

# 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

1<sup>st</sup> supervisor: Dr. Petra Bevandic

2<sup>nd</sup> 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

*August 29, 2025*



## ABSTRACT

Scientific documents often use L<sup>A</sup>T<sub>E</sub>X for typesetting. While numerous packages and templates exist, it makes sense to create a new one. Just because.



# CONTENTS

1	INTRODUCTION	I
2	RELATED WORK	3
3	METHODOLOGY	5
3.1	Formal Definitions	5
3.2	Cross-Domain Graph Generation	6
3.2.1	Selecting Relationships	7
3.3	Synthetic Taxonomy Generation	8
3.3.1	The Need for a Controlled Ground Truth	8
3.3.2	Our Approach: Building Synthetic Datasets	9
3.3.3	Formal Definitions	9
3.3.4	Randomized Domain Generation	10
3.3.5	Modeling Cross-Domain Relationships	11
3.4	Universal Taxonomy Algorithm	12
3.4.1	Taxonomy Building Rules	13
3.4.2	Difference to algorithm of Bevandic et al.	13
3.5	Taxonomy Difference Metrics	15
3.5.1	Constructing Adjacency Matrices	15
3.5.2	Edge Difference Ratio	16
3.5.3	Precision, Recall, and F1 Score	16
3.6	Universal Model Learning	17
3.6.1	Creation of Mapping Matrix from Taxonomy	17
3.6.2	Learning Function for Universal Model	17
4	RESULTS	21
4.1	Domain Models	21
4.1.1	Datasets	21
4.1.2	Training	22
4.1.3	Model Performance	23

*Contents*

4.2	Taxonomy Generation . . . . .	25
4.2.1	Relationship Selection Methods . . . . .	25
4.2.2	Universal Taxonomy Generation . . . . .	32
4.3	Universal Models . . . . .	33
4.3.1	Training . . . . .	33
4.3.2	Performance . . . . .	35
5	DISCUSSION	45
6	CONCLUSION	47
	BIBLIOGRAPHY	49

# I INTRODUCTION

Image classification is one of the earliest and most widely researched tasks in the field of computer vision. It involves assigning a label to an image from a predefined set of categories. Over the years, numerous approaches have been proposed to tackle this problem, ranging from traditional handcrafted feature-based methods to modern deep learning techniques.

One main issue in the field of image classification is the limited range of categories that a single model can effectively recognize. This limitation arises from the fact that most “traditional” models are trained on specific datasets, which have a predefined set of classes to be recognised. To mitigate this issue, researchers have explored various strategies to make a single model applicable to a broader range of datasets and categories.

One of the most common approaches is to use transfer learning. Transfer learning works under the assumption that models, when trained on a large general-purpose dataset (e.g, ImageNet [5, 25]), has learned important fundamental visual features (like shapes and textures) that are useful for any downstream task. By replacing the final classification layer of these pre-trained models with a new layer that is specific to the target task/dataset, researchers can adapt the model to new categories without having to train it from scratch [24, 33].

However, this still results in separate models for each task, which can be inefficient in terms of storage and deployment. One way to address this is a shared backbone architecture, where a single model is used to extract features from the input data (the “backbone”), and multiple task-specific heads are added on top of this backbone for different tasks. This leverages the commonalities between tasks and allows for more efficient use of resources [29].

An extension of this technique are multi-task learning approaches, where a single model is trained on multiple tasks simultaneously. This enforces the learning of shared concepts and representations across tasks, which can lead to improved performance and generalization [32]. This approach however tries to distill shared features automatically, which, depending on how well the task domains align, can prove to be difficult.

We try to address this issue by first building a map of shared concepts and features across tasks and then using this map to train a single universal model that works for all targeted tasks, without any task-specific adaptations or layers. This enforces predefined shared concepts and features to be learned by the model, instead of relying on automatic distillation.

## *1 Introduction*

Our first task is finding an accurate mapping of shared concepts and features across tasks. This mapping should be created without any manual intervention. Some challenges we might face include:

- Creating a method that is able to distinguish where shared concepts and features exist across domains and where no relations exist. Depending on the domains we want to cover, there can be significant variations in the amount of inter-task similarity, which can make the number of shared concepts and features quite limited.
- Defining a consistent threshold on what is considered a shared concept or feature. The definition of a “shared concept” can vary greatly depending on the combination of tasks we try to cover: Two general-purpose datasets like ImageNet and CIFAR-100 might share straightforward concepts like “cat” and “dog” (shared concept “animals”), while more specialized datasets have no overlap at all and would rely on entirely different concepts (like specific textures, patterns, etc.). Our method needs to find a way to identify the best-matching concepts and features available across these diverse datasets.

After we have created a mapping of shared concepts and features, we can use this information to guide the training of our universal model. We will build a custom learning function that uses the mapping to create a single, shared output layer for all tasks. This output layer can then be converted to create task-specific predictions based on a static function, eliminating the need for task-specific learning.

We will compare the performance of our universal model against baseline models trained on individual domain datasets. In this thesis, we will focus on general-purpose datasets like ImageNet [5, 25], CIFAR10 and CIFAR100 [19], and Caltech101 and Caltech256 [10, 21].

## 2 RELATED WORK

We will base our method of finding shared concepts between datasets on works in the field of label-alignment and taxonomy learning. Creating connections between datasets is an important and fundamental task in the field of machine learning: Many datasets are created for specific tasks and may not be directly comparable to each other. Also, inconsistency in labeling and data collection practices can further complicate the integration of datasets. Methods to create connections between datasets can help to align labels, improve data quality, and facilitate knowledge transfer between models trained on different datasets.

The most obvious and straightforward approach to align labels across datasets is a fully manual mapping of labels from one dataset to another. This method is time-consuming and error-prone, especially for large datasets. Also, it can lead to further inconsistencies if multiple annotators are involved or if the datasets are updated independently over time [3, 17, 31].

An improvement over this manual approach is a hybrid method that works automated, but involves manual corrections. For example, Firmani et al. [9] proposes a weakly supervised approach where domain experts answer targeted questions about the datasets to guide the automated label alignment process. This method tries to balance automation and human expertise, leveraging the strengths of both.

With the rise of large language models (LLMs), new methods have emerged that replace the human domain experts with automated systems. These systems leverage the capabilities of LLMs to understand and align labels across datasets without the need for human input, creating a pseudo-hybrid approach where the human role is replaced with LLMs [4, 11, 18].

For fully automated, non-LLM-based approaches, there are several different options available:

- **WordNet** is a lexical database that groups English words into sets of synonyms called synsets, providing short definitions and usage examples. It can be used to find relationships between words and concepts, making it a valuable resource for aligning labels across datasets. However, as we will later see in 4.2.1, this method is error-prone and not suitable for all datasets.
- **Cross-Domain Classification** methods use domain models trained on a specific dataset to create cross-domain predictions. For example, a dataset  $A$  can have a class  $A:vehicle$  while a dataset  $B$  has a class  $B:car$ . By predicting an image from class  $B:car$  using a model trained

## 2 Related Work

on dataset  $A$  as  $A:vehicle$ , we can collect cross-domain co-occurrences and build a mapping between the two datasets [1, 2, 28].

We will base our taxonomy construction algorithm on the works of Bevandic et al. [1, 2]. Our method will be adapted to shift the focus from aligning labels in a universal cross-domain taxonomy towards building relationships between classes in different datasets. Relationships will not, as in the original work, define a hierarchical structure, but rather point out shared attributes between classes.

# 3 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.

## 3.1 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 class :  $V \mapsto C$ . Every edge  $e_{ij}$  between two vertices  $v_i$  and  $v_j$  indicates a relationship  $\text{class}(v_i) \rightarrow \text{class}(v_j)$ .

### 3 Methodology

- **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  $\text{probability} : E \mapsto [0, 1]$ .

## 3.2 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  $\text{select\_relationships}(P_{ab}) : [0, 1]^{|C_a| \times |C_b|} \mapsto \mathcal{P}(\mathbb{N}^2)$  that selects a set of relationships from the probability matrix  $P_{ab}$ .
- b) For every  $(i, j) \in \text{select\_relationships}(P_{ab})$ :
  - 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  $\text{probability}(e_{kl}) = P_{ab}(i, j)$ .

### 3.2.1 SELECTING RELATIONSHIPS

To now filter relationships from the probability matrix, we define a range of different methods and later evaluate their performance.

Our main challenges are:

- **Unknown number of shared concepts:** We don't know how many concepts two classes from different domains share, so we do not know how high the probability of a relationship should be.
- **Noisy predictions:** A low model accuracy can severely impact the relationship predictions, since - depending on the number of foreign classes that share concepts with the class - the target probabilities can be very low, making even a small number of wrong predictions a huge obstacle.
- **Unbalanced datasets:** Some datasets might have more images for a class than others, which can lead to skewed probabilities. This can be mitigated by preprocessing the datasets to balance the number of images per class, but our goal is to create a methodology that can be applied to any dataset without specific requirements on the datasets.

#### NAIVE THRESHOLDING

The most straightforward method is to apply a fixed threshold to the probabilities:

$$\text{select\_relationships}(P_{ab}) = \{(i, j) \mid P_{ab}(i, j) \geq t\}$$

where  $t$  is a threshold value between 0 and 1.

#### MOST COMMON FOREIGN PREDICTIONS

As in the paper that provided the ground work for our methodology [2], we can also select the single most common foreign prediction for each class:

$$\text{select\_relationships}(P_{ab}) = \{(i, j) \mid j = \operatorname{argmax}_{j'} P_{ab}(i, j')\}$$

#### DENSITY THRESHOLDING

Another approach is to use the least amount of relationships whose summed probabilities cover a certain percentage of the total probability mass. This can be done by sorting the probabilities in descending order and then selecting the smallest set of relationships that covers at least  $p$  percent of the total probability mass:

### 3 Methodology

1. We define  $R = \emptyset$  as the set of relationships to select.
2. For every  $i \in \{1, \dots, |C_a|\}$ :
  - a) Let  $X_i$  be the list of all probabilities in row  $i$  of  $P_{ab}$  sorted in descending order.
  - b) We find the smallest  $k$  such that  $\sum_{j=1}^k X_i(j) \geq p$ .
  - c) We add the first  $k$  relationships of the sorted list  $X_i$  to  $R$ .
3. We return  $R$  for the function `select_relationships( $P_{ab}$ )`.

#### RELATIONSHIP HYPOTHESIS

Let us naively assume that every relationship between two classes is based on a single shared concept. In this case, the probability of every outgoing edge from a class  $c_i^a$  should be roughly equal.

We can therefore hypothesise the probability distribution based on the number of relationships and compare this hypothesis against the actual probabilities in the matrix  $P_{ab}$ :

1. We define  $R = \emptyset$  as the set of relationships to select.
2. For every  $i \in \{1, \dots, |C_a|\}$ :
  - a) Let  $X_i$  be the list of all probabilities in row  $i$  of  $P_{ab}$  sorted in descending order.
  - b) We find the  $k \in \{1, \dots, n\}$  such that we minimise:
$$\sum_{j=1}^k \left| X_i(j) - \frac{1}{k} \right| + \sum_{j=k+1}^{|C_b|} X_i(j)$$
  - c) We add the first  $k$  relationships of the sorted list  $X_i$  to  $R$ .
3. We return  $R$  for the function `select_relationships( $P_{ab}$ )`.

In this equation,  $n$  is the upper bound of the number of relationships for a class  $c_i^a$  that we want to test against.

### 3.3 SYNTHETIC TAXONOMY GENERATION

#### 3.3.1 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** [5, 25] uses WordNet’s [8] 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** [20] contains approximately 9 million images with multiple labels per image generated by Google’s Cloud Vision API<sup>1</sup>. 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** [15] offers a detailed taxonomy of plant and animal species, but its domain-specific nature makes it unsuitable for developing a general-purpose evaluation framework.

### 3.3.2 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.

### 3.3.3 FORMAL DEFINITIONS

We define our synthetic taxonomy framework on top of the definitions from [section 3.1](#):

- **Atomic Concepts**  $\mathcal{U} = \{1, 2, \dots, n\}$ : 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  $j \neq k$ .

---

<sup>1</sup><https://cloud.google.com/vision>

### 3 Methodology

#### 3.3.4 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:

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

Where:

- $\phi$  is the standard normal PDF
- $\Phi$  is the standard normal CDF
- $a$  and  $b$  are lower and upper bounds

We implement this using SciPy's `truncnorm` module<sup>2</sup>, 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:

$$l \sim [\text{TruncNorm}(\mu_{\text{concepts}}, \sigma_{\text{concepts}}^2, 1, n)]$$

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

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

---

<sup>2</sup><https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.truncnorm.html>

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

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

a) Sample class size  $s_j$ :

$$s_j \sim [\text{TruncNorm}(\mu_{\text{classes}}, \sigma^2_{\text{classes}}, 1, |P|)]$$

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.

### 3.3.5 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:

$$\begin{aligned} \text{NaiveProbability}(i, j) &= \frac{|c_i^A \cap c_j^B|}{|c_i^A|} \\ P_{i,j} &= \text{NaiveProbability}(i, j) + \frac{1 - \text{NaiveProbability}(i, j)}{|C_B|} \end{aligned} \tag{3.1}$$

Where:

- $\text{NaiveProbability}(i, j)$  is the proportion of concepts in class  $c_i^A$  that also appear in class  $c_j^B$

### 3 Methodology

- 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  $c_1^A \rightarrow c_1^B$ :

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

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

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

This example shows how our framework captures partial relationships between classes and 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 3.2](#).

#### 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.

### 3.4 UNIVERSAL TAXONOMY ALGORITHM

After constructing our initial graph structure from cross-domain predictions (as described in [section 3.2](#)), we now need to transform it into a universal taxonomy that merges classes from different datasets into universal classes where they share similar concepts.

### 3.4.1 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 unilateral 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 unilateral 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.

### 3.4.2 DIFFERENCE TO ALGORITHM OF BEVANDIC ET AL.

The taxonomy algorithm presented above is based on the work by Bevandic et al. [1], but adapted for image classification instead of semantic segmentation. Their original approach was designed to create universal taxonomies for multi-domain semantic segmentation by addressing the problem of incompatible labeling policies across datasets.

### *3 Methodology*

#### ORIGINAL ALGORITHM

Bevandic et al. developed a procedure for constructing universal taxonomies that express dataset-specific labels as unions of disjoint universal visual concepts. Their algorithm operates on the principle that semantic classes can be viewed as sets of pixels, and uses three resolution rules to handle overlaps between classes from different datasets:

1. **Exact Match Rule:** If two classes match exactly across datasets, they are merged into a single universal class.
2. **Subset/Superset Rule:** If one class is a subset of another, the superset is decomposed into the subset plus a remainder class.
3. **Partial Overlap Rule:** If two classes partially overlap, they are split into three disjoint classes: the intersection, and the two remainders.

The algorithm iteratively applies these rules until all classes in the multiset are disjoint, creating a flat universal taxonomy that encompasses the entire semantic range of the dataset collection.

#### OUR MODIFICATIONS

Our approach adapts their methodology for image classification with several key modifications:

- **Classification vs. Segmentation:** While Bevandic et al. focused on pixel-level prediction for semantic segmentation, our method addresses image-level classification where each image has a single label from each dataset.
- **Probabilistic Relationships:** Instead of binary relationships between classes, we derive probabilities from the cross-domain predictions of our neural networks. This allows us to not only evaluate the quantity of relationships but also their quality e.g. useability for later classification tasks.
- **Flexible Relationship Selection:** We introduce multiple methods for selecting relevant relationships from probability matrices (naive thresholding, most common predictions, density thresholding, relationship hypothesis) and try to evaluate their performance on a synthetic ground truth.
- **Simplified Taxonomy Rules:** Our taxonomy building rules are adapted for the graph structure and focus on four main cases: isolated nodes, bidirectional relationships, transitive cycles, and unilateral relationships. They are very similar to the original rules, but simplified for our use case. For example, we do not need to handle the case of partial overlaps since we do not have pixel-level predictions.

- **Multi-domain Extension:** While the Bevandic et al. algorithm was designed for two datasets (and later iteratively adding more datasets), we directly support a multi-domain setting by constructing a single graph that captures relationships across all datasets. This allows us to build a universal taxonomy that connects multiple datasets in one go, rather than iteratively merging them.

These modifications allow our algorithm to work effectively in the image classification domain while maintaining the approach of the original work for handling incompatible taxonomies across multiple datasets.

### 3.5 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.

#### 3.5.1 CONSTRUCTING ADJACENCY MATRICES

For our metrics, we first need to represent our intra-domain relationships as adjacency matrices. We concatenate 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  $A(i, i) = 1$ .

### 3 Methodology

#### 3.5.2 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:

$$\text{EDR}(G_1, G_2) = \frac{\sum_{i,j} |A_1(i,j) - A_2(i,j)|}{\sum_{i,j} \max(A_1(i,j), A_2(i,j))} \quad (3.2)$$

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 [16] as well as the Tanimoto coefficient [27] 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.

#### 3.5.3 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.

## 3.6 UNIVERSAL MODEL LEARNING

After constructing our universal taxonomy, we need to train a model that can classify images using the universal classes rather than the original domain-specific classes. This requires creating a mapping system that converts between domain classes and universal classes, and adapting our learning algorithm to work with this unified representation.

### 3.6.1 CREATION OF MAPPING MATRIX FROM TAXONOMY

To enable a neural network to learn from multiple domains simultaneously, we need to create a mapping matrix that converts domain class labels to universal class targets. This mapping is derived directly from the relationships in our universal taxonomy.

For each domain  $i$  with classes  $C_i = \{c_1^i, c_2^i, \dots, c_{k_i}^i\}$ , we construct a mapping matrix  $M_i \in \mathbb{R}^{|C_i| \times |U|}$ , where  $U$  is the set of universal classes and  $|U|$  is the number of universal classes.

Each element  $M_i(j, u)$  represents the weight of the relationship between domain class  $c_j^i$  and universal class  $u$ . Formally, this weight is defined as:

$$M_i(j, u) = \begin{cases} w & \text{if there exists a relationship } c_j^i \rightarrow u \text{ with weight } w \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

For converting domain class labels to universal class targets, we normalize each row of the matrix to ensure valid probability distributions:

$$\hat{M}_i(j, u) = \frac{M_i(j, u)}{\sum_{u' \in U} M_i(j, u')} \quad (3.4)$$

For example, if domain class  $c_1^0$  has relationships to universal classes  $u_1$  with weight 0.8 and  $u_2$  with weight 0.4, the normalized probabilities would be:

$$\begin{aligned} \hat{M}_0(1, u_1) &= \frac{0.8}{0.8 + 0.4} = \frac{2}{3} \\ \hat{M}_0(1, u_2) &= \frac{0.4}{0.8 + 0.4} = \frac{1}{3} \end{aligned}$$

### 3.6.2 LEARNING FUNCTION FOR UNIVERSAL MODEL

Our universal model architecture consists of a standard convolutional neural network (e.g. ResNet) followed by a fully connected layer that outputs logits for each universal class. The key innovation lies in the loss function and target generation process.

### 3 Methodology

#### TARGET GENERATION

For each training sample  $(x, c_j^i, i)$  where  $x$  is an image,  $c_j^i$  is the domain class label, and  $i$  is the domain identifier, we generate the universal class target using the normalized mapping matrix:

$$\mathbf{t} = \hat{\mathcal{M}}_i[j, :] \quad (3.5)$$

where  $\mathbf{t} \in [0, 1]^{|U|}$  is a probability distribution over universal classes.

#### LOSS FUNCTION

Since our targets are probability distributions rather than one-hot vectors, we define our loss using the cross-entropy formula for discrete distributions. For model predictions  $\mathbf{p} = \text{softmax}(\mathbf{z})$  where  $\mathbf{z}$  are the logits, the loss is defined as:

$$H(\mathbf{t}, \mathbf{p}) = - \sum_{u=1}^{|U|} \mathbf{t}(u) \log(\mathbf{p}(u)) \quad (3.6)$$

$$\mathcal{L}(\mathbf{t}, \mathbf{p}) = H(\mathbf{t}, \mathbf{p})$$

This loss function encourages the model to produce predictions that match the target distribution derived from our universal taxonomy.

#### INFERENCE AND EVALUATION

During inference, the model outputs a probability distribution over universal classes. To evaluate performance on the original domain classes, we need to convert these universal predictions back to domain-specific predictions using the transpose of the original (unnormalized) mapping matrix.

For universal predictions  $\mathbf{p} \in [0, 1]^{|U|}$  and domain  $i$ , the domain class predictions are computed as:

$$\mathbf{d}_i = \mathcal{M}_i^T \mathbf{p} \quad (3.7)$$

$$\hat{c}_i = \text{argmax}(\mathbf{d}_i)$$

where  $\mathbf{d}_i \in \mathbb{R}^{|C_i|}$  represents the predicted probabilities for each class in domain  $i$  and  $\hat{c}_i$  is the final predicted class label.

Note that we use the original unnormalized matrix  $\mathcal{M}_i$  (not  $\hat{\mathcal{M}}_i$ ) for this reverse mapping, as normalization would not change the final predictions and we can save computation during inference.

This approach allows our universal model to learn shared representations across domains represented by universal classes, trying to match the distribution over universal classes that lead to a target domain class.



# 4 RESULTS

## 4.1 DOMAIN MODELS

### 4.1.1 DATASETS

To start with our taxonomy generation, we first need a set of datasets that we can use to train and evaluate our domain models:

- **Caltech-101 and Caltech-256 [10, 21]**: The Caltech-101 dataset contains 101 general object categories with 40 to 800 images per category, while the Caltech-256 dataset extends this to 256 categories with at least 80 images per category. Both datasets have been widely used for image classification tasks<sup>1</sup>. The images are roughly 300x200 pixels in size and contain annotated outlines for each object in the image, which we will not need for our purposes. The dataset has no predefined train/test split, so we will use a 80/10/10 split for training, validation, and testing.
- **CIFAR-100 [19]**: The CIFAR-100 datasets contains 100 classes grouped into 20 super-classes, with 600 images per class. Each image is 32x32 pixels in size, which is significantly smaller than the Caltech datasets. The dataset is one of the most popular datasets for image classification tasks<sup>2</sup>. The dataset has a train/test split of 50000 training images and 10000 test images, which we will further split by dividing the training set into 80% for training and 20% for validation.
- **Synthetic Datasets**: To have a ground truth for our taxonomy generation, we will also create synthetic datasets based on the Caltech-101 and CIFAR-100 datasets. These datasets will be used to evaluate our cross-domain relationship graph generation methods (see Section 3.2). We will create synthetic datasets of varying sizes and complexity and evaluate how well our methods perform for different challenges.

---

<sup>1</sup>Over 500 open-access papers have cited the datasets, according to Papers with Code: <https://paperswithcode.com/dataset/caltech-101> and <https://paperswithcode.com/dataset/caltech-256>

<sup>2</sup>Over 5000 open-access papers have cited the dataset, according to Papers with Code: <https://paperswithcode.com/dataset/cifar-100>

## 4 Results

### 4.1.2 TRAINING

#### MODEL ARCHITECTURE

To now start our training of domain models, we first need to define the architecture of our models. The ResNet architecture[12, 13] is a popular choice for image classification tasks and has been shown to perform well on a variety of datasets. It also has the advantage of being pre-trained on the ImageNet dataset [5, 25], which will save us the effort of training a model from scratch. From the available ResNet architecture sizes, we decide for the leaner ResNet-50 architecture to meet our resource constraints.

We adapt the ResNet-50 architecture to our datasets by switching out the final fully connected layer with a funnel architecture that ends in an output layer matching the number of classes per dataset (see Figure 4.1).

#### TRAINING PROCEDURE

In our initial training runs, we observe severe overfitting on the training data as can be seen in Figure 4.2. To mitigate this, we apply several regularisation techniques:

- **Dropout** [14]: As can be seen in Figure 4.1, our fully connected layers contain dropout layers between them at rates of 0.5 and 0.2. Dropout is a regularisation technique that randomly sets a fraction of the input units to zero during training, which reinforces the model to learn more robust features and thereby reduces overfitting.
- **Data Augmentation:** We apply data augmentation techniques to our training data, such as random cropping, horizontal flipping, random erasing, and color jittering. These techniques artificially increase the size of our training dataset and help the model to generalise better by exposing it to a wider variety of input data.

We now train our models on the Caltech-101, Caltech-256, and CIFAR-100 datasets (i.e. their synthetic variants).

For the easier Caltech-101 and Caltech-256 datasets, we use the SGD optimiser [26] with a learning rate of 0.01, a Nesterov momentum of 0.9, and a weight decay of 0.0001 and train all variants for 50 epochs. We also use a batch size of 64 for the datasets.

For our more complex CIFAR-100 dataset, we use the AdamW optimiser [22] with an initial learning rate of 0.001, a weight decay of 0.001. We train for 100 epochs with a multistep learning rate scheduler that reduces the learning rate by a factor of 0.1 at epochs 30, 60 and 80. For the smaller CIFAR-100 images we use a batch size of 256.

The training is performed on a single NVIDIA RTX 3070 GPU with 8GB of VRAM using the PyTorch Lightning framework [7]. The training process takes approximately 5 hours for

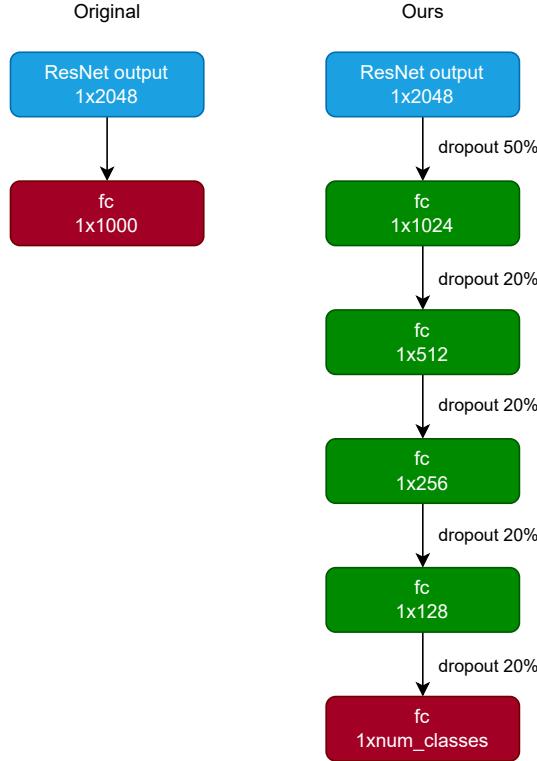


Figure 4.1: Our ResNet-50 architecture with a funnel layer for classification. Blue blocks represent the input from the ResNet-50 architecture, red blocks represent the final output layer, and green blocks represent our new funnel layers.

the Caltech-101 and Caltech-256 synthetic dataset variants and approximately 3 hours for the CIFAR-100 synthetic dataset variants.

#### 4.1.3 MODEL PERFORMANCE

##### SYNTHETIC VARIANTS

For our evaluation of relationship selection methods (see Section 4.2.1), we need synthetic dataset variants to calculate evaluation metrics on.

We select the general-purpose Caltech-256 and CIFAR-100 datasets and create synthetic variants of these datasets:

## 4 Results

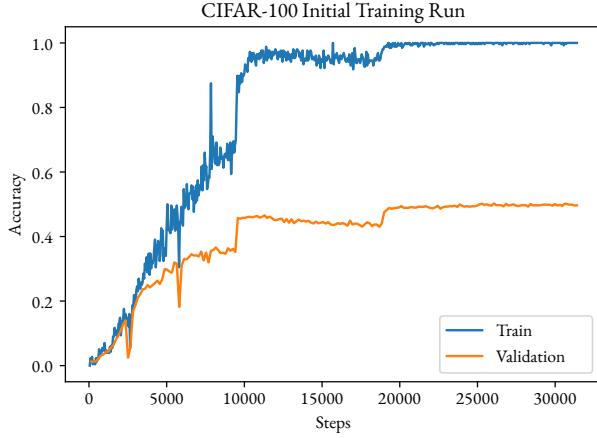


Figure 4.2: Overfitting on the CIFAR-100 dataset during training. The blue line represents the training accuracy, while the orange line represents the validation accuracy. The model overfits on the training data, resulting in a significant gap between the training and validation accuracy.

- **Caltech-256 2-Domain Variant 1:** We create a basic 2-domain variant of the Caltech-256 dataset with parameters  $\mu_{\text{concepts}} = 180$ ,  $\sigma^2_{\text{concepts}} = 10$ ,  $\mu_{\text{classes}} = 3$ , and  $\sigma^2_{\text{classes}} = 1$ . The resulting relationship graph (before applying universal taxonomy algorithms) is shown in Figure 4.3a.
- **Caltech-256 2-Domain Variant 2:** We create a simpler 2-domain variant of the Caltech-256 dataset with parameters  $\mu_{\text{concepts}} = 200$ ,  $\sigma^2_{\text{concepts}} = 10$ ,  $\mu_{\text{classes}} = 2$ , and  $\sigma^2_{\text{classes}} = 1$ . This variant has fewer concepts per class and therefore fewer relationships between the classes, which might be more similar to simple real-world datasets. The resulting relationship graph (before applying universal taxonomy algorithms) is shown in Figure 4.3b.
- **Caltech-256 3-Domain Variant:** We create a more complex 3-domain variant of the Caltech-256 dataset with parameters  $\mu_{\text{concepts}} = 180$ ,  $\sigma^2_{\text{concepts}} = 10$ ,  $\mu_{\text{classes}} = 5$ , and  $\sigma^2_{\text{classes}} = 1$ . This variant has more concepts per class and therefore more relationships between the classes, which makes it more challenging for our relationship selection methods. These extreme numbers should be seen less as a realistic dataset and more as a stress test for our methods. The resulting relationship graph (before applying universal taxonomy algorithms) is shown in Figure 4.3c.
- **CIFAR-100 2-Domain Variant:** For the CIFAR-100 dataset, our 2-domain variant has parameters  $\mu_{\text{concepts}} = 50$ ,  $\sigma^2_{\text{concepts}} = 5$ ,  $\mu_{\text{classes}} = 3$ , and  $\sigma^2_{\text{classes}} = 1$ . In the Caltech-256 dataset variants we have used approximately 70% of the classes as concepts, while in the CIFAR-100 dataset variants we use approximately 50% of the classes as concepts. This

results in a smaller, more manageable relationship graph that can be better manually inspected. The resulting relationship graph (before applying universal taxonomy algorithms) is shown in Figure 4.3d.

### MODEL ACCURACY

Let us now take a look at the accuracy of our models trained on the synthetic dataset variants. We use checkpoints to save the model after each epoch and pick the model checkpoint with the lowest validation loss for our final evaluation.

We can see our final training runs in Figure 4.4. It can be observed that our overfitting mitigation techniques have worked sufficiently well, as our training and validation accuracy curves do not diverge significantly. We present the final model accuracies on the test sets in Table 4.1. Multiple things can be observed:

- All the models achieve an accuracy of around 0.8 on the test set, which is an average performance for these datasets. Our focus is not on achieving state-of-the-art performance, but rather on creating models suitable for our cross-domain prediction task. It should be noted that a lower model accuracy will lead to worse performance in our relationship selection methods, but since we will compare the methods against each other using the same models, this should not be a problem.
- The CIFAR-100 variants have a slightly lower accuracy than the Caltech-256 variants, which is expected since the CIFAR-100 dataset has closely related classes categorised into super-classes, which make it harder for a model to distinguish between them.
- The number of concepts (i.e. classes) in the original dataset that get merged into a new class in the synthetic dataset variants does not seem to have a significant impact on the model accuracy: The Caltech-256 2-domain variant 2 has a  $\mu_{\text{classes}} = 2$ , while the Caltech-256 3-domain variant has a  $\mu_{\text{classes}} = 5$ , but the deviation in accuracy is negligible ( $\leq 0.05$ ).

Now that we have sufficiently good domain models, we can use them to evaluate our relationship selection methods and generate taxonomies from the relationship graphs we create.

## 4.2 TAXONOMY GENERATION

### 4.2.1 RELATIONSHIP SELECTION METHODS

#### SYNTHETIC DATASET VARIANTS

Now that we have trained our domain models on the synthetic dataset variants, we can use them to evaluate our different relationship selection methods.

## 4 Results

Table 4.1: Evaluation results on test sets. Models were checkpointed after every epoch and evaluated on the validation loss. The model with the lowest validation loss was selected for evaluation on the test set. Training time indicates the total duration from start to finish of model training.

Dataset Variant	Domain	Training Time	Accuracy
Caltech-256 2-Domain Variant 1	A	28m	0.83
Caltech-256 2-Domain Variant 1	B	25m	0.81
Caltech-256 2-Domain Variant 2	A	1h 12m	0.77
Caltech-256 2-Domain Variant 2	B	1h 12m	0.80
Caltech-256 3-Domain Variant	A	27m	0.84
Caltech-256 3-Domain Variant	B	26m	0.81
Caltech-256 3-Domain Variant	C	23m	0.81
CIFAR-100 2-Domain Variant	A	42m	0.77
CIFAR-100 2-Domain Variant	B	40m	0.75

A good relationship selection method needs to be versatile enough to work on a range of different datasets with different characteristics:

- The number of relationships between classes can vary significantly, depending on the number of concepts and classes in the dataset. We therefore need a method that adapts to the probability distribution of the relationships instead of picking a fixed number of relationships.
- The number of classes in the dataset can also vary significantly, which means that some datasets might have classes with few, high probability relationships, while others might have classes with many, low probability relationships. Our method needs to be able to handle both cases and not be biased towards either.

Our edge difference ratio metric (see Section 3.5.2) is a good indicator of the overall accuracy of the relationship selection methods since it measures not only the number of relationships selected, but also the correctness (i.e. the edge weights) of the relationships.

However, since every selected relationship will later result in a node in the universal model, we need to give special attention to the correctness of the relationships selected. Therefore, we will also evaluate the precision and recall of the selected relationships for each method on the synthetic dataset variants.

We will evaluate every relationship selection method on every dataset variant by selecting the method parameters that yield the best edge difference ratio on that dataset variant.

When looking at the results in Table 4.2, we can observe two things:

- The naive thresholding method consistently performs best on all dataset variants, closely followed by the relationship hypothesis method. However, the threshold parameter for

Table 4.2: Best EDR results for relationship discovery methods. For each dataset variant and method, the parameter values that yielded the lowest Edge Difference Ratio (EDR) are shown along with the corresponding F1-score.

Dataset Variant	Method	EDR	F1-score	Parameter
Caltech-256 2-Domain Variant 1	MCFP	0.670	0.526	N/A
Caltech-256 2-Domain Variant 1	Naive Thresholding	<b>0.459</b>	<b>0.798</b>	0.15
Caltech-256 2-Domain Variant 1	Density Thresholding	0.508	0.662	0.70
Caltech-256 2-Domain Variant 1	Relationship Hypothesis	0.482	0.737	5
Caltech-256 2-Domain Variant 2	MCFP	0.557	0.625	N/A
Caltech-256 2-Domain Variant 2	Naive Thresholding	<b>0.402</b>	<b>0.834</b>	0.15
Caltech-256 2-Domain Variant 2	Density Thresholding	0.451	0.668	0.70
Caltech-256 2-Domain Variant 2	Relationship Hypothesis	0.429	0.774	4
Caltech-256 3-Domain Variant	MCFP	0.707	0.443	N/A
Caltech-256 3-Domain Variant	Naive Thresholding	<b>0.377</b>	<b>0.864</b>	0.10
Caltech-256 3-Domain Variant	Density Thresholding	0.420	0.740	0.75
Caltech-256 3-Domain Variant	Relationship Hypothesis	0.391	0.830	6
CIFAR-100 2-Domain Variant	MCFP	0.634	0.636	N/A
CIFAR-100 2-Domain Variant	Naive Thresholding	<b>0.525</b>	<b>0.770</b>	0.15
CIFAR-100 2-Domain Variant	Density Thresholding	0.562	0.688	0.40
CIFAR-100 2-Domain Variant	Relationship Hypothesis	0.534	0.595	4

the naive thresholding method stays relatively consistent across the dataset variants (0.10 – 0.15), while the relationship hypothesis method has a much wider range of upper bounds (4 – 6).

- The original most common foreign prediction method from the Bevandic et al. paper [2] performs significantly worse than the other methods with an edge difference ratio of over 0.5 on all dataset variants. This was to be expected since our synthetic dataset variants rarely have classes with only a single outgoing relationship.

Let us take a deeper look at each of the methods and their performance on the synthetic dataset variants using precision-recall curves for different parameters:

- The **naive thresholding method** (see Figure 4.5) performs well on all dataset variants, achieving a high precision and recall for thresholds between 0.10 and 0.15. The displayed precision-recall curves stay consistent across the dataset variants, which indicates that the method is robust for different dataset characteristics.
- The **density thresholding method** (see Figure 4.6) performs worse than the naive thresholding method, but still achieves good performance. Similar to the naive thresholding

## 4 Results

method, the CIFAR-100 dataset variant has a slightly lower precision and recall than the Caltech-256 dataset variants, which is expected since the CIFAR-100 dataset has more closely related classes that are difficult to distinguish.

- The **relationship hypothesis method** (see Figure 4.7) performs similar to the density thresholding method, but has a conspicuous drop for the CIFAR-100 dataset variant (prominently more so than the other methods). This is likely due to the fact that this method assumes that every true relationship has an equal probability, which is not the case for the CIFAR-100 dataset variant (where some merged classes, due to having similar concepts that are hard to distinguish, have a much higher probability than others).

When using our method on real-world datasets, we do not have a ground truth to adjust the parameters to. Therefore, we need a single parameter configuration for every method that works universally across all datasets.

We create a new evaluation on the dataset variants using the parameters that yield the best edge difference ratio averaged across all dataset variants. The results are shown in Table 4.3.

The best performing naive thresholding method achieves an edge difference ratio of 0.463 with a noticeably higher recall (0.889) than precision (0.675). This means that while we hit almost 90% of the true relationships, we also select a lot of false relationships.

It can be observed that, for precision, the mcfp method performs best with a precision of 0.865, which is expected since it only selects relationships that has the highest probability. However, this comes at the cost of a very low recall of 0.421 which ultimately results in a lower F1 score compared to the naive thresholding method.

We can conclude that, if you want to create singular high-confidence relationships, the mcfp method is the best choice (especially for single-hierarchy taxonomies). However, if you want to capture as many shared concepts as possible, the naive thresholding method is the best choice, closely followed by the relationship hypothesis method (maybe as a possible alternative if the naive thresholding method does not yield good results).

However, these synthetic dataset variants might not be representative of real-world datasets, due to their shared image data as well as their “clean” relationships. But since our final goal is to create a universal model that works on multiple domains without separate head models, we can simply compare the universal model performance with our best candidate relationship selection methods (i.e. naive thresholding vs. most common foreign prediction).

### CRITIQUE ON SYNTHETIC DATASET VARIANTS

We can criticise our findings on the synthetic dataset variants by stating that the domains of the datasets are too similar and therefore do not represent a meaningful evaluation of our methods

Table 4.3: Average performance metrics for relationship discovery methods with globally optimal parameters. Each method uses the parameter value that minimizes the average EDR across all dataset variants. Performance metrics are then averaged across all dataset variants using these optimal parameters.

Method	Parameter	EDR	Precision	Recall	F1-score
MCFP	N/A	0.642	<b>0.865</b>	0.421	0.557
Naive Thresholding	0.10	<b>0.463</b>	0.675	<b>0.889</b>	<b>0.760</b>
Density Thresholding	0.70	0.502	0.514	0.876	0.636
Relationship Hypothesis	4	0.472	0.714	0.750	0.727

for real-world datasets: Since the underlying images are the same, the models have a very high accuracy, which makes methods like naive thresholding very effective - even with a low threshold (which will not be the case for real-world datasets with different domains).

To verify or disprove our findings, we will try to create new, more realistic ground truth taxonomies:

- **WordNet Synsets:** The WordNet synsets [8, 30] represents a lexicon of english words with semantic relationships to each other. This allows us to create a taxonomy using WordNet with weighted relationships by calculating the Wu & Palmer similarity score (range 0-1) between two labels (i.e. WordNet words). The score measures the semantic similarity (1 means equal, 0 means no semantic relationship).
- **SVHN-MNIST:** MNIST [6] and SVHN [23] are both labelled datasets for digits. The MNIST dataset contains greyscale images of handwritten digits, while the SVHN dataset contains colour images of house numbers. This allows us to create a ground truth mapping between the two datasets by using the labels of the digits as the classes, while the domain of the images is different.
- **Domain-Shift Synthetic Datasets:** We can try to improve our synthetic dataset variants by applying different transformations to the images of the variants (e.g. greyscale, rotation, etc.). This will change the domain of the images while keeping the classes and relationships intact and therefore create a more “difficult” dataset for our methods.

#### ALTERNATE EVALUATION METHODS - WORDNET SYNSETS

We test the usage of WordNet Synsets on the Caltech-101 and Caltech-256 datasets to build a relationship graph. It can be observed that the relationship weights (Wu & Palmer similarity scores) are very high (ranging between 0.8 and 1.0), even for weakly related synsets (e.g. between the Cal-

## 4 Results

tech256 class *snake* and the Caltech101 class *crocodile*, we observe a 0.88 relationship weight in both directions).

Upon further observation, this problem turns out to be even more severe: WordNet’s synsets build relations based on usage patterns in text corpora, which does not align with our visual relationships based on shared concepts and attributes. In Figure 4.8, we see a relationship cluster with the classes *gas pump*, *anchor* and *iris*. These words do not share any visual similarities or attributes, yet they are clustered together based on their textual relationships (we guess that *gas pump* and *anchor* are related through their usage in similar contexts like ships and maritime activities). Unfortunately upon manual inspection, these false positives occur frequently.

Since we try to build a ground truth taxonomy to evaluate relationship selection methods, these false positives can significantly skew the results and make WordNet unsuitable for our purposes.

### ALTERNATE EVALUATION METHODS - SVHN-MNIST

Next, we try to use the SVHN and MNIST datasets to create a relationship graph. These datasets provide a special usability for our purposes, since the relationships between the classes can be manually defined. Also, since the datasets have a sufficiently shifted domain (i.e. drawn, greyscale images vs. real-world images of house numbers from different angles, lighting conditions, etc.), the differences are large enough to make the task of relationship selection more difficult and realistic.

In Figure 4.9, we can see the complete relationship graph between the SVHN and MNIST datasets. Every relationship is a simple connection between two classes with a weight of 1.0.

While we could rerun our relationship selection method evaluation on this taxonomy, we decide against this method upon further reflection:

- Real-world taxonomies will not be made up of single, isolated relationships with a weight of 1.0. Complex subsets, mismatches or overlaps between classes are completely missing.
- This taxonomy will have only ten universal classes, which makes the sample size too small to draw meaningful conclusions.
- Our previous datasets contained a wide range of general-purpose classes as well as more specific ones, which allowed for a more nuanced evaluation of our methods. This taxonomy will be limited to numbers.

### ALTERNATE EVALUATION METHODS - DOMAIN-SHIFT SYNTHETIC DATASETS

The main advantage of our synthetic dataset variants is their ability to create a proven correct ground truth taxonomy. Instead of trying to derive a ground truth using new, suboptimal methods, we instead try to fix the underlying issues with our existing method.

Our synthetic variants contain the same underlying images, which result in a near-perfect, unrealistic cross-domain model prediction accuracy. Due to this high model confidence, the best performing relationship selection methods are likely to be overly optimistic and therefore perform poorly in real-world scenarios where models will not be able to perform as well across domains.

To mitigate this issue, we try to synthetically create a different domain for each variant by applying transforms to the underlying dataset images. An example subset of the domain-shifted images is shown in Figure 4.10: Using different transforms, the images appear in unique styles and distortions, synthetically changing their visual characteristics while retaining the same classes and relationships.

Table 4.4: Evaluation results on test sets for domain-shifted experiments. Models were trained with different domain shift transformations and checkpointed after every epoch. The model with the lowest validation loss was selected for evaluation on the test set. Training time indicates the total duration from start to finish of model training.

Dataset Variant	Domain	Training Time	Accuracy
Caltech-256 2-Domain Domain Shifted Variant 1	A	1h 44m	0.72
Caltech-256 2-Domain Domain Shifted Variant 1	B	2h 15m	0.76
Caltech-256 2-Domain Domain Shifted Variant 2	A	1h 51m	0.70
Caltech-256 2-Domain Domain Shifted Variant 2	B	2h 27m	0.75
Caltech-256 3-Domain Domain Shifted Variant	A	1h 42m	0.70
Caltech-256 3-Domain Domain Shifted Variant	B	2h 20m	0.75
Caltech-256 3-Domain Domain Shifted Variant	C	2h 22m	0.74
CIFAR-100 2-Domain Domain Shifted Variant	A	1h 1m	0.57
CIFAR-100 2-Domain Domain Shifted Variant	B	1h 39m	0.63

The training performance of these new domain-shifted models can be seen in Table 4.4. We can observe a 10% decrease in accuracy on the test data compared to the original synthetic dataset variants. This was to be expected due to the increased complexity of the domain-shifted tasks.

When comparing the new averaged relationship method evaluation results in Table 4.5 to our original non-domain-shifted results in Table 4.3, we can observe significant differences:

- Our new best performing relationship selection method, based on averaged maximum EDR scores, changes from MCFP to the relationship hypothesis method.

## 4 Results

Table 4.5: Average performance metrics for relationship discovery methods with globally optimal parameters on domain-shifted experiments. Each method uses the parameter value that minimizes the average EDR across all domain-shifted dataset variants. Performance metrics are then averaged across all dataset variants using these optimal parameters.

Method	Parameter	EDR	Precision	Recall	F1-score
MCFP	N/A	0.842	<b>0.490</b>	0.226	0.305
Naive Thresholding	0.10	0.761	0.418	0.519	<b>0.450</b>
Density Thresholding	0.60	0.766	0.349	<b>0.582</b>	0.426
Relationship Hypothesis	5	<b>0.759</b>	0.390	0.543	0.444

- In general, all methods have a lower EDR score than their non-domain-shifted counterparts. This is a direct result of the decreased domain model accuracy and was to be expected.
- The MCFP method still has the highest relative precision with 0.49. If precision is more important than EDR for a high universal model accuracy will be evaluated in the universal model training section (see Section 4.3).

We will use the most promising relationship selection methods, MCFP and relationship hypothesis, for our universal model training. Our final evaluation will focus on the model accuracy of the trained universal Models built with the universal taxonomies of the two preselected relationship selection methods (using the selection method parameters that have the best EDR score).

### 4.2.2 UNIVERSAL TAXONOMY GENERATION

We now have our domain models trained and our relationship selection methods evaluated. In the next step, we run our universal taxonomy generation algorithm (see Section 3.4) on the Caltech-101 and Caltech-256 datasets. To make the results more interpretable, we will use the most common foreign prediction method to have a sparse relationship graph that is easier to look at (for our later universal model training, we will of course work with both the naive thresholding method and the most common foreign prediction method).

The overall universal taxonomy generated from the Caltech-101 and Caltech-256 datasets is shown in Figure 4.11. We can observe many smaller clusters of 2-3 classes which build from a single domain class in the Caltech-101 dataset and multiple domain classes in the Caltech-256 dataset (since the Caltech-256 dataset has more granular classes, this is to be expected).

An example of a “good” cluster is shown in Figure 4.12: This cluster centers around the Caltech101 class “wheelchair” and contains multiple Caltech256 classes that share the concept of “wheel” (e.g. “mountain bike”, “segway”, “touring bike”, etc.). The universal classes between the

Caltech101 and Caltech256 classes will provide a useful mapping for our universal model training.

However, we can also observe “bad” clusters in the universal taxonomy, such as the one shown in Figure 4.13: The Caltech101 class “binocular” is connected to multiple Caltech256 classes of a variety of different concepts. While many of these classes do share concepts with binoculars (e.g. “sextant”, “telescope”, etc.), many of the Caltech256 classes in the cluster do not have an obvious connection to binoculars (e.g. “boxing glove”, “fire extinguisher”, etc.). Since it is likely that the Caltech256 dataset contains classes that do not have a suitable representation in the Caltech101 dataset, we “force” a relationship to a Caltech101 class by using the most common foreign prediction method. However, the incorrect relationships do have a low relationship weight which makes the error less severe for the universal model training.

Addressing these issues is a complex task, since a suitable threshold for the relationship weights depends on the individual composition of the datasets.

## 4.3 UNIVERSAL MODELS

### 4.3.1 TRAINING

#### MODIFICATIONS TO BASELINE ARCHITECTURE

Building upon the ResNet-50 architecture used for the individual domain models, we developed a `UniversalResNetModel` that incorporates several key modifications to enable multi-domain training through our universal taxonomy approach.

The most significant architectural change is the replacement of the standard classification head. While the baseline `ResNetModel` uses a multi-layer fully connected classifier with dropout regularization that outputs logits for domain-specific classes, the `UniversalResNetModel` employs a simplified two-layer fully connected head that outputs logits for universal classes:

- **Baseline classifier:** A 6-layer fully connected network with dropout (0.5, 0.2, 0.2, 0.2, 0.2) that progressively reduces dimensionality from ResNet features  $\rightarrow$  1024  $\rightarrow$  512  $\rightarrow$  256  $\rightarrow$  128  $\rightarrow$  domain classes
- **Universal classifier:** A 2-layer fully connected network (ResNet features  $\rightarrow$  1024  $\rightarrow$  universal classes) without dropout regularization

The simplified architecture proved to be more effective in training runs vs. the more complex hopper + dropout architecture used in the baseline models.

Another crucial modification is the loss function: The baseline models use standard cross-entropy loss with one-hot encoded targets, while the universal model employs cross-entropy for

## 4 Results

discrete probability distributions for a loss function. This change accommodates the fact that domain classes may map to multiple universal classes with different weights, as determined by the taxonomy relationships (full explanation in Section 3.6).

### MULTI-DOMAIN TRAINING PROCEDURE

Instead of training separate models on individual datasets, we create a new combined dataset that merges multiple datasets while preserving domain identity. Each training sample is augmented with a domain identifier, transforming the standard (`image, label`) pairs into (`image, (domain_id, label)`) tuples. This allows the model to handle samples from different datasets within the same batch (since we need to know the image domain to apply the correct target mapping).

During training experiments, the validation loss starts to increase again after a few epochs, indicating potential overfitting. However, the validation accuracy continues to improve, suggesting that the model is still learning useful features. To still use the best performing model checkpoint, we change checkpointing to monitor validation accuracy instead of validation loss.

This seemingly contradictory behavior of rising validation loss with rising validation accuracy can be explained by several factors:

- **Multi-target probability distributions:** Unlike standard classification with one-hot targets, our universal model uses discrete probability distributions as targets. The model may be learning to better predict the correct class ranks while becoming less confident about exact probability values, leading to higher cross-entropy loss but better top-1 accuracy.
- **Label smoothing:** The universal taxonomy creates implicit label smoothing effects where domain classes map to multiple universal classes with different weights. For perfect accuracy across all domains, it might not be possible to also hit the exact probability values, leading to higher loss *and* higher accuracy simultaneously while training.
- **Learning dynamics:** The model may be transitioning from overfitting to individual domain patterns toward learning more generalisable universal features, causing higher loss when switching between phases of learning.

As a final note on the training procedure, we rescale the CIFAR-100 images to match the size of the other datasets (224x224 pixels), so that we can process all images uniformly in the same model architecture.

Table 4.6: Baseline ResNet model performance on individual datasets. These single-domain models serve as reference points for evaluating the universal models. Every baseline model was trained for 50 epochs.

Dataset	Architecture	Optimizer	Test Accuracy
Caltech-101	ResNet-50	SGD	90.54
Caltech-256	ResNet-50	AdamW	69.54
CIFAR-100	ResNet-152	AdamW	60.48

Table 4.7: Universal model evaluation results on multi-domain test datasets. Two-domain models were trained on Caltech-101 + Caltech-256, while three-domain models were trained on all three datasets. Models were evaluated on individual domains as well as the combined test set (no weighting was applied, the individual test sets were simply concatenated). Domain accuracy values show performance differences compared to single-domain baseline models (see Table 4.6). Best results per column are shown in bold. All accuracy values are shown as percentages.

Taxonomy	Caltech-101	Caltech-256	CIFAR-100	Avg
Hypothesis (2 Domain)	91.93 (+1.39)	82.65 (+13.11)	N/A	84.62
MCFP (2 Domain)	91.00 (+0.46)	80.26 (+10.72)	N/A	82.58
MCFP Binary (2 Domain)	92.96 (+2.42)	<b>89.51 (+19.97)</b>	N/A	90.68
Hypothesis (3 Domain)	<b>95.27 (+4.73)</b>	85.88 (+16.34)	73.56 (+13.08)	77.69
MCFP (3 Domain)	82.47 (-8.07)	77.35 (+7.81)	76.10 (+15.62)	76.66
MCFP Binary (3 Domain)	95.27 (+4.73)	85.20 (+15.66)	<b>82.71 (+22.23)</b>	84.05

### 4.3.2 PERFORMANCE

On the Caltech-101 + Caltech-256 multi-domain dataset, we train universal models on the two most promising taxonomies *hypothesis* and *MCFP* (due to high learning times, we will focus on these two instead of testing every relationship selection method).

When looking at the training curves in Figure 4.14, we observe slight overfitting in the later training steps, but overall the models show stable convergence. It is interesting to note that some rapid, short fluctuations occur in the training curves which get quickly mitigated. This could indicate exploding gradients or other instabilities during training, but since they are short-lived and do not affect the overall convergence, we can conclude that the training process is robust.

We compare the performance of the universal models with baseline models (models using the same architecture and training conditions, but trained for a single domain). Our baseline model performances are shown in Table 4.6: We use the same ResNet-50 architecture that we use in the universal models (except for CIFAR-100, where we achieve an unsuitable test accuracy of below 50% when using ResNet-50, so we decided to use ResNet-152). The baseline models were also used for generating the universal taxonomies that the universal models are using.

#### *4 Results*

The final evaluation results of our universal models are shown in Table 4.7. All versions use ResNet-50 with AdamW optimizers and train for 50 epochs. We can observe multiple things:

- The universal models are able to outperform the baseline models. Our method of “pre-defining” the output layer concepts allows the model to better leverage shared knowledge across domains.
- The hypothesis relationship selection method proves superior to the MCFP method, achieving higher accuracies across all datasets except CIFAR-100 (where the MCFP method performs marginally better with 0.9% improvement).

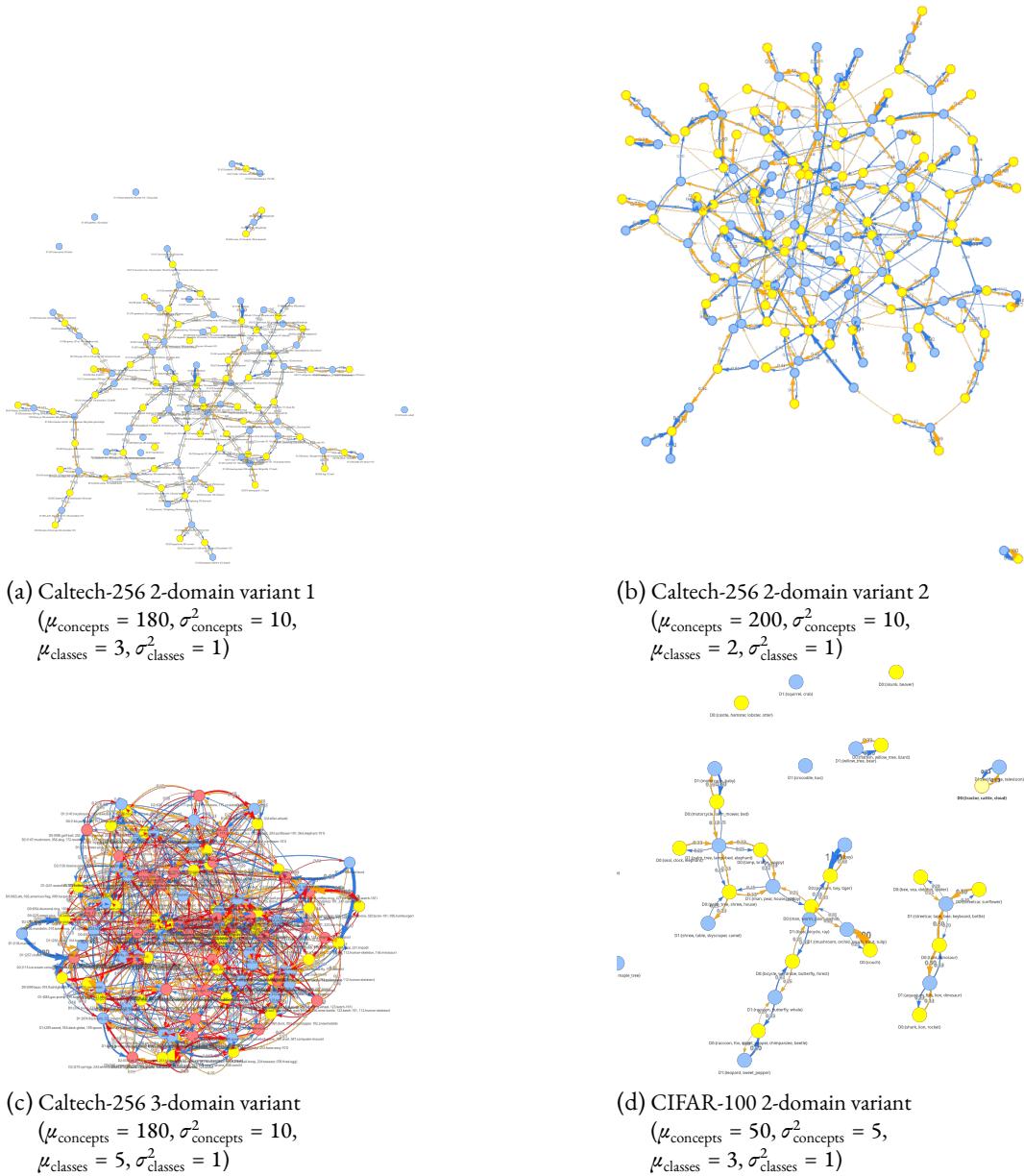


Figure 4.3: Synthetic dataset variants showing their relationship graphs before applying universal taxonomy algorithms. The number of concepts and classes per concept are sampled from truncated normal distributions with the parameters shown in each subfigure caption.

## 4 Results

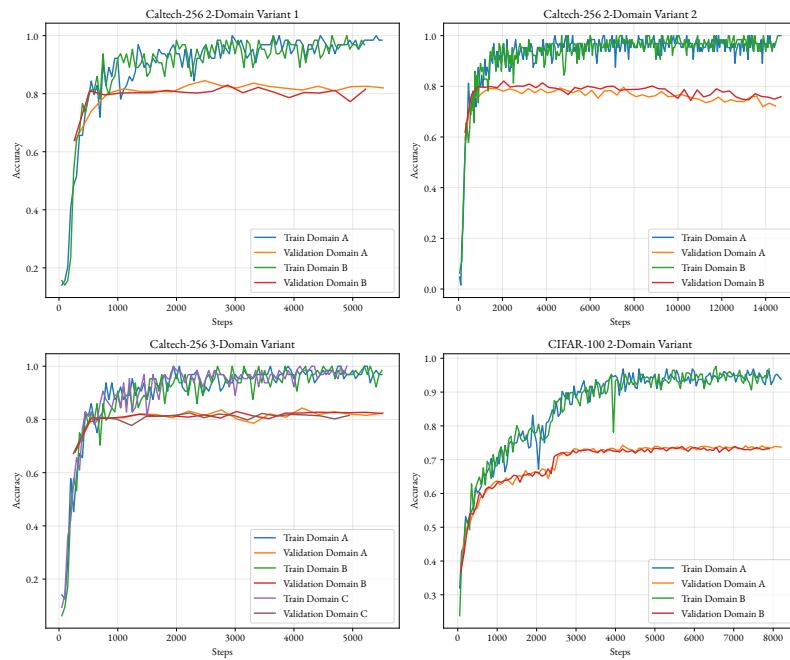


Figure 4.4: Accuracy curves for all synthetic dataset variants. Each subplot shows training and validation accuracy over training steps for all domains in that variant. The models achieve final training accuracies of approximately 0.96-0.98 and validation accuracies of approximately 0.73-0.83.

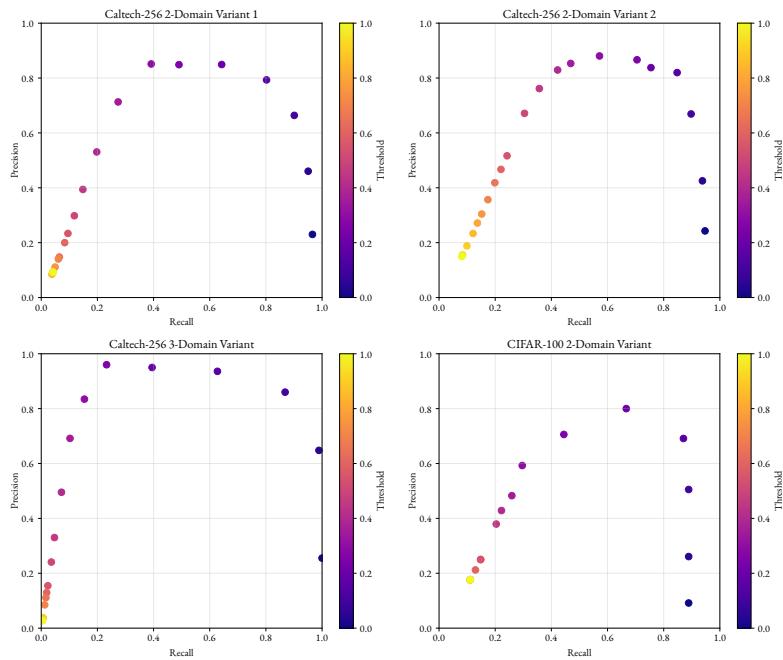


Figure 4.5: Precision and recall plot of the **naive thresholding method** for different thresholds on the Caltech-256 2-domain, Caltech-256 3-domain, and CIFAR-100 2-domain synthetic dataset variants.

## 4 Results

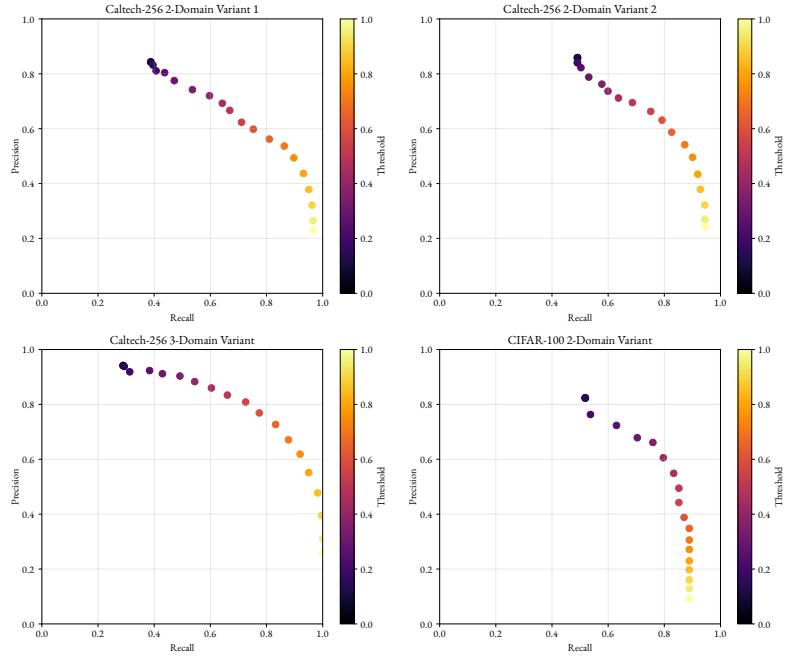


Figure 4.6: Precision and recall plot of the **density thresholding method** for different thresholds on the Caltech-256 2-domain, Caltech-256 3-domain, and CIFAR-100 2-domain synthetic dataset variants.

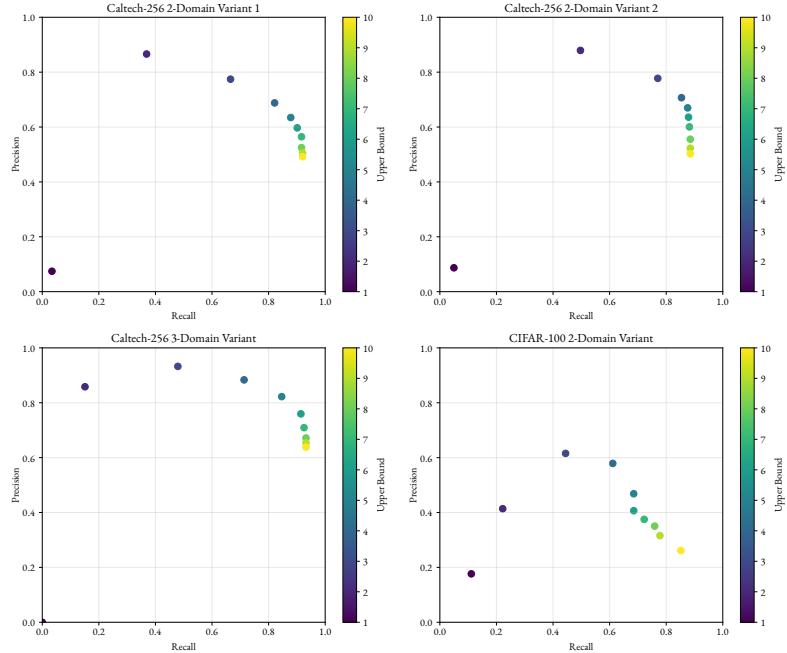


Figure 4.7: Precision and recall plot of the **hypothesis method** for different upper bounds on the Caltech-256 2-domain, Caltech-256 3-domain, and CIFAR-100 2-domain synthetic dataset variants.

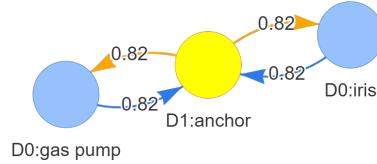


Figure 4.8: An example of a bad WordNet cluster in a relationship graph between Caltech256 and Caltech101. These false positives occur commonly and make WordNet unsuitable as a ground truth taxonomy.

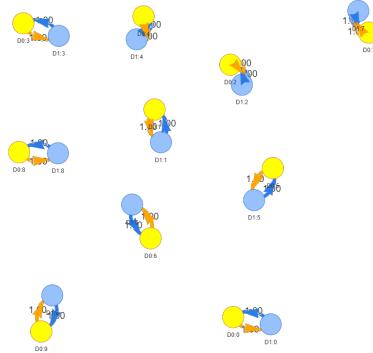


Figure 4.9: This figure shows the complete relationship graph between the SVHN and MNIST datasets. The relationships are unequivocal and straightforward, as the classes are identical across both datasets; however, they lack complexity.



Figure 4.10: Example images from the domain-shifted synthetic variants. We apply different transformations to the images of the original synthetic dataset to create two new domains. The first domain (Domain A) is a noisy greyscale variant, while the second domain (Domain B) is a rotated, blurry version of the original images, and the third domain (Domain C) has random erasings, shifted perspectives, and color jitter.

## 4 Results

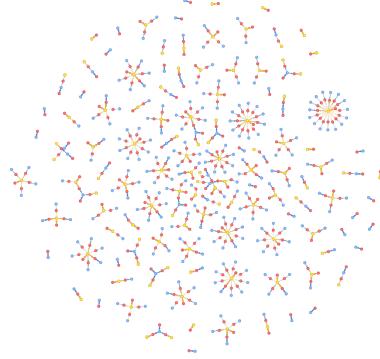


Figure 4.11: This figure shows the complete Caltech101-Caltech256 universal taxonomy. While most classes build smaller clusters of 2-3 classes, some large clusters can be observed.

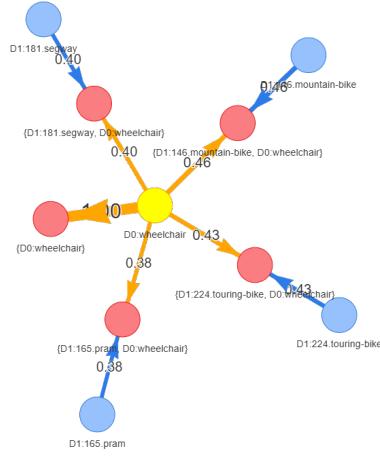


Figure 4.12: In the Caltech101-Caltech256 universal taxonomy, this cluster only contains vehicles. The universal classes created can be interpreted as sharing a common concept of “wheel”.

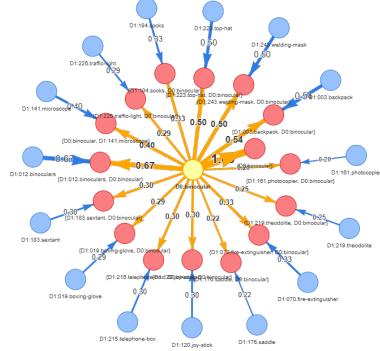


Figure 4.13: This example in the Caltech101-Caltech256 universal taxonomy shows a wrong relationship cluster towards the Caltech101 class “binocular”. Many of the Caltech256 classes do not have a suitable representation in the Caltech101 dataset and then connect to the Caltech101 class “binocular” instead.

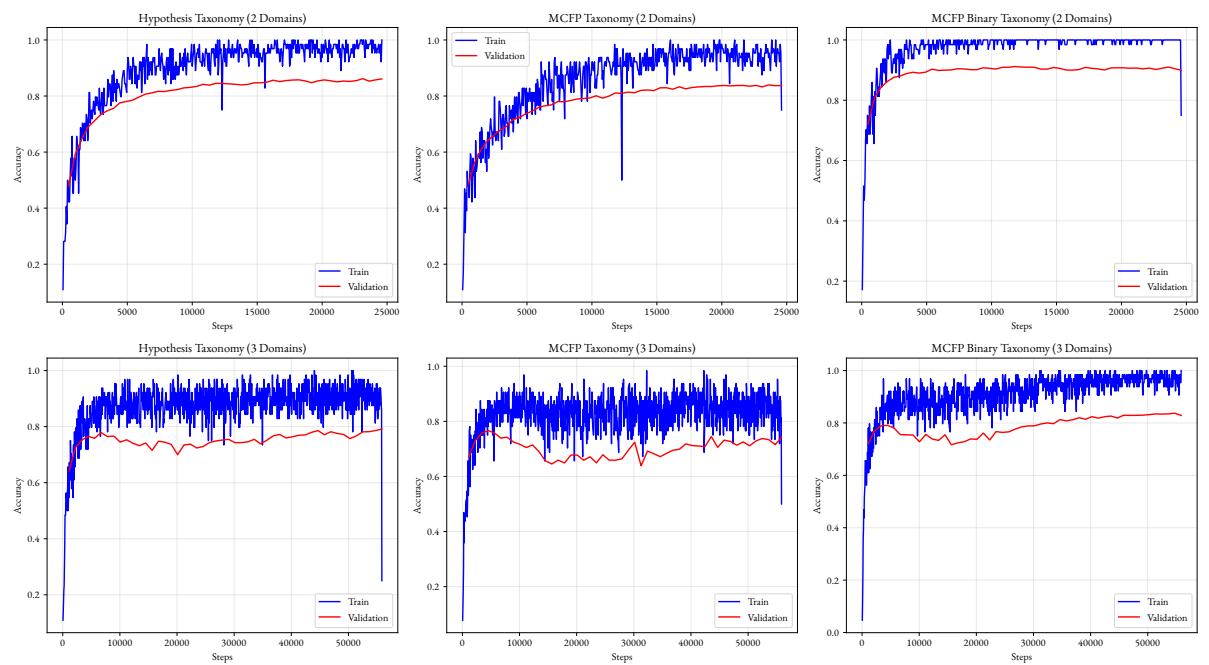


Figure 4.14: Training curves for universal models using both hypothesis and MCFP taxonomies on the multi-domain Caltech-101 + Caltech-256 dataset. Both models show stable convergence with the hypothesis taxonomy achieving slightly better performance.



# 5 DISCUSSION



# **6** CONCLUSION



## BIBLIOGRAPHY

1. P. Bevandić, M. Oršić, I. Grubišić, J. Šarić, and S. Šegvić. “Weakly supervised training of universal visual concepts for multi-domain semantic segmentation”. *International Journal of Computer Vision* 132:7, 2024, pp. 2450–2472. ISSN: 0920-5691, 1573-1405. DOI: [10.1007/s11263-024-01986-z](https://doi.org/10.1007/s11263-024-01986-z). arXiv: [2212.10340\[cs\]](https://arxiv.org/abs/2212.10340). URL: <http://arxiv.org/abs/2212.10340> (visited on 03/19/2025).
2. P. Bevandić and S. Šegvić. *Automatic universal taxonomies for multi-domain semantic segmentation*. 26, 2022. DOI: [10.48550/arXiv.2207.08445](https://doi.org/10.48550/arXiv.2207.08445). arXiv: [2207.08445\[cs\]](https://arxiv.org/abs/2207.08445). URL: <http://arxiv.org/abs/2207.08445> (visited on 04/21/2025).
3. G. Bordea, E. Lefever, and P. Buitelaar. “SemEval-2016 Task 13: Taxonomy Extraction Evaluation (TExEval-2)”. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. SemEval 2016. Ed. by S. Bethard, M. Carpuat, D. Cer, D. Jurgens, P. Nakov, and T. Zesch. Association for Computational Linguistics, San Diego, California, 2016, pp. 1081–1091. DOI: [10.18653/v1/S16-1168](https://doi.org/10.18653/v1/S16-1168). URL: <https://aclanthology.org/S16-1168/> (visited on 08/17/2025).
4. B. Chen, F. Yi, and D. Varró. *Prompting or Fine-tuning? A Comparative Study of Large Language Models for Taxonomy Construction*. 4, 2023. DOI: [10.48550/arXiv.2309.01715](https://doi.org/10.48550/arXiv.2309.01715). arXiv: [2309.01715\[cs\]](https://arxiv.org/abs/2309.01715). URL: <http://arxiv.org/abs/2309.01715> (visited on 08/17/2025).
5. 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.
6. L. Deng. “The mnist database of handwritten digit images for machine learning research”. *IEEE Signal Processing Magazine* 29:6, 2012, pp. 141–142.
7. W. Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. 2019. DOI: [10.5281/zenodo.3828935](https://doi.org/10.5281/zenodo.3828935). URL: <https://github.com/Lightning-AI/lightning>.
8. C. Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998. ISBN: 9780262272551. DOI: [10.7551/mitpress/7287.001.0001](https://doi.org/10.7551/mitpress/7287.001.0001). URL: <https://doi.org/10.7551/mitpress/7287.001.0001>.

## Bibliography

9. D. Firmani, S. Galhotra, B. Saha, and D. Srivastava. “Building Taxonomies with Triplet Queries”. In: Sistemi Evoluti per Basi di Dati. 2024. URL: <https://www.semanticscholar.org/paper/Building-Taxonomies-with-Triplet-Queries-Firmani-Galhotra-91e314dd5df505d036aa49ddeb3562551770afb6> (visited on 08/17/2025).
10. G. Griffin, A. Holub, and P. Perona. *Caltech 256*. 6, 2022. doi: [10.22002/D1.20087](https://doi.org/10.22002/D1.20087). URL: <https://data.caltech.edu/records/20087> (visited on 06/20/2025).
11. M. Gunn, D. Park, and N. Kamath. *Creating a Fine Grained Entity Type Taxonomy Using LLMs*. 19, 2024. doi: [10.48550/arXiv.2402.12557](https://doi.org/10.48550/arXiv.2402.12557). arXiv: [2402.12557\[cs\]](https://arxiv.org/abs/2402.12557). URL: [http://arxiv.org/abs/2402.12557](https://arxiv.org/abs/2402.12557) (visited on 08/17/2025).
12. K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. doi: [10.48550/ARXIV.1512.03385](https://doi.org/10.48550/ARXIV.1512.03385). URL: <https://arxiv.org/abs/1512.03385> (visited on 06/20/2025).
13. K. He, X. Zhang, S. Ren, and J. Sun. “Identity Mappings in Deep Residual Networks”. *arXiv preprint arXiv:1603.05027*, 2016.
14. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. doi: [10.48550/ARXIV.1207.0580](https://doi.org/10.48550/ARXIV.1207.0580). URL: <https://arxiv.org/abs/1207.0580> (visited on 06/21/2025).
15. 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](https://doi.org/10.48550/arXiv.1707.06642). arXiv: [1707.06642\[cs\]](https://arxiv.org/abs/1707.06642). URL: [http://arxiv.org/abs/1707.06642](https://arxiv.org/abs/1707.06642) (visited on 05/03/2025).
16. 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](https://doi.org/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).
17. D. Jurgenas and M. T. Pilehvar. “SemEval-2016 Task 14: Semantic Taxonomy Enrichment”. In: *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. SemEval 2016. Ed. by S. Bethard, M. Carpuat, D. Cer, D. Jurgenas, P. Nakov, and T. Zesch. Association for Computational Linguistics, San Diego, California, 2016, pp. 1092–1102. doi: [10.18653/v1/S16-1169](https://aclanthology.org/S16-1169). URL: <https://aclanthology.org/S16-1169> (visited on 08/17/2025).

18. P. Kargupta, N. Zhang, Y. Zhang, R. Zhang, P. Mitra, and J. Han. *TaxoAdapt: Aligning LLM-Based Multidimensional Taxonomy Construction to Evolving Research Corpora*. version: 1. 12, 2025. doi: [10.48550/arXiv.2506.10737](https://doi.org/10.48550/arXiv.2506.10737). arXiv: [2506.10737\[cs\]](https://arxiv.org/abs/2506.10737). URL: <http://arxiv.org/abs/2506.10737> (visited on 08/17/2025).
19. A. Krizhevsky and G. Hinton. “Learning multiple layers of features from tiny images”, 2009. URL: <http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf> (visited on 03/19/2025).
20. 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.
21. F.-F. Li, M. Andreetto, M. Ranzato, and P. Perona. *Caltech 101*. Version 1.0. 6, 2022. doi: [10.22002/D1.20086](https://doi.org/10.22002/D1.20086). URL: <https://data.caltech.edu/records/20086> (visited on 03/19/2025).
22. I. Loshchilov and F. Hutter. *Decoupled Weight Decay Regularization*. 2017. doi: [10.48550/ARXIV.1711.05101](https://doi.org/10.48550/ARXIV.1711.05101). URL: <https://arxiv.org/abs/1711.05101> (visited on 06/21/2025).
23. Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. “Reading Digits in Natural Images with Unsupervised Feature Learning”. In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. 2011. URL: [http://ufldl.stanford.edu/housenumbers/nips2011\\_housenumbers.pdf](http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf).
24. S. J. Pan and Q. Yang. “A Survey on Transfer Learning”. *IEEE Transactions on Knowledge and Data Engineering* 22:10, 2010, pp. 1345–1359. issn: 1558-2191. doi: [10.1109/TKDE.2009.191](https://doi.org/10.1109/TKDE.2009.191). URL: <https://ieeexplore.ieee.org/document/5288526> (visited on 08/17/2025).
25. 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](https://doi.org/10.48550/arXiv.1409.0575). arXiv: [1409.0575\[cs\]](https://arxiv.org/abs/1409.0575). URL: <http://arxiv.org/abs/1409.0575> (visited on 03/19/2025).
26. I. Sutskever, J. Martens, G. Dahl, and G. Hinton. “On the importance of initialization and momentum in deep learning”. In: *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*. ICML’13. event-place: Atlanta, GA, USA. JMLR.org, 2013, pp. III–1139–III–1147.

## Bibliography

27. T. T. Tanimoto. *An Elementary Mathematical Theory of Classification and Prediction*. Google-Books-ID: yp34HAAACAAJ. International Business Machines Corporation, 1958. 10 pp.
28. J. Uijlings, T. Mensink, and V. Ferrari. *The Missing Link: Finding label relations across datasets*. 9, 2022. doi: [10.48550/arXiv.2206.04453](https://doi.org/10.48550/arXiv.2206.04453). arXiv: [2206.04453\[cs\]](https://arxiv.org/abs/2206.04453). URL: <http://arxiv.org/abs/2206.04453> (visited on 04/23/2025).
29. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2, 2023. doi: [10.48550/arXiv.1706.03762](https://doi.org/10.48550/arXiv.1706.03762). arXiv: [1706.03762\[cs\]](https://arxiv.org/abs/1706.03762). URL: <http://arxiv.org/abs/1706.03762> (visited on 08/17/2025).
30. *WordNet*. URL: <https://wordnet.princeton.edu/homepage> (visited on 08/06/2025).
31. H. Yang, A. Willis, D. Morse, and A. de Roeck. “Literature-driven Curation for Taxonomic Name Databases”. In: *Proceedings of the Joint Workshop on NLP&LOD and SWAIE: Semantic Web, Linked Open Data and Information Extraction*. SWAIE 2013. Ed. by D. Maynard, M. van Erp, B. Davis, P. Osenova, K. Simov, G. Georgiev, and P. Nakov. INCOMA Ltd. Shoumen, BULGARIA, Hissar, Bulgaria, 2013, pp. 25–32. URL: <https://aclanthology.org/W13-5207/> (visited on 08/17/2025).
32. Y. Zhang and Q. Yang. “An overview of multi-task learning”. *National Science Review* 5:1, 1, 2018, pp. 30–43. ISSN: 2095-5138. doi: [10.1093/nsr/nwx105](https://doi.org/10.1093/nsr/nwx105). URL: <https://doi.org/10.1093/nsr/nwx105> (visited on 08/17/2025).
33. F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. *A Comprehensive Survey on Transfer Learning*. 23, 2020. doi: [10.48550/arXiv.1911.02685](https://doi.org/10.48550/arXiv.1911.02685). arXiv: [1911.02685\[cs\]](https://arxiv.org/abs/1911.02685). URL: <http://arxiv.org/abs/1911.02685> (visited on 08/17/2025).