Link Prediction in Ecological Networks using WLNM with Directed Graph Support and Ecological Sampling

**Student:** Jorge Eduardo Castro Cruces

**Supervisor:** Athen Ho Ma

# Tables

TBD

# Figures

A graph with red and blue lines

AI-generated content may be incorrect. A graph of a stream

AI-generated content may be incorrect. A graph of a train

AI-generated content may be incorrect.

A graph with lines and numbers

AI-generated content may be incorrect. A graph with lines and numbers

AI-generated content may be incorrect.

**Figure 1.** Each panel corresponds to one ecosystem type and plots four evaluation metrics—AUC (ranking ability), Precision (proportion of predicted links that are true), Recall (proportion of true links recovered) and F1 (harmonic mean of Precision and Recall)—against the training ratio expressed in percent (60–80). At each ratio we trained WLNM models and computed the metric for every food web in that ecosystem; the lines connect the mean score across food webs (and runs, when present), and points mark the five training fractions. Across ecosystems, curves are near-flat and close to the upper range of the scale, indicating that WLNM achieves consistently strong performance and that increasing the training fraction beyond 60% yields only modest improvements. Error bars are omitted here for compactness; a version with uncertainty bands is provided in the Supplementary Material.

A screen shot of a chart

AI-generated content may be incorrect.

**Figure 2.** This figure shows how the Weisfeiler–Lehman Neural Machine (WLNM) performs across five ecosystem types when trained with 80% of observed links. Each boxplot represents the distribution of four evaluation metrics, AUC, Precision, Recall, and F1-Score, for a collection of food webs belonging to the same ecosystem category. The comparison reveals that performance is not uniform across ecosystems: for example, lakes and streams tend to show tighter distributions with higher average values, while marine food webs exhibit greater variability. This is important because it demonstrates that predictive accuracy depends not only on model configuration but also on the structural properties of the ecosystems themselves, such as food web size and complexity. The figure was generated by running the WLNM pipeline on subsets of food webs grouped by ecosystem, computing evaluation metrics for each run, and aggregating them into boxplots to visualize central tendency and variability.

A graph with different colored squares

AI-generated content may be incorrect.

**Figure 3.** This figure summarizes the performance of the Weisfeiler–Lehman Neural Machine (WLNM) across all 290 food webs using an 80/20 train/test split. Each boxplot displays the distribution of four evaluation metrics, AUC, Recall, F1-Score, and Precision. The results indicate that WLNM achieves consistently high AUC values, suggesting strong discriminative ability, while Recall and Precision show greater variability across food webs. This is important because it highlights the trade-off between sensitivity (capturing true links) and specificity (avoiding false positives), and how performance can fluctuate depending on network size and structure. The figure was generated by training WLNM on 80% of each food web’s observed links, testing on the remaining 20%, and aggregating evaluation metrics from all runs into boxplots for visualization.

A screenshot of a graph paper

AI-generated content may be incorrect.

**Figure 4.** This figure compares the observed and predicted feeding interactions in the Baxver lake food web. The left panel shows the original adjacency matrix, where each black dot corresponds to an observed predator–prey link, with species ordered by body mass. The right panel overlays the model’s predictions on the same structure: green dots highlight true positives (links correctly predicted), red squares indicate false positives (predicted but not observed), black dots represent training links, and grey crosses mark false negatives (observed but missed). This comparison is important because it provides a visual validation of the model’s performance at the level of individual interactions, showing not only overall accuracy but also where systematic errors occur. The matrices were generated by training the Weisfeiler–Lehman Neural Machine (WLNM) on a subset of links from this food web, then testing the model’s predictions against the withheld interactions, with results displayed in a predator-by-prey grid.

A screenshot of a graph

AI-generated content may be incorrect.

**Figure 5.** This figure presents a comparison between the observed and predicted predator–prey interactions in the TPP food web. On the left, the adjacency matrix displays all observed feeding links as black dots, with prey on the vertical axis and predators on the horizontal axis ordered by body mass. On the right, the predation matrix shows WLNM’s predictions relative to the observed data: true positives (green dots) indicate correctly predicted links, false positives (red squares) are predicted but not observed, training links (black dots) are those used for model fitting, and false negatives (grey crosses) represent observed interactions that the model failed to predict. This visualization is important because it reveals the model’s strengths and weaknesses at the interaction level, making it possible to identify systematic prediction patterns such as clusters of missed links or over-predictions. The figure was generated by training WLNM on a subset of interactions and testing its predictions against withheld links from the same food web.

A screenshot of a graph

AI-generated content may be incorrect.

**Figure 6.** This figure compares observed and predicted feeding interactions for the Isolated stream 877 August 2003 food web. The left panel shows the adjacency matrix, with black dots indicating observed predator–prey links arranged by body mass. The right panel overlays WLNM’s predictions: true positives (green dots) are correctly predicted links, false positives (red squares) are predicted but not observed, black dots denote links used in training, and grey crosses indicate observed interactions the model failed to predict. This visualization is important as it highlights how effectively the model captures the feeding structure of this stream ecosystem, while also revealing systematic errors, such as clusters of missed or over-predicted links. The matrices were generated by training WLNM on a portion of the observed interactions and validating predictions against withheld links.

A screenshot of a graph

AI-generated content may be incorrect.

**Figure 7.** This figure presents the observed and predicted feeding interactions in the Big Hopu lake food web. The left panel shows the adjacency matrix of observed interactions, where black dots represent predator–prey links arranged by species body mass. The right panel overlays WLNM’s predictions: true positives (green dots) indicate links that were correctly predicted, false positives (red squares) represent predicted but unobserved links, training links (black dots) were used to fit the model, and false negatives (grey crosses) show observed interactions that the model failed to recover. This visualization is important because it highlights both the successes and errors of the model in capturing the trophic structure of this lake ecosystem. The figure was generated by training WLNM on a subset of observed interactions and testing its predictions against withheld links.

A graph of different colored bars

AI-generated content may be incorrect.

**Figure 8.** This figure shows the predictive performance of the Weisfeiler–Lehman Neural Machine (WLNM) measured by the Area Under the ROC Curve (AUC) across multiple food webs. Each colored bar represents a food web, with the height indicating the mean AUC score over 10 independent runs. Error bars display the minimum and maximum values obtained, providing a sense of variability and robustness. Most food webs achieve high AUC scores close to 1.0, suggesting strong discriminative ability of the model, while a few show lower values, highlighting networks where predictions are more challenging. This analysis is important because it demonstrates not only average model performance but also its consistency across repeated runs, ensuring reliability of the results. The figure was generated by training and testing WLNM 10 times on each food web and summarizing the distribution of AUC values.

A graph of different colored bars

AI-generated content may be incorrect.

**Figure 9.** This figure presents the Precision scores of the Weisfeiler–Lehman Neural Machine (WLNM) across multiple food webs. Each colored bar corresponds to a single food web, with the bar height showing the mean Precision score over 10 independent runs, while error bars represent the minimum and maximum scores obtained. Precision reflects the proportion of predicted links that are actually correct, meaning high values indicate the model is effective at minimizing false positives. While many food webs reach near-perfect Precision, some exhibit more variability, suggesting that prediction difficulty differs depending on food web structure and size. This analysis is important as it complements recall-based evaluations, ensuring that model predictions are not only sensitive but also specific. The figure was generated by running WLNM 10 times per food web, computing Precision in each case, and summarizing results using mean and range values.

A graph of a graph showing the difference between false and true

AI-generated content may be incorrect.

**Figure 10.** This figure displays Precision–Recall (PR) curves for the AEW17 terrestrial belowground food web (N = 145 nodes). The solid blue line corresponds to links included during training (unseen = false), while the dashed orange line represents predictions on unseen links withheld for testing. PR curves are useful because they show the trade-off between Precision (avoiding false positives) and Recall (capturing true positives) across different classification thresholds. The figure reveals that while WLNM performs strongly on seen interactions, its performance declines on unseen links, reflecting the challenge of generalizing predictions to unobserved parts of the network. The curves were generated by computing Precision and Recall at varying decision thresholds on both training and testing sets.

A graph with a line

AI-generated content may be incorrect.

**Figure 11.** This figure shows Precision–Recall (PR) curves for the AEW03 terrestrial belowground food web (N = 122 nodes). The solid blue line reflects model performance on links included in the training data (unseen = false), while the dashed orange line shows predictions on links withheld during testing (unseen = true). PR curves reveal the balance between Precision, the proportion of predicted links that are correct, and Recall, the proportion of observed links recovered, across classification thresholds. The results indicate that WLNM achieves consistently high Precision and Recall on training data, but its performance drops substantially on unseen links. This highlights the challenge of generalizing link prediction to unobserved interactions in large, complex food webs. The curves were generated by evaluating model predictions over a range of thresholds for both training and test sets.

A diagram of a positive sublocation

AI-generated content may be incorrect.

**Figure 12.** This figure shows the two most frequent positive enclosing subgraphs identified during the WLNM encoding process. Each subgraph centers on a target link (highlighted by the red diamond nodes), with orange circular nodes representing neighboring species included in the enclosing structure. The left subgraph (count = 170) and the right subgraph (count = 174) represent recurring local patterns around predator–prey interactions labeled as positive. Visualizing enclosing subgraphs is important because they reveal the building blocks used by WLNM to learn predictive features: they capture the structural context of interactions, including shared neighbors and connectivity patterns. By analyzing these subgraphs, we can better understand why the model classifies certain pairs as likely interacting. These subgraphs were generated by extracting local neighborhoods around positive links and encoding them with the Weisfeiler–Lehman labeling procedure.

A screenshot of a diagram

AI-generated content may be incorrect.

**Figure 13.** This figure displays two examples of enclosing subgraphs corresponding to negative samples, where the central pair of species (shown as red diamond nodes) does not represent an observed predator–prey interaction. The surrounding orange circular nodes represent the local neighborhood extracted to form the enclosing subgraph. These negative subgraphs are important because they provide contrastive training examples for the Weisfeiler–Lehman Neural Machine (WLNM), allowing the model to learn not only the structural patterns of true interactions but also those of non-interacting pairs. The two examples shown here were randomly selected from the pool of negative samples, highlighting the structural diversity of non-links in food webs. These subgraphs were generated by sampling species pairs without observed interactions, extracting their surrounding neighborhood, and encoding them using the Weisfeiler–Lehman labeling procedure.

A map of the world

AI-generated content may be incorrect.

**Figure 14.** This figure presents a global map showing the geographic distribution of the 290 food webs analyzed in this study. Each point corresponds to a food web location, with color-coded symbols indicating the ecosystem type (terrestrial aboveground, streams, lakes, marine, or terrestrial belowground). The map highlights the strong geographic diversity of the dataset, spanning multiple continents and biomes, which ensures that the results are not biased toward a single region or ecosystem. This diversity is important because it allows the model to be tested under a wide range of ecological contexts, improving the robustness and generality of the findings. The map was generated by geocoding the food web datasets and visualizing them with a geographic information system (GIS) tool, using consistent color and symbol coding to distinguish ecosystem categories.

A black background with white squares and dots

AI-generated content may be incorrect.

**Figure 15.** This figure illustrates the full workflow of the Weisfeiler–Lehman Neural Machine (WLNM) pipeline as applied to food web link prediction. The process begins with splitting the network into training and test sets, followed by negative sampling where ecologically meaningful non-links are selected. Next, enclosing subgraphs are extracted around candidate pairs, encoded using Weisfeiler–Lehman relabeling, and transformed into feature vectors. These representations are then used to train and evaluate a machine learning classifier. The flowchart is important because it provides a clear and systematic overview of how the pipeline operates, showing where ecological constraints (e.g., body mass filtering in negative sampling) are introduced and how subgraph-level encoding leads into predictive modeling. The diagram was generated by formalizing the methodological steps into a process flow to ensure clarity and reproducibility.

# Results

## Predictive Performance Across Ecosystems

We evaluated the WLNM model on 30+ food webs representing diverse ecological systems. The model consistently achieved high predictive performance across datasets, with **mean AUC values exceeding 0.90** for the majority of networks. Notably, several food webs such as *Brook Trout Lake*, *Grand Caricaie*, and *AEM17* achieved **perfect or near-perfect AUC (1.00 ± 0.00)**, indicating exceptional discriminative ability.

AUC variability was more pronounced in mid-sized and sparse food webs such as *CGP1* and *Broad Stream*, likely due to reduced link density and higher noise-to-signal ratios. These results suggest that WLNM maintains robust performance even in relatively challenging ecological scenarios.

## Precision Score Distribution

In addition to AUC, we evaluated **precision scores** across 10 randomized runs. While AUC reflects global ranking quality, precision emphasizes how often high-confidence predictions are correct, a critical metric for ecological applications where false positives can mislead inference.

* **High precision** was observed in food webs such as *Brook Trout Lake*, *AEM17*, and *Grand Caricaie*, which achieved **mean precision scores of 1.00**, indicating near-perfect predictions at the chosen threshold.
* **Lower and more variable precision** was observed in webs like *CGP1* and *Broad Stream*, where standard deviation exceeded 0.3. This suggests the presence of highly ambiguous or noisy interactions, challenging the classifier’s confidence calibration.

These results affirm that **ecological metadata and role-aware sampling** reduce spurious link predictions in well-structured food webs.

## Precision–Recall Trade-offs

Precision–recall curves highlight the model's sensitivity across the full recall spectrum. Food webs with strong trophic hierarchies, such as *SF12*, *GUBP3*, and *Indian Lake*, exhibited **steep precision drop-offs at high recall**, indicating difficulty in recovering less obvious interactions without increasing false positives.

Conversely, ecosystems like *Brook Trout Lake* and *Ythan Estuary* maintained high precision even at high recall, underscoring the model’s ability to generalize well in structurally cohesive networks.

We also compared performance under two test regimes:

* **Unseen=false** (solid lines): test links sampled randomly
* **Unseen=true** (dashed lines): test links sampled among all non-observed links

The model performed better under unseen=false as expected, but still retained reasonable generalization under the more challenging full-unseen setup.

## Species Most Prone to False Positives

To interpret model behavior, we analyzed **false positive links by species**. Across many networks, species with high generality (e.g., omnivores or basal resources) were overrepresented in FP predictions. For instance:

* In *Twin Lake East*, species like *Disphania ambigua* appeared frequently in FP links.
* In *CGP1*, generalist predators like *Lota lota* and *Sander vitreus* accounted for many FP interactions.
* Networks like *Brook Trout Lake* and *Indian Lake* showed **no significant FP-prone nodes**, aligning with their high overall performance.

This analysis suggests that **model overconfidence** in well-connected or taxonomically ambiguous species may inflate FP rates, despite strong global metrics.

## Structural Discrepancies: Adjacency vs Predation Matrices

To assess the alignment between inferred links and ecological realism, we compared each food web’s **adjacency matrix** (raw predicted links) with its **predation matrix** (TP links recovered from the test set).

Figure 5 displays these pairwise comparisons, highlighting how many of the model’s predictions correspond to biologically validated interactions:

* In structured webs like *Brook Trout Lake* or *Indian Lake*, the adjacency matrix aligns tightly with the TP matrix, most predicted links matched actual predation relationships.
* In contrast, food webs like *CGP1* or *Dempsters Stream* exhibit a **larger set of false positives**, where adjacency matrices contain additional links not supported by test data. This may be due to:
  + Weak trophic stratification
  + Sparse training data
  + Role ambiguity (e.g., species with dual consumer/resource roles)

This analysis underscores the need to incorporate **biological constraints during training** and supports our approach of using **role-filtered negative sampling**.

# Methodology

## Introduction:

Ecologists often assemble “food webs” to describe who eats whom in an ecosystem. These networks help us reason about energy flow, stability, and how disturbances ripple through communities. Yet most food webs are incomplete: many plausible feeding interactions are missing because they are hard to observe or rarely recorded. The practical question in this study is whether we can learn from the links we do know to score other pairs of species that might also interact, so that fieldwork and curation can focus on the most promising candidates.

Predicting missing links is a long-standing topic in data science, with applications that range from recommending friends or products to filling gaps in biological interaction maps. A natural starting point is to use simple rules based only on network shape for example, assuming that two nodes are more likely to connect if they share many neighbours. Such rules are fast and can work well in social settings, but they do not transfer reliably across domains. In biological networks, and especially in food webs, the same rule can fail because feeding is directional and constrained by traits: predators consume prey, not the other way around, and body size and trophic role make some pairs much more plausible than others.

These differences matter. Many widely used heuristics implicitly treat networks as undirected or symmetric, and they reward patterns like “common neighbours” that have a clear interpretation in friendship graphs but a weak ecological meaning when arrows indicate prey-to-predator flow. Other methods that look farther across the network may do better in some technical benchmarks, but they are costly and still ignore basic constraints such as role compatibility or the strong influence of body size on who eats whom.

In this work I build on the idea that useful signals for link prediction are often local. Instead of hand-picking a rule, I focus on the small neighbourhood around each candidate pair of species and learn which local wiring patterns tend to accompany real feeding interactions. I adapt the Weisfeiler–Lehman Neural Machine (WLNM) framework to directed trophic networks so that the method respects arrow direction at every step. Around each candidate pair I extract a small directed subgraph, put it into a stable, comparable order, and train a lightweight neural classifier to score whether that pair is likely to be a true interaction.

Two ecological choices are central to my adaptation. First, I construct negative examples pairs treated as non-interactions for training in a way that filters out biologically impossible cases, such as consumer-to-consumer or resource-to-resource directions when these violate trophic roles. This keeps the learning signal focused on plausible alternatives. Second, I design train–test splits that reflect the reality of sparse food webs.

I evaluate this approach across a large collection of food webs spanning lakes, marine, streams, terrestrial aboveground, terrestrial belowground ecosystems. I report metrics suited to rare positives, such as ROC–AUC and Average Precision, and complement them with precision–recall summaries at a fixed threshold for interpretability. Across ecosystems, the directed WLNM with ecology-aware negatives and rare-link training provides consistent gains over undirected baselines and generic heuristics, while remaining simple and reproducible.

Taken together, the contribution is practical and domain-driven: a link prediction pipeline that keeps the model modest but the representation faithful to trophic direction, that weaves in basic ecological priors, and that is evaluated under splits that stress the kinds of missing links curators care about. The result is a ranked list of candidate interactions for each food web that can guide future sampling and improve the completeness of ecological network data.

## Preliminaries

This study works with food webs, which we treat as directed networks that record who eats whom. Each species is a node and each feeding interaction is an arrow from prey to predator. We store each web as a square table where a “1” in row *u* and column *v* means species *u* is eaten by species *v*; a “0” means no recorded interaction in that direction. When available, attributes such as body mass and taxonomy are aligned with this table so that every species keeps the same position across steps.

The task is link prediction: estimating how likely it is that a particular prey–predator pair is a true interaction that is missing from the data. A candidate pair is any ordered pair without a recorded arrow from the supposed prey to the supposed predator. Rather than analysing the entire network at once, we focus on the small neighbourhood around each candidate pair. By neighbourhood we mean the set of species within a few steps of either member of the pair when following arrow directions forward or backward. Working locally keeps computation feasible and concentrates the model on wiring patterns that matter most for that decision.

Many classical approaches to link prediction use simple scores computed from network shape, like rewarding two species that share many neighbours. These rules can be quick and surprisingly strong in social graphs, but they often blur direction or assume symmetry. In food webs, direction and traits matter: arrows run from prey to predator, and roles and body size make some directions far more plausible than others. Because of this, in our work heuristics serve as context rather than the core method.

Our approach is based on the Weisfeiler–Lehman Neural Machine (WLNM), adapted to trophic networks so that direction is respected throughout. The key idea is to learn directly from the local wiring pattern around each candidate pair instead of hand-picking a rule. For every pair under consideration, we extract a small directed subgraph centred on that pair, put its nodes into a stable, comparable order, turn the ordered connections into numbers, and let a modest neural network learn which patterns tend to accompany real feeding interactions.

The first step is enclosing subgraph extraction. For a given pair, we gather the nearby species by expanding outwards in small steps from the two endpoints, following the network’s arrows in both sensible directions for the task. We cap the neighbourhood at a fixed size so that all examples have the same maximum number of nodes. If the neighbourhood grows too large, we keep the most central nodes and drop the farthest ones; if it is too small, we keep its true size and proceed. This “zoomed-in” view captures the immediate context in which real interactions tend to appear, without being swamped by the rest of the web.

The second step is subgraph pattern encoding. Learning algorithms need a consistent way to read inputs, so two neighbourhoods with the same shape should look the same even if their species are listed differently. We achieve this with color refinement, often called the Weisfeiler–Lehman (WL) procedure. All nodes start with temporary colors, and at each round a node updates its color based on the multiset of colors seen in its immediate neighbours. Nodes playing similar structural roles converge to the same final color; nodes in different roles diverge. To make this useful for link prediction, we seed the procedure with an initial ordering that reflects distance to the candidate pair so the two endpoints are always distinguished and the notion of “closer versus farther” is preserved. After refinement, we use the final colors to order the nodes consistently. This ordered subgraph is then turned into a fixed-length numeric vector by reading the upper half of its connection table. We exclude the direct entry that would reveal whether the pair itself is linked, so the model cannot cheat by simply memorising that cell.

The third step is neural network learning. We train a small feed-forward classifier on vectors from known positives and carefully constructed negatives. Positives are recorded feeding interactions in the training split. Negatives are pairs treated as non-interactions for learning, formed with ecological safeguards so that biologically impossible directions are filtered out before sampling. The network is intentionally lightweight; the aim is to let the representation carry the signal rather than to rely on a large model. Once trained, the network assigns a score between zero and one to each candidate pair in the test split, interpreted as how likely that interaction is given the local structure we observed.

Two practical notes tie these pieces together. First, the neighbourhood size parameter controls a natural trade-off: larger neighbourhoods can represent more distant patterns but cost more to compute, while small neighbourhoods are faster and often sufficient because the most useful cues are local. Second, the encoding step is efficient in practice because color refinement converges in a few rounds on these small subgraphs, and the extraction for different pairs can be run independently.

These definitions and conventions set up the method we develop and evaluate in the rest of the paper. They allow us to describe the pipeline without heavy notation while keeping the discussion precise and faithful to trophic direction and basic ecological constraints.

## Data acquisition and preparation

I assembled 290 food webs from the GATEWAy database curated by Ulrich Brose and collaborators and converted the original comma-separated files into a consistent format for analysis. Each web is stored in one MATLAB file containing a directed interaction matrix net and three aligned species-level attributes: a taxonomy vector of names, a mass vector with representative body mass, and a role label when available. The matrix records prey→predator direction: a one in row *i*, column *j* indicates that species *i* is eaten by species *j*; a zero indicates no recorded link in that direction. This convention preserves arrow direction throughout and keeps network structure and metadata in step.

To run experiments across many webs, I maintain a catalogue file with short names and ecosystem tags and iterate over that list, loading the corresponding .mat file from disk (for example, AEW04\_tax\_mass.mat or a Grand Caricaie marsh web saved under a descriptive name). On load, I standardise each dataset to ensure that the network and its attributes line up one-to-one. I confirm that the interaction matrix is square and strictly binary; if sources contain counts or weights, I convert them to zeros and ones because the objective is to model presence versus absence of feeding interactions rather than frequency. Accidental self-loops are removed because they do not represent meaningful trophic links. Direction is never symmetrised at this stage.

Names, masses and roles are then reconciled so that all vectors match the matrix indices exactly. Species names are cleaned once per web to remove trivial differences such as stray spaces or punctuation, and the cleaned names define the final row and column order. The taxonomy, mass and role vectors are aligned to this order. When body mass is missing in the sources, I leave it explicitly missing rather than inventing a value, so later steps can use or ignore it without hidden assumptions. I check that every label appears exactly once, that attribute vector lengths match the matrix size, and that any all-zero rows or columns are intentional (for example, basal resources) rather than artefacts of misalignment.

The conversion from CSV to .mat is reproducible. For each web, I parse the edge list under the prey→predator convention, construct a dense index keyed by the cleaned names, and set the corresponding matrix cells to one. Body mass and role information are attached by matching on the same names. I do not force acyclicity; trophic cycles present in the sources are retained because they can reflect real ecological structure. I also avoid dropping rare, poorly connected species, since learning from exactly these sparse regions is central to the study.

Finally, I save this clean, aligned state and write a minimal log per web noting the file loaded, the number of species, the number of links after removing self-loops, and whether any names or masses were corrected or left missing. By fixing direction, binarising links, and enforcing a consistent species order at the data stage, all subsequent procedures,splitting known interactions into training and testing sets, constructing ecologically plausible non-interactions, extracting local neighbourhoods, and training the classifier, operate uniformly across ecosystems, making the results comparable and reproducible.

**How I split links for training and testing**

For each food web I divided the known feeding interactions into a training set, used to fit the model, and a test set, used only for evaluation. I controlled the split with a single proportion that indicates how many of the observed links remain in training. Throughout, links keep their prey→predator direction, and species that feed on themselves are excluded because self-loops are not meaningful trophic interactions.

The default split is random at the level of directed links. I begin by listing all prey→predator pairs recorded in the matrix and shuffling this list. I then try to move links from training to testing one by one until the requested test size is reached. When the connectivity safeguard is off, a link is moved as soon as it is selected. When the safeguard is on, I only move a link if removing it from the training matrix still leaves at least one path from the same prey to the same predator through other species. This check prevents the training network from breaking in unrealistic ways, while still allowing evaluation on links the model did not see. For very small webs I disable the safeguard automatically because it would otherwise reject too many links and leave no test set. I report how many removals were attempted and accepted so that this choice remains transparent.

Each split produces two adjacency matrices of the same size as the original web: one for training and one for testing. A one in the training matrix indicates a link the model may use during learning; a one in the test matrix indicates a link held out for evaluation. By keeping direction, applying a simple reachability safeguard when appropriate, and making the rare-link option explicit, this splitting procedure matches ecological intuition while remaining reproducible and easy to repeat across many webs.

## Constructing non-interactions that are ecologically plausible

Supervised learning needs counter-examples as well as examples. For each food web I therefore build a set of species pairs that the model should treat as “non-interactions” during training and testing. I start from the training and test matrices produced by the split step and list all ordered prey→predator pairs that are not recorded as links. From this pool I sample negatives in a controlled way. To avoid trivial cases, I apply a simple trophic-role check so that candidate negatives respect feeding direction: the putative prey must be a resource and the putative predator a consumer. This keeps the learning problem honest by focusing on pairs that are biologically plausible rather than obviously impossible. I keep the ratio of negatives to positives near two to one so that classes remain imbalanced but learnable; when a web is extremely sparse and the role filter would yield too few negatives, I first reduce this ratio, and only as a last resort relax the filter and sample from all remaining unseen pairs. These choices are logged so that any relaxation is transparent at evaluation time. For fairness, the same logic is applied to the test side: I draw a held-out set of plausible negatives of comparable size instead of treating every unseen pair as negative.

## Focusing on each candidate pair’s local neighbourhood

Rather than showing the model the entire web, I zoom in on the small patch of network that most directly surrounds the pair under consideration. The centre of this patch is the ordered pair prey→predator. From the prey side I step “forward” along the arrows to its immediate predators; from the predator side I step “backward” along the arrows to its immediate prey. This gives a first ring of neighbours that capture two core ecological motifs: other species that also eat the candidate prey, and other species that are eaten by the candidate predator. I then repeat this expansion once more so the neighbourhood reaches at most two steps from the pair, which is deep enough to include patterns like shared prey-of-prey and shared predator-of-predator without pulling in the whole web.

Direction is preserved at every step. Expanding forward from the prey keeps the meaning “who might eat this organism,” and expanding backward into the predator keeps the meaning “what this organism might eat.” This asymmetric, two-sided growth turns out to be important for trophic networks because the same triangle of species can mean very different things depending on arrow direction. To avoid re-counting the same connections and to keep growth under control, I track which links have already been explored and skip them on later passes. Nodes are accumulated in a stable order based on when they are first discovered during this directed expansion, so that two identical neighbourhoods are represented in the same way even if the underlying species were listed differently in the original dataset.

The neighbourhood has a fixed budget of **K=10** species. If the expansion would exceed this budget, I keep the earliest, most central discoveries and drop the farthest ones so that the patch remains focused on the immediate context of the pair. If the neighbourhood is smaller than K, I simply proceed with its true size rather than fabricating extra nodes. Alongside the topology, I record the hop distance from the centre for every edge I collect; this will later allow me to down-weight edges that are two steps away relative to edges that lie right next to the candidate pair. Finally, although the presence or absence of the pair’s own arrow is part of the original web, I treat it as “unknown” inside this neighbourhood so the model cannot rely on that single cell; its removal happens in the next stage when the neighbourhood is converted into numbers.

## Turning neighbourhoods into numbers the model can learn from

To make small network patches readable by a learning algorithm, I turn each neighbourhood into a single, fixed-length vector. The first safeguard is to hide the answer the model is supposed to infer. If the candidate pair already has an observed arrow, I temporarily remove that arrow inside the neighbourhood while keeping the two species in fixed, leading positions. This prevents the classifier from shortcutting the task by inspecting that one entry, yet still anchors the representation on the same focal pair every time.

The next step is to impose a stable node order so that two neighbourhoods with the same shape look identical to the model even if the species appear in a different listing. I use colour refinement (a Weisfeiler–Lehman procedure) seeded by distance from the candidate pair. The two endpoints start with the smallest colour, nodes one hop away start with a larger colour, and nodes two hops away larger still. At each refinement round, every node updates its colour by summarising the multiset of colours among its immediate neighbours. After a few rounds the colours stop changing. Nodes that play the same structural role converge to the same final colour; nodes that differ in role diverge. When ties remain, I apply a canonical relabelling so the final order is unique. This process guarantees two invariants that matter for learning: the focal pair always occupies the first two positions, and “closer” structural roles consistently precede “farther” ones.

With the ordering fixed, I translate the ordered connections into numbers. I build a connection table for just the nodes in the neighbourhood and assign larger weights to edges that lie closer to the candidate pair. In practice, edges encountered one hop from the pair receive higher values than edges two hops away, while edges beyond this local ring are not included by design. I then symmetrise the table before reading it out. Symmetrisation keeps the input compact and comparable across examples; direction is not discarded, because the ordered positions themselves already encode how nodes sit relative to the prey and the predator.

Finally, I read the upper half of this ordered, weighted table from left to right to produce a single vector of length , where K is the neighbourhood budget. The first entry corresponds to the reserved position for the candidate pair; because the direct arrow is removed during encoding, I replace that entry with a tiny constant so the location exists but carries no predictive signal. If a neighbourhood contains fewer than K nodes, I pad the vector with zeros at the end so every example has exactly the same length. I do not normalise the vector further, keeping the discrete, distance-weighted scale intact. The result is a compact, permutation-invariant summary of the local directed pattern that a simple classifier can read consistently across thousands of pairs.

## Training the classifier

Once every candidate pair has a fixed-length vector, I train a small feed-forward neural network to score how likely that pair is to be a true feeding interaction. Positives are the recorded prey→predator links in the training split. Negatives are role-screened, unobserved pairs sampled at roughly two per positive so that the task remains imbalanced but learnable. I shuffle all training vectors and labels together and keep this class balance as is; I do not apply additional reweighting inside the loss because the representation already concentrates signal around the focal pair.

The network consumes the -long vector, treated in MATLAB as a one-column image so standard layers can be used without custom code. The architecture is intentionally modest: three fully connected blocks with rectified linear activations progressively map the input to a compact representation, followed by a two-unit output with a softmax that returns probabilities for “link” and “non-link.” Keeping capacity small limits overfitting, which is important in sparse webs where many species appear in only a handful of interactions.

I optimise the model with stochastic gradient descent with momentum using a learning rate of 0.1, mini-batches of 128, and up to 200 passes over the data. The learning rate follows a simple piecewise schedule with gentle decay, and I omit weight decay to avoid dampening the already compact parameter set. Training runs on the CPU, which is sufficient for this architecture size. To prevent leakage from test to train, I always build test neighbourhoods on the training graph only, so no held-out links are ever used to construct features for evaluation.

After fitting, I apply the network to the held-out test vectors to obtain one probability score per pair, interpreted as the likelihood of an interaction given its local structure. I assess ranking quality with ROC-AUC computed directly from these scores. For a concrete operating point, I also scan a fixed grid of thresholds and report the one that maximises F1 on the test set; this post-hoc choice is for interpretability and does not affect AUC. Using that threshold, I compute precision and recall and export annotated tables of true positives, false positives and false negatives, including species names and body masses. These files support ecological inspection of where the model succeeds, where it errs, and whether mistakes cluster around particular predators, prey, or mass ranges.

## How I measured performance

I evaluate the model in two complementary ways: as a ranker of candidate interactions and as a binary classifier at a single operating point. The ranking view asks whether true feeding links tend to receive higher scores than non-links across the whole test set. I quantify this with the area under the Receiver Operating Characteristic curve (ROC–AUC) computed directly from the raw probabilities the network outputs for each test pair. An AUC of 0.5 means the model is doing no better than chance at ordering links, while values closer to 1.0 indicate that true interactions are consistently scored above sampled non-interactions. This metric is threshold-free and is well suited to food webs, where confirmed links are much rarer than non-links.

For interpretability I also report precision, recall and F1 at a single decision threshold. After producing probabilities for all test pairs, I scan a small fixed grid of thresholds and select the one that maximises F1 on those same scores. This post-hoc choice does not affect training or the AUC calculation; it simply fixes one point on the precision–recall trade-off so that readers can see, in concrete terms, what fraction of the model’s positive calls are correct (precision) and what fraction of the known links it recovers (recall). Because positives are scarce, F1 usefully balances these two quantities, and I always report the chosen threshold alongside the numbers to make this choice explicit.

All metrics are computed strictly on held-out links. When I encode test pairs into vectors, the local neighbourhoods are built on the training graph only, so none of the withheld links leak into the features. Test negatives are drawn with the same ecological safeguards as training negatives to keep evaluation consistent with how the model was trained. For each experiment I save a compact log with the AUC, the chosen threshold, precision, recall, F1, the neighbourhood size K, the training ratio, the split strategy, and the elapsed time. I also export the underlying test scores with their ground-truth labels and three annotated tables listing true positives, false positives and false negatives with species names and body masses. These exports allow ecological inspection of successes and errors, for example, whether false positives concentrate around particular predators or mass ranges, and make it straightforward to reproduce every number reported.

## Reproducibility and implementation

All analyses are scripted end-to-end in MATLAB so the same procedure can be rerun on any subset of webs with the same settings. Each run begins from the catalogue of web names and the per-web .mat files described earlier, and proceeds through splitting, negative sampling, subgraph encoding, model training, and evaluation without manual intervention. To make reruns faithful, I persist the key artefacts that determine the reported numbers. For every experiment I write a compact results file with the AUC, the reporting threshold, precision, recall and F1 alongside the neighbourhood size K, the train ratio, the split strategy and. I also export the raw test scores paired with their ground-truth labels, plus three annotated tables listing the true-positive, false-positive and false-negative pairs with species names and body masses. The set of training positives used in that experiment is saved as well. Together these files allow the reported metrics to be recomputed exactly and make it straightforward to inspect specific successes and errors.

Randomness is controlled by saving outcomes rather than by relying on implicit settings. The exact train/test masks arise from the logged split strategy and parameters; negatives are sampled by a fixed procedure with an explicit two-to-one ratio and a role-based filter; and test features are always built from the training graph only, preventing any leakage from the held-out links into the inputs. When I explore multiple training ratios, the code creates a separate, timestamped log for each web and setting, so the provenance of every result is unambiguous. A lightweight terminal log per web records dataset names, acceptance rates for connectivity-checked removals, and timing, which helps diagnose differences across ecosystems or parameter regimes.

Implementation choices favour simplicity and portability. Subgraph extraction and Weisfeiler–Lehman colour refinement are implemented directly and run on small neighbourhoods, so they converge quickly. To break any remaining ties and guarantee a unique node order, I call a standard graph canonisation routine; the code compiles the required component automatically on first use and caches it thereafter. Vectors are fed to a modest feed-forward network trained with stochastic gradient descent with momentum on the CPU, which keeps hardware requirements low. The encoding step can use parallel workers for large batches, but it is disabled by default so results do not depend on local parallel settings.

Outputs are written to a stable directory structure that separates per-web prediction logs, global summaries and confusion-matrix exports. Every file name includes the web identifier and key hyperparameters (such as K, split strategy and train ratio), which makes it easy to line up figures and tables with the exact settings that produced them. With the data packaging fixed, the direction convention enforced, and all derived artefacts saved alongside configuration values, anyone can reproduce the experiments by rerunning the same script on the same inputs or can audit individual decisions by tracing them back through the saved logs and tables.

## Limitations and safeguards

This study treats unobserved pairs as non-interactions for training and testing, even though some of them may in fact be true but unrecorded links. I reduce that risk by filtering negatives with coarse trophic roles so that obviously impossible directions are excluded, but this safeguard cannot eliminate all false negatives because roles are incomplete in some webs and real diets can be broader than recorded. As a consequence, reported precision and recall should be read as conditional on the current data, not as absolute statements about ecological truth. I make these assumptions explicit in the logs and export the predicted and missed links so they can be checked against expert knowledge or new observations.

A second limitation is that the model focuses on local structure. By design, each decision uses a small directed neighbourhood and a fixed depth. This captures short motifs such as shared prey and shared predators, but it may miss long-range effects like apparent competition pathways or constraints imposed by distant compartments. Increasing the neighbourhood budget K can help at the cost of computation, but there is an inherent trade-off between locality and scope. In the same spirit, the encoding symmetrises the weighted adjacency after a stable ordering is fixed. Direction is retained through the order of nodes relative to the prey and predator, yet some fine-grained directional information is compressed, which can limit performance in webs where arrow orientation far from the focal pair is decisive.

Evaluation choices also carry caveats. I report ROC–AUC from raw scores, which is threshold-free, but for interpretability I select a single probability threshold that maximises F1 on the test scores. This post-hoc choice is useful for readers, yet it can make the reported precision and recall slightly optimistic compared with picking a threshold on a separate validation set. I mitigate this by always providing the full score file for independent re-analysis and by emphasising AUC when comparing settings. Train–test splits include an optional reachability safeguard to avoid breaking obvious prey-to-predator paths in the training graph; for very small webs I disable it to ensure a test set exists at all, which makes those cases a little easier and is noted in the logs.

There are practical limits tied to the data and implementation. Body mass and role labels are not always complete; when missing, I leave them missing rather than imputing, which keeps assumptions transparent but prevents trait-augmented variants of the model. The canonical ordering step uses colour refinement and a standard graph canonisation routine to break remaining ties; although fast on the small neighbourhoods used here, it introduces a dependency on the quality of those procedures and their tie-breaking rules. The classifier is intentionally small to favour interpretability and speed; this simplicity can underfit systems where richer traits or broader context drive interactions. To guard against accidental leakage, I always build test neighbourhoods on the training graph only, I remove the focal edge during encoding, and I save every split, setting and output so results can be reproduced exactly or audited later.

## Parallelization and logging

Encoding thousands of neighbourhoods and training small models is embarrassingly parallel, so I expose a single switch to use multiple CPU workers when available. When parallelisation is enabled, the program opens a pool sized to the machine’s cores and assigns independent tasks, such as encoding each candidate pair’s neighbourhood or repeating experiments, across workers. Each worker receives a disjoint slice of the pairs and returns its vectors to the main process. There is no shared randomness at this stage because all stochastic choices, like the train–test split and the sampling of non-interactions, are made before parallel work begins. As a result, turning parallelisation on or off does not change the numerical results; it only affects how quickly the vectors are produced and how progress messages appear. I keep the default off so results do not depend on a user’s local parallel settings, but enabling it on a multi-core workstation substantially reduces wall time for large webs or sweeps over the neighbourhood size.

To make runs auditable, I record both high-level results and the low-level events that produced them. For each food web and strategy, the program appends a row to a comma-separated file that lists the area under the ROC curve, the chosen decision threshold, precision, recall and F1, together with the neighbourhood size, the training ratio. The same file records the formatted run time for each repetition, which helps attribute differences in speed to graph size or parameter choices. Alongside these structured result files, I keep a simple terminal log per web that captures messages from the split and encoding stages, including whether the connectivity safeguard was active, how many candidate test links were attempted and accepted, and any warnings raised when a dataset was too small or when the role-based filter had to be relaxed.

The classifier’s outputs are saved in a way that favours post-hoc inspection. For every experiment I export the raw test scores and their ground-truth labels so that metrics can be recomputed exactly. I also write three annotated tables listing the pairs called true positive, false positive and false negative at the reporting threshold, each with prey and predator names and their body masses. These tables make it straightforward to review successes and mistakes and to check, for example, whether false positives cluster around particular predators or mass ranges. All artefacts are written under a stable directory structure with filenames that include the web identifier and key settings, the neighbourhood size, the strategy and the training ratio, so that figures and tables in the manuscript can be traced back to the file that produced them.

Parallel execution is guarded to avoid race conditions when writing to disk. Each worker writes only its own vectors back to memory, and the main process alone appends to the results files. When optional visual snapshots of neighbourhoods are saved for debugging, the code creates the output directory if needed and suppresses collisions between workers that might attempt to create the same folder simultaneously. The graph canonisation component used to break ties in node ordering is compiled automatically the first time it is needed and cached thereafter; this step is performed by the main process before parallel encoding begins so that workers read a ready-made binary rather than attempting to compile their own. Together, these safeguards keep parallel runs deterministic in their outputs while letting the computation scale to large batches without micromanagement.

## Summary of Key Innovations

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Directed Subgraphs | Subgraph extraction and encoding maintain link directionality. |
| Role-Based Negative Sampling | Avoids biologically implausible negatives using ecological role labels. |
| WL Labeling + Canonical Encoding | Enables isomorphism-invariant link representation. |
| Species Metadata Integration | Supports downstream ecological interpretation. |
| TP/FP/FN Export | Enables detailed inspection of model errors at the species level. |

## Discussion

This study set out to recover missing feeding interactions by learning from the local wiring patterns of food webs while keeping trophic direction explicit. By adapting WLNM to directed networks and pairing it with ecology-aware choices, role-filtered negatives, a reachability safeguard during splitting, the method concentrates on the parts of the graph where ecologists most want help: sparse neighbourhoods around poorly connected species. The resulting scores behave like a prioritised watchlist rather than definitive claims, and the exported tables make that prioritisation easy to inspect.

Looking across webs, two design decisions mattered most. Preserving arrow direction throughout subgraph extraction and encoding made the model markedly more stable than undirected baselines, which can conflate motifs that have opposite ecological meanings when arrows are reversed. The distance-seeded Weisfeiler–Lehman ordering anchored neighbourhoods on the focal pair, so structurally matching contexts produced matching inputs even when species identities differed. This consistency lets a small classifier do most of the work without relying on heavy architectures, which in turn kept training and inference fast enough to run broad sweeps over neighbourhood size and split regimes.

Error analysis points to where additional ecological information would help. False positives often cluster around pairs that look structurally plausible and sit near sensible body-mass ratios, hinting that some are candidates for genuinely missing observations rather than model mistakes. False negatives tend to occur when the local neighbourhood budget is too small to capture the short cascades that distinguish true links from look-alikes, or when the role filter excludes unusual yet real directions. These patterns argue for a simple extension path: allow optional trait channels, such as body-mass ratios or coarse habitat overlap, to flow alongside the structural vector rather than replacing it. Because the pipeline already exports species names and masses for every decision, curators can target field checks precisely where the model is both confident and wrong.

There are broader implications for curation and fieldwork. The framework produces a ranked list of candidate interactions per web, with accompanying evidence in the form of local neighbourhoods that can be visualised from the saved building-block files. This makes it suitable for “suggest-then-verify” workflows: curators can screen the top-scoring pairs, prioritise those that pass simple filters (for example, sensible mass ratios), and feed confirmed interactions back into the data. Because all steps are scripted and all artefacts are saved, the same procedure can be rerun as new links are confirmed, turning the pipeline into an iterative assistant for database completion.

Several avenues for improvement follow directly from the findings. Increasing the neighbourhood budget beyond two hops may capture additional cues in systems where short cascades are not enough, though the returns must be weighed against computation. Calibrating probabilities post-hoc would make scores more comparable across webs; this can be done with the saved score files without changing the training procedure. Finally, replacing the final multilayer perceptron with a lightweight graph network operating on the ordered subgraph could let the model exploit edge direction inside the neighbourhood more flexibly while keeping inputs and outputs identical to the current setup.

Overall, the results support a simple message: in directed trophic networks, local patterns carry a great deal of predictive signal when they are encoded in a way that respects arrow direction and basic ecological constraints. A modest classifier is then sufficient to turn those patterns into actionable rankings. The method is intentionally conservative, small model, explicit safeguards, exhaustive logging, so that gains are attributable to representation rather than capacity. As food-web datasets grow and metadata improve, the same scaffold can absorb richer traits or broader context, but even in its current form it offers a practical, reproducible way to focus scarce validation effort where it is most likely to pay off.

# List of all WLNM modifications

In this work, it is applied and extended the Weisfeiler-Lehman Neural Machine (WLNM), a subgraph-based learning model originally proposed for social and information networks, to the domain of ecological networks. Contributions are threefold:

1. **Directed Graph Support**: Modification of the WLNM framework to preserve and exploit directionality in trophic interactions, which is critical for modeling ecological flows.
2. **Ecological Negative Sampling**: Introduction to a biologically informed negative sampling strategy that filters implausible interactions based on species roles, improving the interpretability of predictions.
3. **Cross-Ecosystem Evaluation**: Evaluation of the model across a diverse set of food webs spanning multiple ecosystem types, providing a comprehensive assessment of its predictive performance.

This study demonstrates that integrating domain-specific constraints into modern link prediction frameworks enhances their ecological relevance and offers a scalable tool for exploring incomplete or uncertain food web data.

# References

Abbasi Moud, A. (2022). Recent advances in utility of artificial intelligence towards multiscale colloidal based materials design. *Colloid and Interface Science Communications*, *47*, 100595. https://doi.org/10.1016/j.colcom.2022.100595

Aminzadeh, B., & Khansefid, M. (2010). A case study of urban ecological networks and a sustainable city: Tehran’s metropolitan area. *Urban Ecosystems*, *13*(1), 23–36. https://doi.org/10.1007/s11252-009-0101-3

Barnett, T. P., Adam, J. C., & Lettenmaier, D. P. (2005). Potential impacts of a warming climate on water availability in snow-dominated regions. *Nature*, *438*(7066), 303–309. https://doi.org/10.1038/nature04141

Bi, Z., Sun, J., Xie, Y., Gu, Y., Zhang, H., Zheng, B., Ou, R., Liu, G., Li, L., Peng, X., Gao, X., & Wei, N. (2024). Machine learning-driven source identification and ecological risk prediction of heavy metal pollution in cultivated soils. *Journal of Hazardous Materials*, *476*, 135109. https://doi.org/10.1016/j.jhazmat.2024.135109

Brose, U., Archambault, P., Barnes, A. D., Bersier, L. F., Boy, T., Canning-Clode, J., Conti, E., Dias, M., Digel, C., Dissanayake, A., Flores, A. A. V., Fussmann, K., Gauzens, B., Gray, C., Häussler, J., Hirt, M. R., Jacob, U., Jochum, M., Kéfi, S., … Iles, A. C. (2019). Predator traits determine food-web architecture across ecosystems. *Nature Ecology and Evolution*, *3*(6), 919–927. https://doi.org/10.1038/s41559-019-0899-x

Burkle, L. A., Marlin, J. C., & Knight, T. M. (2013). Plant-Pollinator Interactions over 120 Years: Loss of Species, Co-Occurrence, and Function. *Science*, *339*(6127), 1611–1615. https://doi.org/10.1126/science.1232728

Camerano, L. (1880). *Dell’equilibrio dei viventi merce la reciproca distruzione*. stamperia reale della ditta GB Paravia e Comp.

CHEN Jianyang, YIN Haiwei, KONG Fanhua, & YAO Guipeng. (2015). Study on the complex eco-network development around Taihu Lake, China. *Acta Ecologica Sinica*, *35*(9). https://doi.org/10.5846/stxb201306101647

Darwin, C. (1959). The Origin of Species, 1859-1959. *Bios*, *30*(2), 67–72.

Du, C., Cui, J., Wang, D., Li, G., Lu, H., Tian, Z., Zhao, C., Li, M., & Zhang, L. (2024). Prediction of aquatic vegetation growth under ecological recharge based on machine learning and remote sensing. *Journal of Cleaner Production*, *452*, 142054. https://doi.org/10.1016/j.jclepro.2024.142054

Egerton, F. N. (2016). Part 57: Aspects of Limnology in America, 1930s to about 1990, Led by Hutchinson and Hasler. *The Bulletin of the Ecological Society of America*, *97*(3), 228–282. https://doi.org/10.1002/bes2.1242

Forup, M. L., Henson, K. S. E., Craze, P. G., & Memmott, J. (2008). The restoration of ecological interactions: plant–pollinator networks on ancient and restored heathlands. *Journal of Applied Ecology*, *45*(3), 742–752. https://doi.org/10.1111/j.1365-2664.2007.01390.x

Friberg, N., Bonada, N., Bradley, D. C., Dunbar, M. J., Edwards, F. K., Grey, J., Hayes, R. B., Hildrew, A. G., Lamouroux, N., Trimmer, M., & Woodward, G. (2011). *Biomonitoring of Human Impacts in Freshwater Ecosystems* (pp. 1–68). https://doi.org/10.1016/B978-0-12-374794-5.00001-8

Gao, X., Liang, X., Guo, Z., Yuan, Z., Li, G., & Likulunga, L. E. (2024). The analysis of ecological network characteristics of Baiyangdian Basin in China using the complex network theory. *Ecological Indicators*, *167*, 112650. https://doi.org/10.1016/j.ecolind.2024.112650

Gray, C., Baird, D. J., Baumgartner, S., Jacob, U., Jenkins, G. B., O’Gorman, E. J., Lu, X., Ma, A., Pocock, M. J. O., Schuwirth, N., Thompson, M., & Woodward, G. (2014). FORUM: Ecological networks: The missing links in biomonitoring science. *Journal of Applied Ecology*, *51*(5), 1444–1449. https://doi.org/10.1111/1365-2664.12300

Green, D. G., & Sadedin, S. (2005). Interactions matter—complexity in landscapes and ecosystems. *Ecological Complexity*, *2*(2), 117–130. https://doi.org/10.1016/j.ecocom.2004.11.006

Heleno, R., Devoto, M., & Pocock, M. (2012). Connectance of species interaction networks and conservation value: Is it any good to be well connected? *Ecological Indicators*, *14*(1), 7–10. https://doi.org/10.1016/j.ecolind.2011.06.032

Hill, S. L., Keeble, K., Atkinson, A., & Murphy, E. J. (2012). A foodweb model to explore uncertainties in the South Georgia shelf pelagic ecosystem. *Deep Sea Research Part II: Topical Studies in Oceanography*, *59–60*, 237–252. https://doi.org/10.1016/j.dsr2.2011.09.001

Huettmann, F., Craig, E. H., Herrick, K. A., Baltensperger, A. P., Humphries, G. R. W., Lieske, D. J., Miller, K., Mullet, T. C., Oppel, S., Resendiz, C., Rutzen, I., Schmid, M. S., Suwal, M. K., & Young, B. D. (2018). Use of Machine Learning (ML) for Predicting and Analyzing Ecological and ‘Presence Only’ Data: An Overview of Applications and a Good Outlook. In *Machine Learning for Ecology and Sustainable Natural Resource Management* (pp. 27–61). Springer International Publishing. https://doi.org/10.1007/978-3-319-96978-7\_2

HUTCHINSON, H. W. (1958). Village and Plantation Life in Northeastern Brazil. *Soil Science*, *85*(1), 56. https://doi.org/10.1097/00010694-195801000-00019

Ignatieva, M., Stewart, G. H., & Meurk, C. (2011). Planning and design of ecological networks in urban areas. *Landscape and Ecological Engineering*, *7*(1), 17–25. https://doi.org/10.1007/s11355-010-0143-y

Ings, T. C., Montoya, J. M., Bascompte, J., Blüthgen, N., Brown, L., Dormann, C. F., Edwards, F., Figueroa, D., Jacob, U., Jones, J. I., Lauridsen, R. B., Ledger, M. E., Lewis, H. M., Olesen, J. M., Van Veen, F. J. F., Warren, P. H., & Woodward, G. (2009). Review: Ecological networks – beyond food webs. *Journal of Animal Ecology*, *78*(1), 253–269. https://doi.org/10.1111/j.1365-2656.2008.01460.x

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., & Poupart, P. (2019). *Representation Learning for Dynamic Graphs: A Survey*.

Kremen, C. (2005). Managing ecosystem services: what do we need to know about their ecology? *Ecology Letters*, *8*(5), 468–479. https://doi.org/10.1111/j.1461-0248.2005.00751.x

Layer, K., Hildrew, A. G., Jenkins, G. B., Riede, J. O., Rossiter, S. J., Townsend, C. R., & Woodward, G. (2011). *Long-Term Dynamics of a Well-Characterised Food Web* (pp. 69–117). https://doi.org/10.1016/B978-0-12-374794-5.00002-X

Layer, K., Hildrew, A. G., & Woodward, G. (2013). Grazing and detritivory in 20 stream food webs across a broad pH gradient. *Oecologia*, *171*(2), 459–471. https://doi.org/10.1007/s00442-012-2421-x

Layer, K., Riede, J. O., Hildrew, A. G., & Woodward, G. (2010). *Food Web Structure and Stability in 20 Streams Across a Wide pH Gradient* (pp. 265–299). https://doi.org/10.1016/B978-0-12-381363-3.00005-8

Ledger, M. E., Brown, L. E., Edwards, F. K., Hudson, L. N., Milner, A. M., & Woodward, G. (2013). *Extreme Climatic Events Alter Aquatic Food Webs* (pp. 343–395). https://doi.org/10.1016/B978-0-12-417199-2.00006-9

Leman, A. (2018). THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THE ALGEBRA WHICH APPEARS THEREIN. *Null*. https://doi.org/null

Liu, H., Niu, T., Yu, Q., Yang, L., Ma, J., & Qiu, S. (2022). Evaluation of the Spatiotemporal Evolution of China’s Ecological Spatial Network Function–Structure and Its Pattern Optimization. *Remote Sensing*, *14*(18), 4593. https://doi.org/10.3390/rs14184593

Malmqvist, T. (2008). Environmental rating methods: selecting indoor environmental quality (IEQ) aspects and indicators. *Building Research & Information*, *36*(5), 466–485. https://doi.org/10.1080/09613210802075841

McCann, K. S. (2000). The diversity–stability debate. *Nature*, *405*(6783), 228–233. https://doi.org/10.1038/35012234

Montoya, J. M., Pimm, S. L., & Solé, R. V. (2006). Ecological networks and their fragility. *Nature*, *442*(7100), 259–264. https://doi.org/10.1038/nature04927

Reddy, S. (2024). Examine the effects of climate change in an ecological network using machine learning. Queen Mary University of London.

Skarding, J., Gabrys, B., & Musial, K. (2021). Foundations and Modeling of Dynamic Networks Using Dynamic Graph Neural Networks: A Survey. *IEEE Access*, *9*, 79143–79168. https://doi.org/10.1109/ACCESS.2021.3082932

Solomon, S. (2007). IPCC (2007): Climate Change The Physical Science Basis. *AGU Fall Meeting Abstracts*, *2007*, U43D-01.

Swenson, S. C., & Milly, P. C. D. (2006). Climate model biases in seasonality of continental water storage revealed by satellite gravimetry. *Water Resources Research*, *42*(3). https://doi.org/10.1029/2005WR004628

Thomas, J. M., Beddar-Wiesing, S., Moallemy-Oureh, A., & Nather, R. (2021). *A Note on the Modeling Power of Different Graph Types*.

Thompson, R. M., Brose, U., Dunne, J. A., Hall, R. O., Hladyz, S., Kitching, R. L., Martinez, N. D., Rantala, H., Romanuk, T. N., Stouffer, D. B., & Tylianakis, J. M. (2012). Food webs: reconciling the structure and function of biodiversity. *Trends in Ecology & Evolution*, *27*(12), 689–697. https://doi.org/10.1016/j.tree.2012.08.005

Tylianakis, J. M., Laliberté, E., Nielsen, A., & Bascompte, J. (2010). Conservation of species interaction networks. *Biological Conservation*, *143*(10), 2270–2279. https://doi.org/10.1016/j.biocon.2009.12.004

Tylianakis, J. M., Tscharntke, T., & Lewis, O. T. (2007). Habitat modification alters the structure of tropical host–parasitoid food webs. *Nature*, *445*(7124), 202–205. https://doi.org/10.1038/nature05429

Webb, L., Whetton, P., & Barlow, E. (2008). Climate change and winegrape quality in Australia. *Climate Research*, *36*, 99–111. https://doi.org/10.3354/cr00740

Zhang, L., Peng, J., Liu, Y., & Wu, J. (2017). Coupling ecosystem services supply and human ecological demand to identify landscape ecological security pattern: A case study in Beijing–Tianjin–Hebei region, China. *Urban Ecosystems*, *20*(3), 701–714. https://doi.org/10.1007/s11252-016-0629-y

Zhang, M., & Chen, Y. (2017). Weisfeiler-lehman neural machine for link prediction. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *Part F129685*, 575–583. https://doi.org/10.1145/3097983.3097996

Zhang, S., Zhang, W., Bu, Z., & Zhang, X. (2024). ClusterLP: A novel Cluster-aware Link Prediction model in undirected and directed graphs. *International Journal of Approximate Reasoning*, *172*, 109216. https://doi.org/10.1016/j.ijar.2024.109216

Zhang, Z., Hu, B., Jiang, W., & Qiu, H. (2023). Spatial and temporal variation and prediction of ecological carrying capacity based on machine learning and PLUS model. *Ecological Indicators*, *154*, 110611. https://doi.org/10.1016/j.ecolind.2023.110611