Research and implementation of multi-dataset training for image classification with discrepant taxonomies

A master thesis in the field of computer science

by

Björn Buschkämper

1st supervisor: Dr. Petra Bevandic

2nd supervisor: M.Sc. Riza Velioglu

Submitted in the research group "HammerLab"
of the faculty of technology for the degree of
Master of Science
at
Universität Bielefeld

June 18, 2025

Abstract

Scientific documents often use LaTeX for typesetting. While numerous packages and templates exist, it makes sense to create a new one. Just because.

Contents

I Introduction

2 Methodology

Our main goal is to create a universal taxonomy that connects multiple image classification datasets. This taxonomy maps every dataset class to a universal class, which allows us to analyse the relationships and shared concepts between datasets.

In the end, our taxonomy will allow us to train models that can classify images from multiple datasets at once, building a robust and flexible system that can quickly adapt to new domains.

2.1 Formal Definitions

To formalise our algorithm for building a universal taxonomy, we first need to define some terms:

- **Dataset** D: A collection of images and labels written as $D = \{(x_1, c_1), (x_2, c_2), \dots, (x_n, c_n)\}$, where x_i is an image and c_i is its label. Since we are dealing with multiple datasets, we number them as $D_i = \{(x_1^i, c_1^i), (x_2^i, c_2^i), \dots, (x_n^i, c_n^i)\}$, where D_i is the dataset D with index i. In the same way, we denote the set of all classes in a dataset as $C_i = \{c_1^i, c_2^i, \dots, c_k^i\}$.

- **Model** m: A neural network trained on a dataset D_I which maps an image $x \in X$ to a class $c_i^I \in C_I$, denoted as $m_I : X \mapsto C_I$.

- **Domain**: Since both models and classes are dataset-specific, we define the term **domain** as the dataset D_i and its classes C_i that we are working with.

- **Universal Classes**: Our universal taxonomy will contain a set of classes that are not specific to any dataset. We denote these classes as $C_U = \{c_1^U, c_2^U, \dots, c_k^U\}$. A universal class is a concept represented by a set of domain classes that share similar characteristics. We therefore define a function $\text{classes} : C_U \mapsto \mathcal{P}(C)$, where $\mathcal{P}(C)$ is the power set of C, to represent the set of domain classes that belong to a universal class.

- **Graph**: We represent our taxonomy as a directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. Each vertex v_i represents a single class or universal class, which we define with $\text{class} : V \mapsto C$. Every edge e_{ij} between two vertices v_i and v_j indicates a relationship $\text{class}(v_i) \rightarrow \text{class}(v_j)$.

- **Probability**: Every edge e_{ij} has a probability associated with it, which indicates the likelihood of classifying an image from class class(v_i) as class class(v_j). We denote this as a function probability : $E \mapsto [0, 1]$.

2.2 Cross-Domain Graph Generation

Before building our universal taxonomy, we need to construct our initial graph that captures the relationships between classes across different domains:

1. **Foreign predictions:** For each dataset with its corresponding model, we run the model on all images from all other datasets. This gives us a set of predictions $P_{ab} = \{(x_i^a, c_j^b)\}$, where x_i^a is an image from dataset D_a and c_j^b is the class predicted by model m_b for that image.

2. **Prediction probabilities:** We count the number of times each class c_i^a was predicted as a foreign-domain class c_j^b. We denote this count in a matrix $M_{ab} \in \mathbb{N}^{+|C_a|\times|C_b|}$, where $M_{ab}(i, j)$ is the number of times class c_i^a was predicted as class c_j^b. We then divide each entry in the matrix by its row sum to get the probability of classifying an image from class c_i^a as class c_j^b:

$$P_{ab}(i, j) = \frac{M_{ab}(i, j)}{\sum_{k=1}^{|C_a|} M_{ab}(i, k)}$$

   This gives us a matrix $P_{ab} \in [0, 1]^{|C_a|\times|C_b|}$, where $P_{ab}(i, j)$ is the probability of classifying an image from class c_i^a as class c_j^b.

3. **Graph construction:** We now create a directed graph that represents the relationships between classes and datasets by iterating over every dataset D_a with every dataset D_b where $a \neq b$ for cross-predictions:

   a) We want to evaluate different methods for selecting the most relevant relationships, so we formalise a function select_relationships(P_{ab}) : $[0, 1]^{|C_a|\times|C_b|} \mapsto \mathcal{P}(\mathbb{N}^2)$ that selects a set of relationships from the probability matrix P_{ab}.

   b) For every $(i, j) \in$ select_relationships(P_{ab}):

      i. We create the vertices v_k and v_l for classes c_i^a and c_j^b respectively if they do not already exist and add them to the graph (otherwise we find the existing vertices for these classes as v_k and v_l).

      ii. We create an edge e_{kl} between the vertices v_k and v_l and add it to the graph.

      iii. We define probability(e_{kl}) = $P_{ab}(i, j)$.

2.3 Synthetic Taxonomy Generation

2.3.1 The Need for a Controlled Ground Truth

To evaluate our taxonomy generation methods, we need a reliable ground truth with known relationships between datasets. This presents a challenge, as most existing image classification datasets lack clear inter-dataset relationships:

- **ImageNet** [1, 6] uses WordNet's [2] hierarchical structure to organize classes. However, this strict hierarchy doesn't match our use case where we need to connect datasets with different class structures and partial overlaps.
- **Open Images** [5] contains approximately 9 million images with multiple labels per image generated by Google's Cloud Vision API[1]. This multi-label approach makes it difficult to determine a single class for each image, which is required for our evaluation. Additionally, since most labels were automatically generated, it doesn't provide the verified ground truth we need.
- **iNaturalist** [3] offers a detailed taxonomy of plant and animal species, but its domain-specific nature makes it unsuitable for developing a general-purpose evaluation framework.

2.3.2 Our Approach: Building Synthetic Datasets

Instead of relying on existing taxonomies, we developed a method to generate synthetic datasets with controlled relationships. Our approach:

1. Define a set of "atomic concepts" that serve as building blocks for classes
2. Create multiple domains by sampling these concepts to form classes
3. Calculate inter-domain relationships based on shared concepts

This method allows us to precisely control the taxonomy structure while creating realistic relationships between domains. To generate images for these synthetic classes, we can leverage existing datasets by treating each original class as an atomic concept.

2.3.3 Formal Definitions

We define our synthetic taxonomy framework on top of the definitions from section 2.1:

[1] https://cloud.google.com/vision

- **Atomic Concepts** $\mathcal{U} = \{1, 2, \ldots, n\}$: A set of atomic concepts will be a universe of concepts that make up the basis for our synthetic class generation.
- **Synthetic Class**: A class c_j^i will contain a subset of the atomic concepts from our universe: $c_j^i \subseteq \mathcal{U}$. To maintain disjoint class definitions, we ensure that $c_j^i \cap c_k^i = \emptyset$ for all $j \neq k$.

2.3.4 Randomized Domain Generation

To create realistic domains, we use normal distributions to sample the number of classes and concepts per class. This allows us to generate domains with varying sizes and complexities, mimicking different real-world datasets.

Parameter Sampling

We sample the number of classes per domain and the number of concepts per class from truncated normal distributions to ensure realistic variation while maintaining control. Since normal distributions are unbounded, we use a truncated version:

$$f(x|\mu, \sigma, a, b) = \begin{cases} \frac{\phi\left(\frac{x-\mu}{\sigma}\right)}{\sigma\left[\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)\right]} & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

Where:

- ϕ is the standard normal PDF
- Φ is the standard normal CDF
- a and b are lower and upper bounds

We implement this using SciPy's `truncnorm` module[2], handling SciPy's standardization of bounds internally:

$$X \sim \text{TruncNorm}(\mu, \sigma^2, a, b)$$

Domain Generation Algorithm

To generate a domain C_i, we follow these steps:

1. **Sample set size:** Determine how many concepts l to use for the domain:

$$l \sim \lfloor \text{TruncNorm}(\mu_{\text{classes}}, \sigma^2_{\text{classes}}, 1, n) \rfloor$$

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.truncnorm.html

2. **Sample concept pool:** Randomly select l concepts from the universe $\mathcal{U}$:

$$P = \{a, b, c, ...\} \quad \text{where } a, b, c, ... \text{ are sampled without replacement from } \mathcal{U}$$

3. **Initialise domain:** $C_i = \{\}$

4. **Generate classes:** While concepts remain in the pool ($P \neq \emptyset$):

   a) Sample class size s_j:

$$s_j \sim \lfloor \text{TruncNorm}(\mu_{\text{class\_size}}, \sigma^2_{\text{class\_size}}, 1, |P|) \rfloor$$

   b) Form class c_j^i by selecting s_j concepts randomly from P

   c) Remove selected concepts: $P = P \setminus c_j^i$

   d) Add class to domain: $C_i = C_i \cup \{c_j^i\}$

This algorithm ensures that each concept is assigned to exactly one class within the domain, maintaining our disjointness constraint.

2.3.5 Modeling Cross-Domain Relationships

Once we've generated multiple domains, we need to model the relationships between them to create our ground truth.

Simulating Neural Network Predictions

Our taxonomy generation method assumes that neural network classifiers will predict related classes across domains with certain probabilities. To simulate this, we create "perfect" synthetic probabilities based on concept overlap.

Relationship Calculation

For any two domains C_A and C_B, we calculate the probability of classifying an instance of class c_i^A as class c_j^B using:

$$\begin{aligned} \text{NaiveProbability}(i, j) &= \frac{|c_i^A \cap c_j^B|}{|c_i^A|} \\ P_{i,j} &= \text{NaiveProbability}(i, j) + \frac{1 - \text{NaiveProbability}(i, j)}{|C_B|} \end{aligned} \tag{2.1}$$

Where:

- NaiveProbability(i, j) is the proportion of concepts in class c_i^A that also appear in class c_j^B
- The second term distributes remaining probability mass evenly across all classes in domain C_B, simulating the behavior of a neural network when encountering concepts it hasn't seen before

A Concrete Example

To illustrate this approach, consider two domains:

- Domain A: $C_A = \{c_1^A = \{1, 2\}, c_2^A = \{3, 4\}\}$
- Domain B: $C_B = \{c_1^B = \{1, 2, 4\}, c_2^B = \{5, 6\}\}$

For the relationship $c_1^A \rightarrow c_1^B$:

- NaiveProbability$(1, 1) = \frac{|\{1,2\} \cap \{1,2,4\}|}{|\{1,2\}|} = \frac{2}{2} = 1$
- $P_{1,1} = 1 + \frac{1-1}{2} = 1$

For the relationship $c_2^A \rightarrow c_1^B$:

- NaiveProbability$(2, 1) = \frac{|\{3,4\} \cap \{1,2,4\}|}{|\{3,4\}|} = \frac{1}{2} = 0.5$
- $P_{2,1} = 0.5 + \frac{1-0.5}{2} = 0.5 + 0.25 = 0.75$

This example shows how our framework captures partial relationships between classes and how it simulates a perfect neural network classifier's behavior. The resulting probability prediction matrix between two domains can then be used to build a graph of relationships between classes, which can then be turned into a universal taxonomy using the methods described in section 2.2.

No-Prediction Classes

Some datasets have a special class that indicates that the model could not classify the image. For these "no-prediction" classes, we need to adapt the relationship probability calculation: Instead of distributing the remaining probability mass evenly across all classes, we simply ignore it and therefore only have the probability of the overlapping concepts.

While this doesn't change the relationship that is picked as the most common foreign prediction, it does change the probability of the relationship, which is important for thresholding later on.

2.4 Universal Taxonomy Algorithm

After constructing our initial graph structure from cross-domain predictions (as described in section 2.2), we now need to transform it into a universal taxonomy that merges classes from different datasets into universal classes where they share similar concepts.

2.4.1 Taxonomy Building Rules

1. **Isolated Node Rule:** For any domain class A that has no relationships (neither incoming nor outgoing edges), create a new universal class B and add the relationship $A \rightarrow B$. We also define the probability of the relationship's edge as 1 and the classes of the universal class as $\{A\}$.

   This ensures that all domain classes without relationships (which can be created by later rules) are still represented in the universal taxonomy.

2. **Bidirectional Relationship Rule:** When two classes have bidirectional relationships (A $\rightarrow$ B and B $\rightarrow$ A), they likely represent the same concept. We resolve this by creating a new universal class C and adding relationships $A \rightarrow C$ and $B \rightarrow C$ to the graph. The probability of the new relationships is set to the average of the bidirectional relationships and the classes of the universal class will be the two classes that were merged (or, if the two classes are universal classes themselves, the union of their classes).

3. **Transitive Cycle Rule:** If we have relationships A $\rightarrow$ B $\rightarrow$ C where A and C are in the same domain, we have a problem since classes within a domain are disjoint, which means that one of the relationships must be incorrect. We solve this by removing the relationship with the lower probability, thus breaking the cycle.

4. **Unilateral Relationship Rule:** A uniliteral relationship A $\rightarrow$ B indicates that the concepts of class A are a subset of the concepts of class B. We therefore create two new universal classes:

   - Class C, which contains both classes A and B and has incoming relationships from both classes with the probability of the uniliteral relationship. This universal class represents the union of the two classes.
   - Class D, which contains only class B and has a relationship from class B with a probability 1. This universal class represents the concepts of class B that are not in class A.

2.5 Taxonomy Difference Metrics

Now that we have methods for generating a ground truth synthetic taxonomy, we need to define metrics to compare the predicted taxonomy against the ground truth. Comparison can happen at two points in our pipeline:

- **Universal Taxonomy Comparison:** Comparing the predicted universal taxonomy against the ground truth universal taxonomy. This is done after applying our universal taxonomy generation algorithm and allows us to evaluate the quality of the final taxonomy. However, the algorithm might change the scale of differences between our predicted and ground truth taxonomies (e.g. a unilateral vs. bidirectional relationship would be a small difference before the algorithm, but would result in a subset hypothesis with two universal classes vs. one universal class after the algorithm).

- **Relationship Graph Comparison:** Comparing the predicted graph of relationships between classes against the ground truth graph. This is done before converting the relationship graph into a universal taxonomy and allows us to evaluate the quality of the relationships between classes.

2.5.1 Constructing Adjacency Matrices

For our metrics, we first need to represent our intra-domain relationships as adjacency matrices. We concatinate every class from every domain into a single set of classes $C = \bigcup_{i=1}^{n} C_i$, where n is the number of domains. We then create an adjacency matrix $A \in [0, 1]^{|C|\times|C|}$, where $A(i, j)$ is the relationship probability between classes c_i and c_j.

Additionally, we need to handle the case where a class has no relationships at all: Since these classes will later become a single universal class, we additionally create a self-loop for every class c_i without relationships, which is defined as $A(i, i) = 1$.

2.5.2 Edge Difference Ratio

Our first metric is the edge difference ratio (EDR), which measures the difference in edge weights between two relationship graphs G_1 and G_2. The metric is bounded between 0 and 1, where 0 indicates that the two graphs are identical and 1 indicates that the two graphs have no edges in common.

For two adjacency matrices A_1 and A_2 of graphs G_1 and G_2, we define the edge difference ratio as follows:

$$\mathrm{EDR}(G_1, G_2) = \frac{\sum_{i,j} |A_1(i,j) - A_2(i,j)|}{\sum_{i,j} \max(A_1(i,j), A_2(i,j))} \tag{2.2}$$

This definition captures the difference in edge weights between the two graphs, while normalizing it by the total edge weights in both graphs (without double counting edges).

Our EDR metric is similar to the Jaccard index [4] as well as the Tanimoto coefficient [7] when we consider the adjacency matrices as sets of edges. In contrast to these metrics, however, our EDR metric supports weighted edges, which allows us to respect the probabilities of relationships between classes.

2.5.3 Precision, Recall, and F1 Score

While the edge weights in our relationship graphs are important, every single edge (even with a very low probability) can create a new universal class and therefore change the universal taxonomy.

To account for this, we also define precision, recall, and F1 score metrics for the relationship graphs.

For two adjacency matrices A_1 and A_2 of graphs G_1 and G_2, we first create binarised versions of the matrices as B_1 and B_2, where $B_1(i,j) = 1$ if $A_1(i,j) > 0$ and $B_2(i,j) = 1$ if $A_2(i,j) > 0$.

Next, we compute the true positives, false positives, and false negatives as follows:

- **True Positives (TP):** The number of edges that are present in both B_1 and B_2.
- **False Positives (FP):** The number of edges that are present in B_1 but not in B_2.
- **False Negatives (FN):** The number of edges that are present in B_2 but not in B_1.

Using these counts, we can then compute the precision, recall, and F1 score as follows:

- **Precision:** The ratio of true positives to the sum of true positives and false positives.
- **Recall:** The ratio of true positives to the sum of true positives and false negatives.
- **F1 Score:** The harmonic mean of precision and recall.

Bibliography

1. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.
2. C. Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998. ISBN: 9780262272551. DOI: 10.7551/mitpress/7287.001.0001. URL: https://doi.org/10.7551/mitpress/7287.001.0001.
3. G. V. Horn, O. M. Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. *The iNaturalist Species Classification and Detection Dataset*. 10, 2018. DOI: 10.48550/arXiv.1707.06642. arXiv: 1707.06642[cs]. URL: http://arxiv.org/abs/1707.06642 (visited on 05/03/2025).
4. P. Jaccard. "The Distribution of the Flora in the Alpine Zone." *New Phytologist* 11:2, 1912, pp. 37–50. ISSN: 1469-8137. DOI: 10.1111/j.1469-8137.1912.tb05611.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8137.1912.tb05611.x (visited on 06/01/2025).
5. A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari. "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale". *IJCV*, 2020.
6. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 30, 2015. DOI: 10.48550/arXiv.1409.0575. arXiv: 1409.0575[cs]. URL: http://arxiv.org/abs/1409.0575 (visited on 03/19/2025).
7. T. T. Tanimoto. *An Elementary Mathematical Theory of Classification and Prediction*. Google-Books-ID: yp34HAAACAAJ. International Business Machines Corporation, 1958. 10 pp.