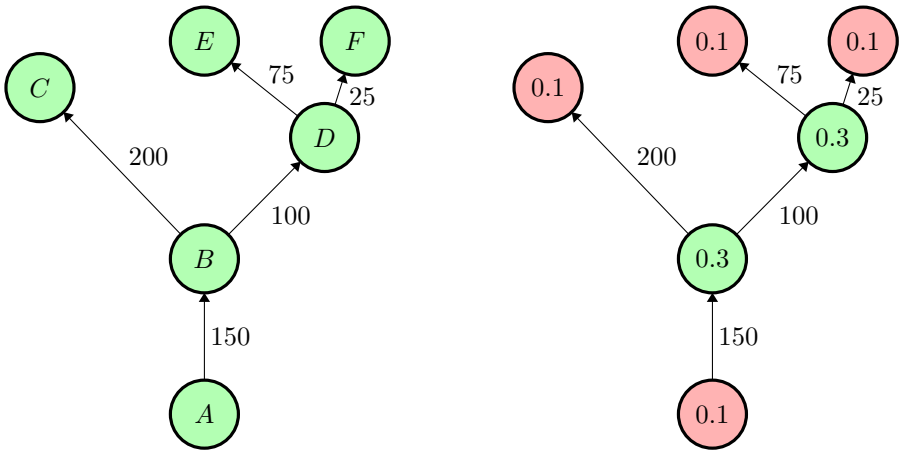


Figure 11: Stationary Distribution of Toy Example

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 1/3 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \rightarrow \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array}\!\!\begin{array}{c} \\ \begin{pmatrix} 0.1 \\ 0.3 \\ 0.1 \\ 0.3 \\ 0.1 \\ 0.1 \end{pmatrix} \end{array} \tag{47}$$

In Equation 47, we go from the adjacency matrix of the tree given in Figure 11 to the probabilistic transition matrix within the tree itself, then we obtain the stationary distribution. If we then consider the state space with $\mathcal{S}$ with maximum outgoing degree 2 and maximum distance from root 3. The feature vector for the toy example in Figure 11 is then:

$$\Phi(X) = \Big(\underbrace{\begin{array}{ccc} 0 & 0.1 & 0 \end{array}}_{\rho(i)=0} \underbrace{\begin{array}{ccc} 0 & 0 & 0.3 \end{array}}_{\rho(i)=1} \underbrace{\begin{array}{ccc} 0.1 & 0 & 0.3 \end{array}}_{\rho(i)=2} \underbrace{\begin{array}{ccc} 0.2 & 0 & 0 \end{array}}_{\rho(i)=3} \Big) \tag{48}$$

$$\mathbf{E} = \begin{array}{c} \\ A \\ AB \\ BA \\ BC \\ CB \\ BD \\ DB \\ DE \\ ED \\ DF \\ FD \end{array}\!\!\begin{array}{c} \begin{array}{ccccccccccc} A & AB & BA & BC & CB & BD & DB & DE & ED & DF & FD \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array} \tag{49}$$

In Equation 49, we obtain the edge transition matrix of the tree in Figure 12. From here, we calculate the expected distance from the root (A, also the start node which is unioned with the rest of the edges) to each of the nodes. If we then consider the state space $\mathcal{S}$, with maximum outgoing degree 2 and maximum distance from root 3. The feature vector for the toy example in Figure 11 is then:

$$\Phi(X) = \Big(\underbrace{\begin{array}{ccc} 0 & 0 & 0 \end{array}}_{\rho(i)=0} \underbrace{\begin{array}{ccc} 0 & 0 & 150 \end{array}}_{\rho(i)=1} \underbrace{\begin{array}{ccc} 1050 & 0 & 950 \end{array}}_{\rho(i)=2} \underbrace{\begin{array}{ccc} 4000 & 0 & 0 \end{array}}_{\rho(i)=3} \Big) \tag{50}$$

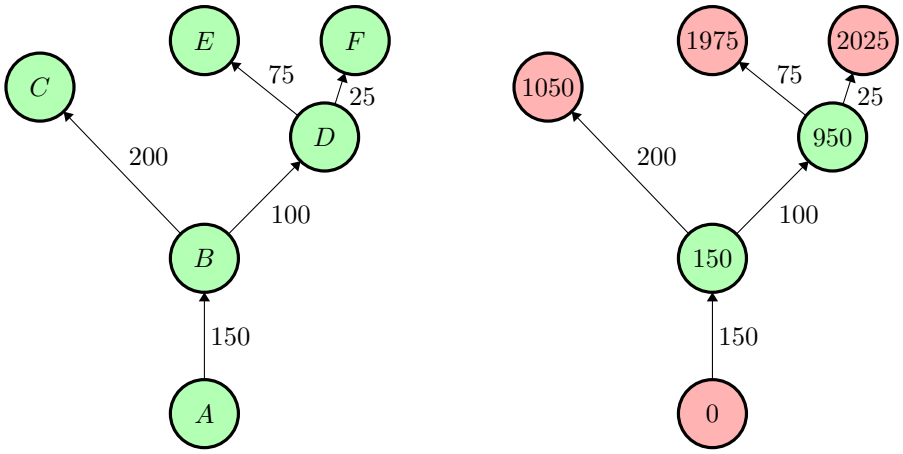


Figure 12: Expected Distance Representation of Toy Example

C Neural Network Training and Testing

In this section we will give all train and test curves ommited from the main text.

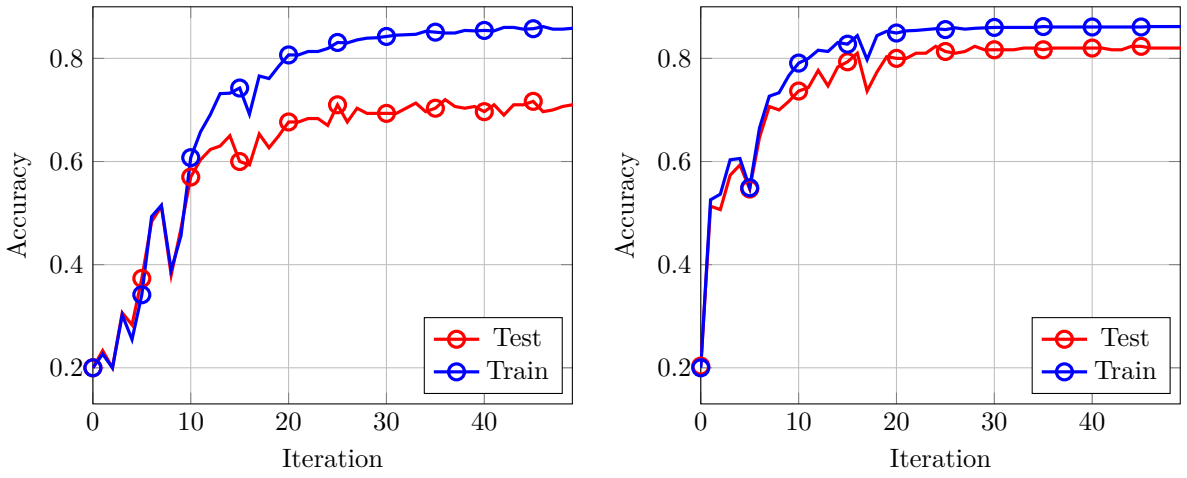


Figure 13: Training and testing results on 750 synthetic images (5 classes, 120 train, 30 test) from synthetically developed trees without background are displayed after training on `AlexNet` in *(Left)*. Training and testing results on 750 synthetic images (5 classes, 120 train, 30 test) from synthetically developed trees with random items in the background unique to each class are displayed after training on `AlexNet` in *(Right)*.

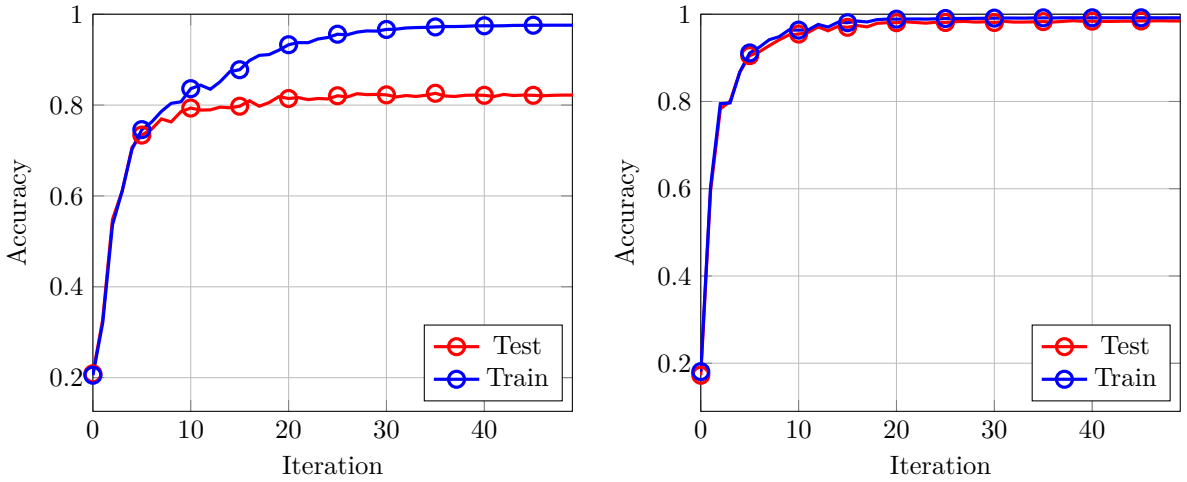


Figure 14: Training and testing results on 5000 synthetic images (5 classes, 800 train, 200 test) from synthetically developed trees with random items in the background unique to each class are displayed after training on `AlexNet` in *(Right)*. As we see in the Background to Background case, more data helps with the training. Furthermore, we see better generalization with more data in the blank case, but we do not see a high increase in train accuracy. This leads us to believe learning the blank trees is in general difficult.

C.1 Experimental Details

All experiments were run on a single RTX 2070 GPU and intel i5-8600K 6-core processor in Pytorch Paszke et al. (2019).

(a) (b) (c) (d) (e)

Figure 15: We show 5 pictures of a tree going from no leaves to full foliage.

D Dataset Details

In this section, we give more details of our dataset. We take all photos on iPhone 13 and Google Pixel phones. All photos are taken in high resolution and resized to 600×800 to standardize across the different devices. Each day for one and a half months from April to May, at various times throughout the day, and different weathers (e.g. rain, cloudy, sunny) we took approximately 5 pictures of the trees from different angles and from various distances from the tree. Due to the timing of when we took the pictures of the trees, from the time we started to the time we stopped taking pictures, the trees went from no leaves to having full foliage. One extension of this dataset is to continue taking pictures until the trees develop fruits. In total, we took approximately 150 pictures of 55 different trees of various species and locations. All this came to a total of 8221 pictures. We will now list some of the difficulties of classification in our dataset.

1. Occlusion. As we see in Figure 15a, there is no occulsion and in Figure 15e there is full occlusion. This means valuable structure is no longer visible.

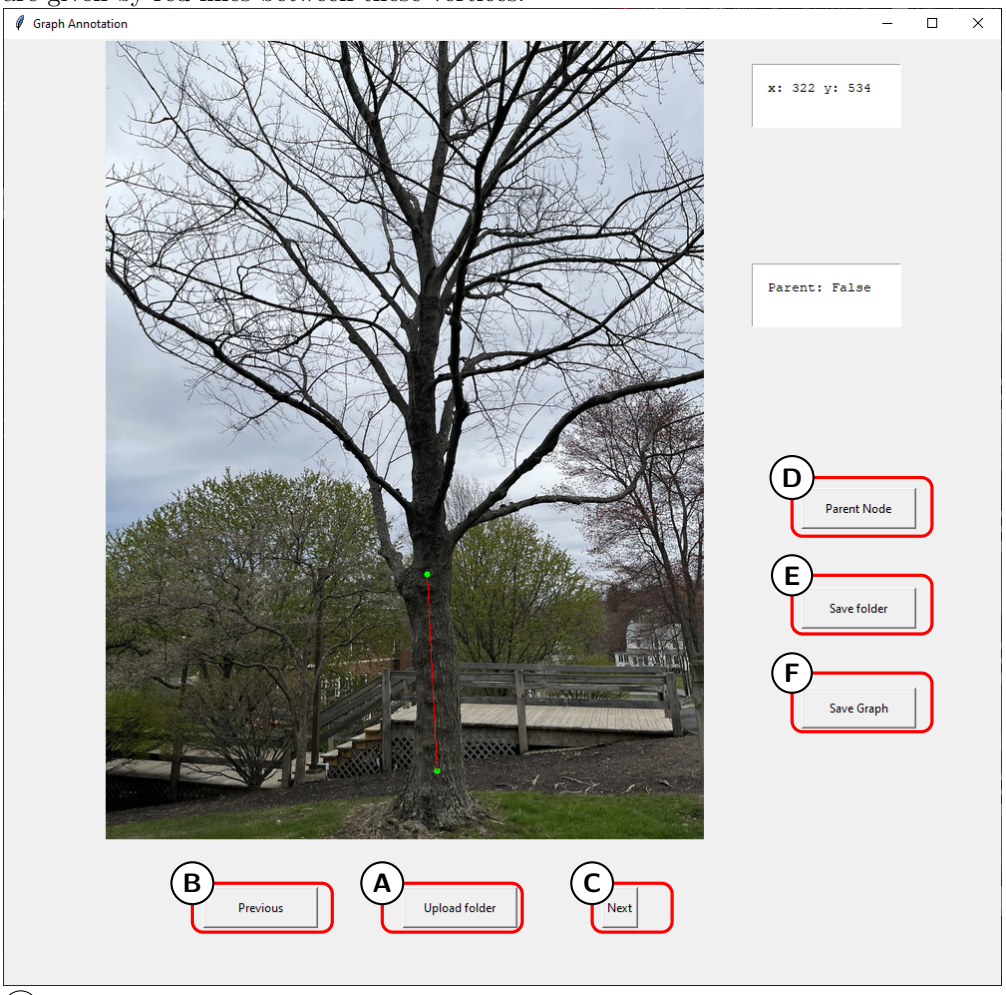
2. Background Conditions. In Figures 15a and 15b we can see the background is much more cloudy than in Figure 15c and 15e.

3. Viewpoint changes. The images are taken from around the tree at various locations.

Thus, this dataset poses to be a challenge for the Computer Vision Community to develop an algorithm robust to occlusions, changing backgrounds, and viewpoint changes.

E Geometric Tree Generation Interface

We provide details on our software implementation of a Geometric Tree Generation Interface. We will in detail describe all the features of our software. In general, vertices are denoted by green circles and edges are given by red lines between these vertices.



(A) Allows the user to open a directory of PNG or JPG images to annotate.

(B) & (C) When in the directory of images, you can press Previous and Next to iterate through the directory to open new unannotated images.

(D) There are two modes the user can be in while annotating the graph. Either in *parent* mode or not in *parent* mode. If the user is in *parent* mode, then either the user can place a new node which will be a *parent*, or if the user clicks within 5 pixels of another node, then that node will become a parent. After you click on parent mode, you will be taken out of *parent* mode. When you are not in parent mode, any node you click will have an edge from the *parent* node. If you want to delete any nodes, click middle-button on the mouse and it will delete the whole subtree of the node within 5 pixels of your click point.

(E) The user can select a directory for the pickle files to be saved after *Save Graph* is clicked.

(F) The graph is saved in a pickle file to the save directory the user already chose as an adjacency list. The keys in the adjacency list are the pixel locations of the nodes, and the list is the pixel locations of the neighbor nodes.