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

May 4, 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

1 Introduction

In which the reasons for creating this package are laid bare for the whole world to see and we encounter some usage guidelines.

This package contains a minimal, modern template for writing your thesis. While originally meant to be used for a Ph. D. thesis, you can equally well use it for your honour thesis, bachelor thesis, and so on—some adjustments may be necessary, though.

1.1 Why?

I was not satisfied with the available templates for LaTeX and wanted to heed the style advice given by people such as Robert Bringhurst or Edward R. Tufte. While there *are* some packages out there that attempt to emulate these styles, I found them to be either too bloated, too playful, or too constraining. This template attempts to produce a beautiful look without having to resort to any sort of hacks. I hope you like it.

1.2 How?

The package tries to be easy to use. If you are satisfied with the default settings, just add

```
\documentclass{mimosis}
```

at the beginning of your document. This is sufficient to use the class. It is possible to build your document using either LaTeX|, XeLaTeX, or LuaLaTeX. I personally prefer one of the latter two because they make it easier to select proper fonts.

1.3 Making this template *yours*

Prior to using this template, the first thing you want to do is probably a little bit of customisation. You can achieve quick changes in look and feel by picking your own fonts. With the `fontspec` package loaded and XeLaTeXor LuaLaTeXas your compiler, this is pretty simple:

```
\setmainfont{Your main font}
\setsansfont{Your sans-serif font}
\setmonofont{Your monospaced font}
```

Make sure to select nice combinations of that are pleasing to *your* eyes—this is your document and it should reflect your own style. Make sure to specify font names as they are provided by your system. For instance, you might want to use the following combination:

```
\setmainfont{Libre Baskerville}
\setsansfont[Scale=MatchLowercase]{IBM Plex Sans}
\setmonofont[Scale=MatchLowercase]{IBM Plex Mono}
```

If these fonts exist on your system, your normal text will look a little bit different from the other font used in this example PDF, while your sans-serif font will pair nicely with your `monospaced font`. You can also remove the `Scale` directive, but I find that most fonts pair better if they are adjusted in size a little bit. Experiment with it until you finds a combination that you enjoy.

XeLaTeXand LuaLaTeXalso offer you a way to change the font that is used for mathematical equations. If installed, the `garamond-math` package permits you to choose from different stylistic sets that slightly change how certain mathematical symbols look. For instance, the following command changes 'Fraktur' symbols:

```
\setmathfont{Garamond-Math.otf}[StylisticSet={6}]
```

1.4 Features

The template automatically imports numerous convenience packages that aid in your typesetting process. Table 1.1 lists the most important ones. Let's briefly discuss some examples below. Please refer to the source code for more demonstrations.

1.4.1 Typesetting mathematics

This template uses `amsmath` and `amssymb`, which are the de-facto standard for typesetting mathematics. Use numbered equations using the `equation` environment. If you want to show multiple equations and align them, use the `align` environment:

$$V := \{1, 2, \dots\} \tag{1.1}$$

$$E := \{(u, v) \mid \operatorname{dist}(p_u, p_v) \leq \epsilon\} \tag{1.2}$$

Package	Purpose
`amsmath`	Basic mathematical typography
`amsthm`	Basic mathematical environments for proofs etc.
`babel`	Language settings
`booktabs`	Typographically light rules for tables
`bookmarks`	Bookmarks in the resulting PDF
`csquotes`	Language-specific quotation marks
`dsfont`	Double-stroke font for mathematical concepts
`graphicx`	Graphics
`hyperref`	Hyperlinks
`multirow`	Permits table content to span multiple rows or columns
`paralist`	Paragraph ('in-line') lists and compact enumerations
`scrlayer-scrpage`	Page headings
`setspace`	Line spacing
`siunitx`	Proper typesetting of units
`subcaption`	Proper sub-captions for figures

Table 1.1: A list of the most relevant packages required (and automatically imported) by this template.

Define new mathematical operators using `\DeclareMathOperator`. Some operators are already pre-defined by the template, such as the distance between two objects. Please see the template for some examples. Moreover, this template contains a correct differential operator. Use `\diff` to typeset the differential of integrals:

$$f(u) := \int_{v \in \mathbb{D}} \operatorname{dist}(u, v) \, \mathrm{d}v \tag{1.3}$$

You can see that, as a courtesy towards most mathematicians, this template gives you the possibility to refer to the real numbers $\mathbb{R}$ and the domain $\mathbb{D}$ of some function. Take a look at the source for more examples. By the way, the template comes with spacing fixes for the automated placement of brackets.

1.4.2 Typesetting text

Along with the standard environments, this template offers `paralist` for lists within paragraphs. Here's a quick example: The American constitution speaks, among others, of (i) life (ii) liberty (iii) the pursuit of happiness. These should be added in equal measure to your own conduct. To typeset units correctly, use the `siunitx` package. For example, you might want to restrict your daily intake of liberty to 750 mg.

Likewise, as a small pet peeve of mine, I offer specific operators for *ordinals*. Use `\th` to typeset things like July 4$^{\text{th}}$ correctly. Or, if you are referring to the 2$^{\text{nd}}$ edition of a book, please use `\nd`. Likewise, if you came in 3$^{\text{rd}}$ in a marathon, use `\rd`. This is my 1$^{\text{st}}$ rule.

If you want to write a text in German and use German hyphenation rules, set the language of your text to german using `\selectlanguage{ngerman}`, or add

```
\PassOptionsToPackage{spanish}{babel}
```

before the `\documentclass` command to load a specific language. The languages `ngerman`, `french`, and `english` are loaded by default, with `english` being selected.

Quotation marks can be typeset using the `\enquote{...}` command from the `csquotes` package, which is preloaded by `latex-mimosis`. Depending on the currently selected language, quotes will look like “this”, „this“, or « this ». One must never use ”ASCII” quotation marks or even ’apostrophe’ symbols.

1.5 Changing things

Since this class heavily relies on the `scrbook` class, you can use *their* styling commands in order to change the look of things. For example, if you want to change the text in sections to **bold** you can just use

```
\setkomafont{sectioning}{\normalfont\bfseries}
```

at the end of the document preamble—you don’t have to modify the class file for this. Please consult the source code for more information.

2 Methodology

In which we describe our approaches to building a universal taxonomy for image classification.

2.1 Universal Taxonomy

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.

2.1.1 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 $C_i = \{c_1^i, c_2^i, \dots, c_k^i\}$. We denote the set of all datasets as $\mathbf{D} = \{D_1, D_2, \dots, D_m\}$.

- **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 $C_U = \{c_1^U, c_2^U, \dots, c_k^U\}$.

- **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 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 $\text{class}(v_i)$ as class $\text{class}(v_j)$. We denote this as a function probability : $E \mapsto [0, 1]$.

2.1.2 Graph Construction

Before building our universal taxonomy, we need to construct our initial graph:

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 denot this count in a matrix $M_{ab} \in \mathbb{N}_0^{|C_a| \times |C_b|}$, 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 $a \neq b$:

   a) We collect the indices of the per-row maximum values in the matrix P_{ab}:

$$I = \left\{ \text{argmax}_{j \in \{1, \ldots, |C_b|\}} P_{ab}(i, j) \mid i \in \{1, \ldots, |C_a|\} \right\}$$

   b) For every $i \in \{1, \ldots, |C_a|\}$ where $P_{ab}(i, I_i) > 0$:

      i. We create the vertices v_k and v_l for classes c_i^a and $c_{I_i}^b$ 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 $\text{probability}(e_{kl}) = P_{ab}(i, I_i)$.

2.1.3 Taxonomy Generation

2.2 Synthetic Taxonomy Generation

2.2.1 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, 5] 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** [4] 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.

2.2.2 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.

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

2.2.3 Formal Definitions

We define our synthetic taxonomy framework as follows:

$$\begin{aligned}
&\mathcal{U} = \{1, 2, \ldots, n\} \quad \text{(Universe of atomic concepts)} \\
&\mathcal{C} \subseteq \mathcal{U} \quad \text{(A class is a subset of concepts)} \\
&\mathcal{D}_i = \{\mathcal{C}_1^i, \mathcal{C}_2^i, \ldots, \mathcal{C}_k^i\} \quad \text{with } \forall j \neq k : \mathcal{C}_j^i \cap \mathcal{C}_k^i = \emptyset \quad \text{(A domain is a set of disjoint classes)} \\
&\mathcal{T} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m\} \quad \text{(A taxonomy is a set of domains)}
\end{aligned} \tag{2.1}$$

2.2.4 Randomized Domain Generation

To create realistic domains, we use controlled randomness:

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:

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

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

Domain Generation Algorithm

To generate a domain $\mathcal{D}_i$, we follow these steps:

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

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

2. **Create concept pool:** Randomly select l concepts from the universe:

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

3. **Initialize domain:** $\mathcal{D}_i = \{\}$

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 $\mathcal{C}^i_j$ by selecting s_j concepts randomly from P

   c) Remove selected concepts from P: $P = P \setminus \mathcal{C}^i_j$

   d) Add class to domain: $\mathcal{D}_i = \mathcal{D}_i \cup \{\mathcal{C}^i_j\}$

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

2.2.5 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 $\mathscr{D}_A$ and $\mathscr{D}_B$, we calculate the probability of classifying an instance of class $\mathscr{C}_i^A$ as class $\mathscr{C}_j^B$ using:

$$\begin{aligned}\text{NaiveProbability}(i,j) &= \frac{|\mathscr{C}_i^A \cap \mathscr{C}_j^B|}{|\mathscr{C}_i^A|} \\ P_{i,j} &= \text{NaiveProbability}(i,j) + \frac{1 - \text{NaiveProbability}(i,j)}{|\mathscr{D}_B|}\end{aligned} \tag{2.2}$$

Where:

- $\text{NaiveProbability}(i,j)$ is the proportion of concepts in class $\mathscr{C}_i^A$ that also appear in class $\mathscr{C}_j^B$
- The second term distributes remaining probability mass evenly across all classes in domain $\mathscr{D}_B$, 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: $\mathscr{D}_A = \{\mathscr{C}_1^A = \{1,2\}, \mathscr{C}_2^A = \{3,4\}\}$
- Domain B: $\mathscr{D}_B = \{\mathscr{C}_1^B = \{1,2,4\}, \mathscr{C}_2^B = \{5,6\}\}$

For the relationship $\mathscr{C}_1^A \rightarrow \mathscr{C}_1^B$:

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

For the relationship $\mathscr{C}_2^A \rightarrow \mathscr{C}_1^B$:

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

This example shows how our framework captures partial relationships between classes and simulates how a neural network might handle concepts that don't perfectly match across domains.

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. 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.

5. 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).