# Prototype Propagation Networks for Weakly-supervised Few-shot Learning on Category Graph

**Lu Liu** [1] **Tianyi Zhou** [2] **Guodong Long** [1] **Jing Jiang** [1]

## Abstract

A variety of machine learning applications expect to achieve rapid learning from limited labeled data. However, the success of most current models relies on heavy training and big data. Meta-learning addresses this problem by extracting common knowledge across different tasks that can be quickly adapted to new tasks. However, they rarely explore weakly-supervised information, which is usually free or cheap to collect. In this paper, we show that weakly-labeled data can significantly improve the performance of meta-learning on few-shot classification. We propose *prototype propagation network (PPN)* trained on few-shot tasks together with coarsely annotated data. Given a category graph of the targeted fine-classes and some weakly-labeled coarse-classes, PPN learns to propagate the prototype of one class to another on the graph, so that the K-nearest neighbor (KNN) classifier defined on the propagated prototypes results in high accuracy across different tasks. The training tasks are generated by subgraph sampling, and the training objective is the accumulated level-wise classification loss on the subgraph. The resulting graph of prototypes can be reused and updated for new classes. On two benchmarks, PPN significantly outperforms most recent few-shot learning methods in two settings, even when they are also allowed to train on weakly-labeled data.

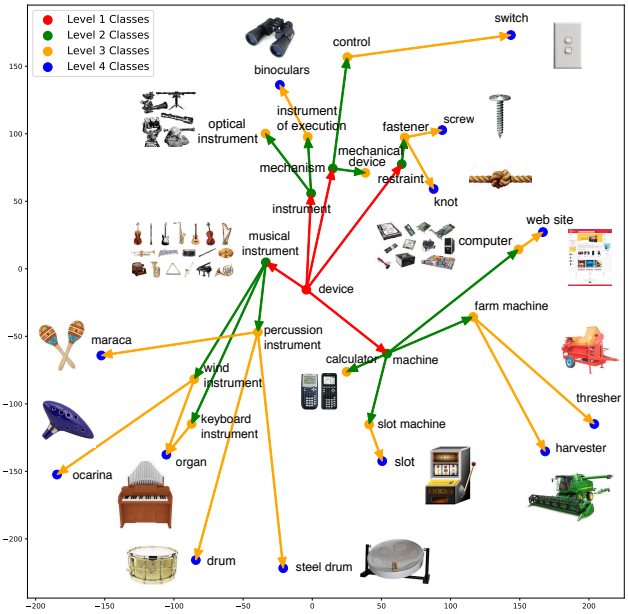


*Figure 1.* Prototypes learned by PPN and transformed to a 2D space by t-SNE. Each edge connects a child class to a parent. The prototypes spread out as classes become finer, preserve the graph structure, and reflect the semantic similarity.

## 1. Introduction

Machine learning (ML) has achieved breakthrough in various application fields. Nowadays, we can train powerful deep neural networks containing thousands of layers on millions of data within an acceptable time with expensive hardware. However, as AI becomes democratized for personal use with concerns about data privacy, demand is rapidly growing for few-shot learning of highly customized models on edge/mobile devices with limited data..

This few-shot learning problem can be addressed by a class of approaches called "meta-learning", which learns the common knowledge shared across different tasks, or "learning to learn". For example, it can be shared initialization weights (Finn et al., 2017), an optimization algorithm (Ravi & Larochelle, 2017) or a distance/similarity metric (Vinyals et al., 2016). In contrast to single-task learning, the "training data" in meta-learning are tasks and the goal is to maximize the validation accuracy of these sampled tasks.

Although recent meta-learning studies have shown success on few-shot learning, most do not leverage weakly-

[1]Center for Artificial Intelligence, FEIT, University of Technology Sydney [2]Paul G. Allen School of Computer Science & Engineering, University of Washington. Correspondence to: Guodong Long <guodong.long@uts.edu.au>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data 

Code at: https://github.com/liulu112601/PPN

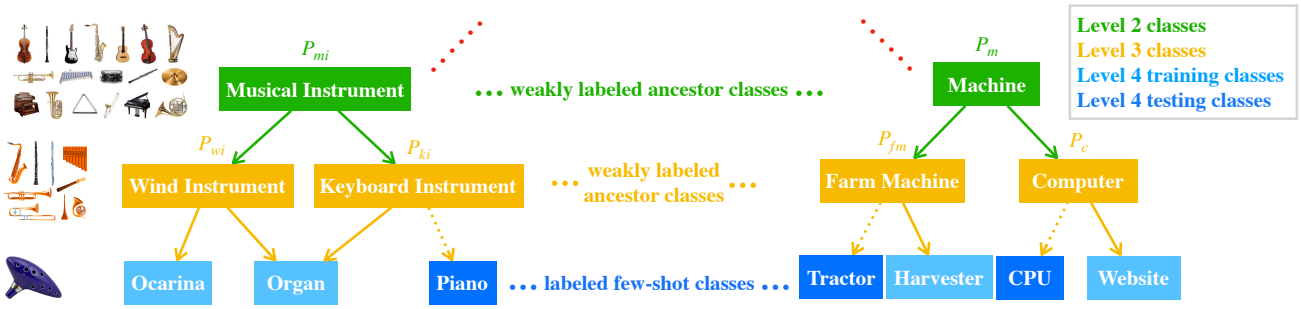


*Figure 2.* The few-shot classes (leaf nodes) in training and test tasks are non-overlapping, but they are allowed to share some ancestor classes. Weakly-labeled data only has non-leaf class labels. PPN is trained on classification tasks with both fine and coarse classes.

supervised information, which has been proved to be helpful when training data is limited, e.g., in weakly-supervised (Zhou, 2017) and semi-supervised learning (Zhu & Ghahramani, 2002). In this paper, we show that weakly-supervised information, e.g., an image of a Tractor with a coarse label of Machine, can significantly improve the performance of meta-learning on few-shot classification. We additionally assume that a category graph (Fig. 2) describing the class relationship is available.

We propose a meta-learning model called prototype propagation network (PPN) to explore the above weakly-supervised information for few-shot classification tasks. PPN produces a prototype per class by propagating the prototype of each class to its child classes on the category graph, where an attention mechanism generates the edge weights used for propagation. The learning goal of PPN is to minimize the validation errors of a KNN classifier built on the propagated prototypes for few-shot classification tasks. The classification error on weakly-labeled data can thus be back-propagated to improve the prototypes of other classes. The resulting graph of prototypes can be repeatedly used, updated and augmented on new tasks as an episodic memory.

To fully explore the weakly-labeled data, we develop a level-wise method to train tasks, generated by subgraph sampling, for both coarse and fine classes on the graph. In addition, we introduce two testing settings that are common in different practical application scenarios: one (PPN+) is allowed to use weakly-labeled data in testing tasks and is given the edges connecting test classes to the category graph, while the other (PPN) cannot access any extra information except for the few-shot training data of the new tasks. In experiments, we extracted two benchmark datasets from ILSVRC-12 (ImageNet) (Deng et al., 2009), specifically for weakly-supervised few-shot learning. In different test settings, our method consistently and significantly outperforms the three most recently proposed few-shot learning models and their variants, which also trained on weakly-labeled data. The prototypes learned by PPN is visualized in Fig. 1 (Maaten & Hinton, 2008).

## 2. Prototype Propagation Networks (PPN)

### 2.1. Weakly-Supervised Few-Shot Learning

In weakly-supervised few-shot learning, we learn from two types of data: the few-shot data $\mathcal{X}$ annotated by the target fine-class labels $\mathcal{Y}$ and the weakly-labeled data $\mathcal{X}^w$ annotated by coarse-class labels $\mathcal{Y}^w$. We assume a directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{Y} \cup \mathcal{Y}^w, E)$ exists, where each node $y \in \mathcal{Y} \cup \mathcal{Y}^w$ denotes a class, and each edge $z \to y \in E$ connects a parent class $z$ to one of its child classes $y$.

We follow the setting of few-shot learning, where each task $T$ is defined by a subset of classes. In our problem, as shown by Fig. 2, the few-shot classes used for training and test are non-overlapping, but we allow them to share some ancestors on the graph. We also allow training tasks to cover any classes on the graph. The training aims to solve the risk minimization of all training tasks. Similar to prototypical networks (Snell et al., 2017), Given a data point $x$, we first compute its representation $f(x) \in \mathbb{R}^d$, where $f(\cdot)$ is convolutional neural networks (CNN) with parameter $\Theta^{cnn}$, then the prediction is computed as

$$\Pr(y|\boldsymbol{x}; \boldsymbol{P}_{\mathcal{Y}^T}) \triangleq \frac{\exp(-\|f(\boldsymbol{x}) - \boldsymbol{P}_y)\|^2)}{\sum_{z \in \mathcal{Y}^T} \exp(-\|f(\boldsymbol{x}) - \boldsymbol{P}_z)\|^2)}, \quad (1)$$

In the following, we will introduce prototype propagation which generates $\boldsymbol{P}_{\mathcal{Y}^T}$ for any task $T$.

### 2.2. Prototype Propagation

In PPN, each training task $T$ is a level-wise classification on a sampled subgraph $\mathcal{G}_i \subseteq \mathcal{G}$, i.e., a classification task over $\mathcal{Y}^T = \mathcal{Y}^{\mathcal{G}_i^j}$, where $\mathcal{G}_i^j$ denotes the level-$j$ of subgraph $\mathcal{G}_i$ and $\mathcal{Y}^{\mathcal{G}_i^j}$ is the set of all the classes on $\mathcal{G}_i^j$. The prototype propagation is defined on each subgraph $\mathcal{G}_i$, which covers classes $\mathcal{Y}^{\mathcal{G}_i}$. Given the associated training data $\mathcal{X}^y$ for class $y \in \mathcal{Y}^{\mathcal{G}_i}$, the prototype of class $y$ is initialized by averaging the representations $f(\boldsymbol{x})$ of samples $\boldsymbol{x} \in \mathcal{X}^y$, i.e.,

$$\boldsymbol{P}_y^0 \triangleq \frac{1}{|\mathcal{X}^y|} \sum_{\boldsymbol{x} \in \mathcal{X}^y} f(\boldsymbol{x}). \quad (2)$$

For each parent class $z$ of class $y$ on $\mathcal{G}_i$, we propagate $\boldsymbol{P}_z^0$ to class $y$ with edge weight $a(\boldsymbol{P}_y^0, \boldsymbol{P}_z^0)$ measuring the similarity between class $y$ and $z$, and aggregate the propagation (the messages) from all the parent classes by

$$\boldsymbol{P}_y^+ \triangleq \sum_{z \in \mathcal{P}_y^{\mathcal{G}_i}} a(\boldsymbol{P}_y^0, \boldsymbol{P}_z^0) \times \boldsymbol{P}_z^0, \tag{3}$$

where $\mathcal{P}_y^{\mathcal{G}_i}$ denotes the set of all parent classes of $y$ on subgraph $\mathcal{G}_i$, and the edge weight $a(\boldsymbol{P}_y^0, \boldsymbol{P}_z^0)$ is a learnable similarity metric defined by dot-product attention (Vaswani et al., 2017), i.e.,

$$a(\boldsymbol{p}, \boldsymbol{q}) \triangleq \frac{\langle g(\boldsymbol{p}), h(\boldsymbol{q}) \rangle}{\|g(\boldsymbol{p})\| \times \|h(\boldsymbol{q})\|}, \tag{4}$$

where $g(\cdot)$ and $h(\cdot)$ are learnable transformations applied to prototypes with parameters $\Theta^{att}$, e.g., linear transformations $g(\boldsymbol{p}) = \boldsymbol{W}_g \boldsymbol{p}$ and $h(\boldsymbol{q}) = \boldsymbol{W}_h \boldsymbol{q}$.

The prototype after propagation is a weighted average of $\boldsymbol{P}_y^0$ and $\boldsymbol{P}_y^+$ with weight $\lambda \in [0, 1]$, i.e.,

$$\boldsymbol{P}_y \triangleq \lambda \times \boldsymbol{P}_y^0 + (1 - \lambda) \times \boldsymbol{P}_y^+. \tag{5}$$

For each classification task $T$ on subgraph $\mathcal{G}_i$, $\boldsymbol{P}_y$ is used in Eq. (1) to produce the likelihood probability.

### 2.3. Level-wise Training of PPN on Subgraphs

The goal of meta-training is to learn a parameterized propagation mechanism defined by Eq. (2)-Eq. (5) on few-shot tasks. In each iteration, we randomly sample a subset of few-shot classes, which together with all their ancestor classes and edges on $\mathcal{G}$ form a subgraph $\mathcal{G}_i$. A training task $T$ is drawn from each level $\mathcal{G}_i^j \sim \mathcal{G}_i$ as the classification over classes $\mathcal{Y}^T = \mathcal{Y}^{\mathcal{G}_i^j}$. The goal in our problem is defined as:

$$\min_{\Theta^{cnn}, \Theta^{att}} \sum_{\mathcal{G}_i \sim \mathcal{G}} \sum_{\mathcal{G}_i^j \sim \mathcal{G}_i} \sum_{(x,y) \sim \mathcal{D}^{\mathcal{G}_i^j}} -\log \Pr(y | x; \boldsymbol{P}_{\mathcal{Y}^{\mathcal{G}_i^j}}), \tag{6}$$

where $\mathcal{D}^{\mathcal{G}_i^j}$ is the data distribution of data-label pair $(x, y)$ from classes $\mathcal{Y}^{\mathcal{G}_i^j}$. Since the prototype propagation is defined on the whole subgraph, it generates a computational graph relating each class to all of its ancestor classes. Hence, during training, the classification error on each class is back-propagated to the prototypes of its ancestor classes, which will be updated to improve the validation accuracy of finer classes and propagated to generate the prototypes of the few-shot classes later.

The complete level-wise training procedure of PPN is given in Algorithm 1, each of whose iterations comprises two main stages: prototype propagation (lines 9-11) which builds a computational graph over prototypes of the classes on a sampled subgraph $\mathcal{G}_i$, and level-wise training (lines 12-16) which updates the parameters $\Theta^{cnn}$ and $\Theta^{att}$ on per-level classification tasks. To improve computational efficiency, we lazily update $\boldsymbol{P}_y^0$ for all classes $y \in \mathcal{Y} \cup \mathcal{Y}^w$ every $m$ epochs, as shown in lines 3-7.

**Algorithm 1** Level-wise Training of PPN

**Input:** few-shot data $\mathcal{X}$ with labels from $\mathcal{Y}$,
weakly-labeled data $\mathcal{X}^w$ with labels from $\mathcal{Y}^w$,
category graph $\mathcal{G} = (\mathcal{Y} \cup \mathcal{Y}^w, E)$,
learning rate scheduler for SGD, $\lambda$, $m$;

1: **Initialize:** randomly initialize $\boldsymbol{P}$, $\Theta^{cnn}$, $\Theta^{att}$, $\tau \leftarrow 0$;
2: **while** not converge **do**
3: **if** $\tau \bmod m = 0$ **then**
4: **for** class $y \in \mathcal{Y} \cup \mathcal{Y}^w$ **do**
5: Update $\boldsymbol{P}_y^0$ by Eq. (2) and save it in buffer;
6: **end for**
7: **end if**
8: Sample a subgraph $\mathcal{G}_i \sim \mathcal{G}$;
9: **for** class $y \in \mathcal{Y}^{\mathcal{G}_i}$ **do**
10: Get $\boldsymbol{P}_y^0$ from buffer, and propagate by Eq. (3)-(5);
11: **end for**
12: initialize loss $L \leftarrow 0$;
13: **for** level-$j$ $\mathcal{G}_i^j \sim \mathcal{G}_i$ **do**
14: accumulate loss using $\mathcal{G}_i^j$ by Eq. (1);
15: **end for**
16: Mini-batch SGD to minimize $L$, update $\Theta^{cnn}$, $\Theta^{att}$;
17: $\tau \leftarrow \tau + 1$;
18: **end while**

### 2.4. Meta-Test: Apply PPN to New Tasks

We study two test settings for weakly-supervised few-shot learning. They differ in if the weakly-supervised information, i.e., the weakly-labeled data and the connections of new classes to the category graph, is still accessible (PPN+) in the test tasks or not (PPN). The PPN setting is more challenging but is preferred in a variety of applications, for example, where the test tasks happen on different devices, whereas the first setting is more appropriate for life-long/continual learning on a single machine. In the PPN setting, we can still leverage the trained prototypes: for an unseen test class $y$, we find the $K$-nearest neighbors of $\boldsymbol{P}_y^0$ among all achieved prototypes, and treat the training classes associated with the $K$-nearest neighbor prototypes as the parents of $y$ on $\mathcal{G}$.

In both settings, for each class $y$ in a test task $T$, we apply the prototype propagation in Eq. (2)-(5) on a subgraph composed of $y$ and its parents $\mathcal{P}_y^{\mathcal{G}}$. This produces the final prototype $\boldsymbol{P}_y$, which will be used in KNN classification on task $T$ as a candidate of nearest neighbor within $\boldsymbol{P}_{\mathcal{Y}^T}$. In the first setting (PPN+), when a parent class $y' \in \mathcal{P}_y^{\mathcal{G}}$ is among the training classes, we use the buffered prototype $\boldsymbol{P}_{y'}^0$ from training for propagation; otherwise, we use

*Table 1.* Validation accuracy (mean±CI%95) on 600 test tasks of PPN/PPN+ and baselines on WS-ImageNet-Pure(-P) and WS-ImageNet-Mix(-M). "W-S" refers to "weakly-Supervised", and "*" marks the modified baselines using the same W-S information as our method.

| Model | W-S | 5way1shot-P | 5way1shot-M | 10way1shot-P | 10way1shot-M |
|---|---|---|---|---|---|
| Prototypical Net (Snell et al., 2017) | N | 33.17±1.65% | 31.93±1.62% | 20.48±0.99% | 21.02±0.97% |
| GNN (Garcia & Bruna, 2018) | N | 30.83±0.66% | 33.60±0.11% | 20.33±0.60% | 22.00±0.89% |
| Closer Look (Chen et al., 2019) | N | 32.27±1.58% | 33.10±1.57% | 22.78±0.94% | 20.85±0.92% |
| Prototypical Network* | Y | 32.13±1.48% | 31.80±1.48% | 20.07±0.93% | 20.33±0.98% |
| GNN* | Y | 32.33±0.52% | 30.33±0.80% | 22.50±0.67% | 23.33±1.03% |
| Closer Look* | Y | 32.63±1.55% | 31.13±1.51% | 20.03±0.83% | 20.25±0.87% |
| PPN (Ours) | Y | 37.37±1.64% | 36.23±1.69% | 24.17±1.00% | 23.30±1.06% |
| PPN+(Ours) | Y | **48.00±1.70%** | **41.60±1.67%** | **35.75±1.13%** | **29.87±1.08%** |

Eq. (2) to compute $\boldsymbol{P}^0_{y'}$ over all weakly-labeled samples belonging to class $y'$. In the second setting (PPN), since all the parents of $y$ have to be training classes, we directly use their buffered prototypes from training for propagation.

## 3. Experiments

We compare PPN/PPN+ to three baseline methods, i.e., Prototypical Networks, GNN and Closer Look (Chen et al., 2019), and their variants of using the same weakly-labeled data as PPN/PPN+. For their variants, we apply the same level-wise training on the same weakly-labeled data as in PPN/PPN+ to the original implementations. The results are reported in Table 1, where the variants of baselines are marked by "*" following the baseline name.

In PPN/PPN+ and all the baseline methods (as well as their variants), we use the same backbone CNN that has been used in most previous few-shot learning works (Snell et al., 2017; Finn et al., 2017; Vinyals et al., 2016). It has 4 convolutional layers, each with 64 filters of $3 \times 3$ convolution, followed by batch normalization (Ioffe & Szegedy, 2015), ReLU nonlinearity, and $2 \times 2$ max-pooling. The transformation $g(\cdot)$ and $h(\cdot)$ in the attention module are linear layers.

In PPN/PPN+, the variance of $\boldsymbol{P}^0_y$ increases when the number of samples per class reduces. Hence, we set $\lambda = 0$ in Eq. (5) for $N$-way 1-shot classifications, and $\lambda = 0.5$ for $N$-way 5-shot classification. During training, we lazily update $\boldsymbol{P}^0_y$ for all the classes on the graph $\mathcal{G}$ every $m = 5$ epochs and choose the nearest $K = 3$ neighbours as parents among all prototypes gained after training for PPN. ADAM (Kingma & Ba, 2015) is used to train the model for 150k iterations, with an initial learning rate of $10^{-3}$. We reduce the learning rate by a factor of $0.7\times$ every 15k iterations starting from the 10k-th iterations.

### 3.1. Results

**WS-ImageNet-Pure** is a subset of ILSVRC-12 with 212 classes and 49,002 images in total. On the ImageNet Word-Net Hierarchy, we extract 80% classes from level-7 as leaf nodes of the category graph $\mathcal{G}$ and use them as the targeted classes $\mathcal{Y}$ in few-shot tasks. The ancestor nodes of these classes on $\mathcal{G}$ are then sampled from level-6 to level-3, which compose weakly-labeled classes $\mathcal{Y}^w$. The number of sampled data points associated with each class reduces exponentially when the level of the class increases. This is consistent with many practical scenarios, i.e., samples with finer-class labels can provide more information about targeted few-shot tasks, but they are much more expensive to obtain and usually insufficient. We divide the classes from level-7 into two disjoint subsets with ratio 4:1 for training and test respectively.

The experimental results of PPN/PPN+ and all the baselines (and their weakly-supervised variants ending with "*") on WS-ImageNet-Pure are shown in Table 1. PPN/PPN+ outperform all other methods. The table shows that PPN/PPN+ are very advantageous in 1-shot tasks, and that PPN+ achieves $\sim 15\%$ improvement compared to other methods. This implies that the weakly-supervised information can be much helpful when supervised data is highly insufficient, and our method is able to significantly boost performance by exploring the weakly-labeled data. Although all the weakly-supervised variants of baselines are trained on the same data as PPN/PPN+, they do not achieve similar improvement because their model structures do not have such mechanisms as the prototype propagation in PPN which relates different classes and tasks. In addition, training on unrelated tasks can be distracting and even detrimental to performance. In contrast, PPN/PPN+ build a computational graph of prototypes with both coarse and fine classes, and the error on any class can be used to update the prototypes of other classes via backpropagation on the computational graph.

**WS-ImageNet-Mix** To verify if PPN can still learn from weakly-labeled data that belong to other fine classes not involved in the few-shot tasks, we propose another subset of ILSVRC-12, WS-ImageNet-Mix with 215 classes and 338,548 images in total. We extract WS-ImageNet-Mix by following the same procedure as extracting WS-ImageNet-

Pure except that data points sampled for a coarse (non-leaf) class can belong to the remaining $\sim 20\%$ level-7 classes outside of the $\sim 80\%$ level-7 classes used for generating few-shot tasks.

The experimental results are reported in Table 1, which shows that PPN/PPN+ still outperform all other methods, and PPN+ outperforms them by $\sim 10\%$ for 1-shot classification. This indicates that PPN/PPN+ is robust to weakly-labeled data unrelated to the final few-shot tasks.

## References


Chen, W.-Y., Liu, Y.-C., Liu, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. *ICLR*, 2019.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Garcia, V. and Bruna, J. Few-shot learning with graph neural networks. *ICLR*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

Zhou, Z.-H. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2017.

Zhu, X. and Ghahramani, Z. Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002.