Linköping University | Department of Electrical Engineering
Master's thesis, 30 ECTS | Datateknik
2023 | LiTH-ISY-EX--23/5538--SE

Topical classification of images in Wikipedia

– Development of topical classification models followed by a study of the visual content of Wikipedia

---

Ämneklassificering av bilder i Wikipedia – utveckling av ämneklassificeringsmodeller följd av studier av Wikipedias bilddata

Matheus V. Bernat

Supervisor : Pavlo Melnyk
Examiner : Per-Erik Forssén

External supervisors : Tiziano Piccardi (Stanford University), Miriam Redi (Wikimedia Foundation), Robert West (EPFL)

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Upphovsrätt

Copyright

Abstract

With over 53 million articles and 11 million images, Wikipedia is the greatest encyclopedia in history. The number of users is equally significant, with daily views surpassing 1 billion. Such an enormous system needs automation of tasks to make it possible for the volunteers to maintain. When it comes to textual data, there is a system based on machine learning called ORES providing automation to tasks such as article quality estimation and article topic routing. A visual counterpart system also needs to be developed to support tasks such as vandalism detection in images and for a better understanding of the visual data of Wikipedia. Researchers from the Wikimedia Foundation identified a hindrance to implementing the visual counterpart of ORES: the images of Wikipedia lack *topical* metadata. Thus, this work aims at developing a deep learning model that classifies images into a set of *topics*, which have been pre-determined in parallel work. State-of-the-art image classification models and other methods to mitigate the existing class imbalance are used. The conducted experiments show, among others, that: using the data that considers the hierarchy of labels performs better; resampling techniques are ineffective at mitigating imbalance due to the high label concurrence; sample-weighting improves metrics; and that initializing parameters as pre-trained on ImageNet rather than randomly yields better metrics. Moreover, we find interesting outlier labels that, despite having fewer samples, obtain better performance metrics, which is believed to be either due to bias from pre-training or simply more signal in the label. The distribution of the visual data predicted by the model is displayed. Finally, some qualitative examples of the model predictions to some images are presented, proving the ability of the model to find correct labels that are missing in the ground truth.

Acknowledgments

I want to thank **Tiziano Piccardi**, **Robert West**, and **Miriam Redi** for granting me the opportunity to work on this exciting Wikipedia project and around the brilliant members of EPFL's one and only Data Science Lab. A big thanks also to **Francesco Salvi**, who has worked on a parallel project providing me with the image labels I used in this thesis. I also thank **Pavlo Melnyk** and **Per-Erik Forssén** for their active supervision during the thesis work. Moreover, a heartfelt thanks to **Linköping University**, which has been so welcoming to me and has opened doors I could only have dreamt of.

Now, I extend my gratitude to all my **teachers**, **colleagues**, **friends**, and to **Béa**; thanks to you, it all has been *magnifique*. A huge thanks to my heavenly friends whose life examples and prayers gave me the boost of strength I needed at many moments: **Bridget of Sweden**, **Carlo Acutis**, **Thérèse de Lisieux**, **Clare Crocket**, **Chiara Corbella**, **Thomas Aquinas**, **Faustina Kowalska** and many more. I especially thank **mom**, **dad**, **Teti**, and the rest of the family for your unlimited love and support every step of the way; little by little, we got there! Lastly, I thank **God** for His infinite love and mercy toward me.

Contents

List of Figures

List of Tables

1 Introduction

This work introduces a topical classifier for Wikipedia images. The experimental motivation for the structure of the devoted model is presented in the following chapters, and the proposed model is then used to draw valuable insights from the visual data of the world's largest encyclopedia. All the research process, from design to implementation, was done in close collaboration with the external advisors, who are research scientists in the Wikimedia Foundation.

1.1 Motivation

Why it is important. Wikipedia is the largest encyclopedia in history, containing over 53 million articles and having around 1 billion page views per day. Such an enormous system needs automation as the number of volunteer maintainers is orders of magnitude smaller than the number of users. Thus, to ensure quality and consistency, automating tasks with algorithms is crucial. When it comes to tasks related to text, a system called ORES [15] is used to automate tasks such as vandalism detection and classification of articles.

Besides text, images play an essential role in readers' interaction with Wikipedia articles [34]. Unlike textual data, the visual data of Wikipedia lacks a system similar to ORES. And with the number of unique images on Wikipedia surpassing 11 million, organizing this visual data is becoming increasingly important.

The problem. Despite its volume and value, navigating and retrieving images from Wikipedia is hard due to their poor quality or missing metadata. With the end goal tasks in mind – finding visual knowledge gaps and visual vandalism detection – the Wikimedia foundation has decided that labeling the images in a pre-defined set of *topics* is the best way to approach the problem of unorganized and low-quality metadata.

Challenges. The challenges in developing a classification pipeline for images in Wikipedia are many. First, the ground truth labels used are only predictions drawn by traversing the *Commons category network*[1], so they are not ground truth in a strict sense; the process of obtain-

[1]Every page and image on Wikimedia Commons is linked with *Categories*, grouping together similar content. The *Commons category network* hierarchically organizes these Categories. The network should ideally be a tree but has evolved into a complex and noisy network with one common ancestor.

ing labels is further explained in Section 3.1. Second, the distribution of the images among the labels is heavily long-tailed, complicating the classification task. We present techniques to mitigate the consequences of the imbalance in Section 4. Third, the images in Wikipedia have a unique and complex distribution, unlike any other existing image datasets normally used for classification tasks, such as ImageNet [36] or COCO-stuff [5]. Some images can be seen in Figure 3.1. Finally, we want to classify the images into broader topics rather than in terms of the actual objects in them. This classification is challenging since a certain topic may englobe distinct visual objects. For instance, the topic *Belief* can be assigned to a church building, to rosary beads, or to a person in a prayerful position.

1.2 Aim

The greater purpose of the project of organizing the visual data of Wikipedia is to improve the quality and consistency of the articles on Wikipedia by better understanding the distribution of images in the platform and finding its knowledge gaps. To reach this, first, a taxonomy of topics is developed with the aim of labeling images from existing metadata such as Commons categories. This part is developed in a parallel project by Francesco Salvi [1]. Secondly, an image classification model is developed with the aim of, among others, inferring missing labels and validating the quality of the labels. This master's thesis implements the second part of the project.

1.3 Research questions

The research questions that we study and answer through this thesis are presented below:

- **RQ1. How to mitigate the consequences of having heavily imbalanced data when performing multilabel classification on Wikipedia images?**

  A long-tailed data distribution among the labels directly affects the capacity of a model to classify data correctly. In such cases, the rarer classes present significantly poorer results than the more common classes. This is due to the usual classification loss functions and metrics designed for balanced data. As the class distribution of Wikipedia images is heavily imbalanced, this is an important problem to tackle.

- **RQ2. How does a *hierarchical* topic classification of the images in Wikipedia perform compared to a *flat* classification?**

  A flat classification approach (also known as *big-bang*) is not given any knowledge of the inherent parent-child relations of the taxonomy labels, while a hierarchical approach does know the relations between the labels. In this thesis, the labels are organized in a hierarchy due to their semantical relationship. Thus, we have the hypothesis that this additional information should allow the model to perform better. Since it is desirable for the use cases of Wikipedia that the images are classified in a top-down manner, researching the effects of having a hierarchical model is highly relevant.

- **RQ3. Using the developed model, what insights can be drawn from the visual data of Wikipedia regarding its distribution and possible knowledge gaps?**

  Today, there needs to be knowledge of the topic distribution of Wikipedia's visual data. How many images are about medicine, sports, or technology is still unknown. These statistics are of great importance when Wikipedia researchers want to study how readers interact with articles and if that changes depending on the topic. Thus, applying the model to all of the data and getting first insights into the visual distribution of Wikipedia is valuable.

1.4 Summarized methods

The research methods we use to study each research question are presented below.

- **RQ1. Mitigating imbalance.** The method used in this research question is testing the techniques in an isolated manner using the same global setup. The global setup is a deep neural network – EfficientNet – either initialized with random weights or pre-trained on ImageNet. The network architecture, batch size, loss function, the number of epochs, and data separation are kept the same within the experiments. Moreover, the metrics used to compare the performance of the networks are the precision and recall scores of the common classes and the rarer classes, as commonly done in multilabel classification literature.
- **RQ2. Hierarchical classification.** The method used to answer this research question is the same as the first: implementation of a well-established model architecture followed by experiments on Wikipedia data. The training setup is kept the same, with the only difference being that, in the hierarchical case, the model is given knowledge of the label structure through a tailor-made loss function.
- **RQ3. Using the prototype to perform studies.** In this research question, a study is pursued to explain the distribution of the images in Wikipedia and find possible gaps in the visual topic distribution. The model is trained on a part of the available data and is then used to predict the topics of the training data (images from English Wikipedia).

Assumptions. Due to computational and time resources constraints, the model is only trained on images from English Wikipedia.

1.5 Contributions

The main contributions of the thesis are presented below.

- A novel empirical study of techniques to mitigate imbalance on Wikipedia images.
- Insights on the visual data of Wikipedia and the used ground truth labels.
- A prototype that can be used as a baseline for further development of tasks that automate the management of visual data of Wikipedia.

1.6 Outline

The remainder of the thesis is organized as follows:

- Chapter 2 goes through the background theory specific to hierarchical multilabel topic classification and related work used to develop the methods for this thesis.
- Chapter 3 first describes the origin of the images and their labels, followed by a description of the methodology developed based on work introduced in the previous chapter.
- Chapter 4 goes on and describes the experiments performed to answer the research questions.
- Chapter 5 uses the model and the techniques with the best performance and studies the data and the labels.
- Chapter 6 discusses the methods and the experiments.
- Chapter 7 concludes this work by summarizing the answers to the research questions and points out possibilities for future work.

2 Theory

This chapter presents information about methods, tools, and techniques used in this work, followed by an exposition of related research results.

2.1 Background

This section presents specific concepts that we use in the experiments. We assume the reader already has sufficient knowledge of artificial neural networks, activation functions, loss functions, cost-function optimization, and model selection [2] to follow along.

2.1.1 Classification: binary, multi-class and multilabel

Classification in machine learning is the task of assigning labels to the data. There are three modalities of classification problems: binary, multi-class, and multilabel [11].

In binary problems, there are two disjoint target classes, and the input data is mapped to a single one of them. Fraud and spam detection are examples of binary classification problems [37], in which, given the input features, the output is either positive or negative.

Multi-class is a generalization of the binary problem where there are more than two classes. Object classification is an example of a multi-class problem; datasets such as ImageNet [36] and MNIST [23] are standard for multi-class classification.

Finally, the input can be mapped to more than one class in the multilabel case. Image, music, and video categorization are examples of such a classification problem. In our case, we have a multilabel problem, i.e., the input images are mapped to one or more target topics.

2.1.2 Classification metrics

The metrics that we use in this thesis are the following: precision, recall, F1-score, and *AUC(PR)*. Their definitions are provided below.

First, let us define the following notions which will be reused later on:

- True positive (TP): the prediction is positive and so is the target;
- True negative (TN): the prediction is negative and so is the target;

- False positive (FP): the prediction is positive while the target is negative;
- False negative (FN): the prediction is negative while the target is positive.

Precision

This metric describes the ratio of correctly classified positive samples by the total number of positive predictions [25]. It can be calculated for class i in the following way:

$$\text{precision}_i = \frac{TP_i}{TP_i + FP_i}.$$

Recall

Recall describes the ratio of correctly classified positive samples by the total number of positive samples [25]. It can be calculated for class i in the following way:

$$\text{recall}_i = \frac{TP_i}{TP_i + FN_i}.$$

F1-score

The F1-score summarizes the precision and the recall metrics in a single number, and it is defined the following way [25]:

$$\text{F1-score}_i = \frac{2 \cdot \text{precision}_i \cdot \text{recall}_i}{\text{precision}_i + \text{recall}_i}.$$

This is an appropriate metric for problems where positive samples are scarce compared to negative ones, such as fraud or disease detection.

AUC(PR)

The area under the precision-recall curve, *AUC(PR)*, is appropriate for usage in contexts where positive samples are scarce, just as the F1-score metric. The advantage of the *AUC(PR)* metric compared to the F1-score is that it is *threshold*-independent. A threshold in classification problems is defined as the least possible probability (between 0 and 1) that a sample needs to have to be predicted as positive.

A perfect binary classifier has *AUC(PR)* equal to one, while a random classifier has an *AUC(PR)* equal to the ratio of positive samples and the total number of samples.

2.1.3 Image topic vs. semantic objects

The target labels in standard object classification tasks are physical objects in the image. On the other hand, topics are defined as a more general concept that spans a much greater spectrum of visual objects. See Figure 2.1 for examples of the differences between identifying objects and identifying topics. Note that a single topic might contain several semantic objects simultaneously and that different topics may also share visual objects. For instance, the painting and the football team images have people on them, but while one has Sports as a topic, the other one has Art as a topic.

2.1.4 Flat vs. hierarchical classification

Classification can be done in a flat or in a hierarchical way. In flat classification, the model is unaware of the parent-child relationships between the labels, which becomes a problem in real-world data where many concepts are related to or subconcepts of others. In hierarchical classification, the model is made aware of the hierarchy among the labels, either by

Figure 2.1: The objects and the topics of some images.

a customized loss function or by the model architecture itself. In this thesis, the developed hierarchical model is made aware of the hierarchy through an adjacency matrix used in a customized loss function.

An important concept in hierarchical classification is that of the *hierarchy constraint*. This means that a parent label must have a greater probability of being assigned to a sample than all of its children. When a model respects this hierarchy constraint, the model is said to output *coherent* predictions. This concept is exploited in the work by [14] as presented in Section 2.2.3.

2.1.5 Transfer learning

Transfer learning is a paradigm of using insights learned in a problem to solve a related problem [31]. A real-world example, presented in Weiss et al. [31], is of using one's knowledge of the piano to learn how to play the guitar. In the context of image classification problems, this means using a neural network pre-trained on one source dataset on another target dataset. The more similar the distribution of the target dataset is to the source dataset, the better the transferred knowledge performs on the target data.

2.1.6 Augmentation

Augmentation [2] is a data-level technique to tackle, among other things, imbalance. Augmentation is done by applying morphological operations to the available data and increasing its amount so that the model has more examples. Typical operations are shifting, rotation, scaling, and brightness change. In this project, we use the augmentations provided by TensorFlow [27] and class-dependant augmentations using Albumentations [1].

[1] `https://albumentations.ai/`

2.1.7 Binary cross-entropy loss and focal loss

Changing the loss function is an algorithm-level technique against imbalanced datasets. We test two different loss functions. Binary cross-entropy (BCE) is the standard loss function for multilabel classification problems [45, 21], while focal cross-entropy [24] successfully solves imbalance problems in binary classification settings. Let us look into it in more detail. Let the BCE loss function be $L_{BCE}(\mathbf{y}, \mathbf{p})$, where $\mathbf{y}$ is a $N \times 1$ vector with the ground truth labels for a given image, and $\mathbf{p}$ is a $N \times 1$ vector with the probabilities of an image having each of the N labels. Let also K be the number of labels. The loss function is then:

$$L_{BCE}(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} y_j \log(p_j) + (1 - y_j) \log(1 - p_j). \tag{2.1}$$

The binary focal cross-entropy loss function adds the factor $(1 - p_j)^\gamma$ to the standard cross entropy criterion [24]. Setting $\gamma > 0$ reduces the relative loss of well-classified examples (where the probability is higher than 0.6), thus letting hard examples dominate the loss function. The focal loss function has been used with success in the context of dense object detection. The definition is:

$$L_{BF}(\mathbf{y}, \mathbf{p}; \gamma) = -\sum_{j=1}^{K} y_j (1 - p_j)^\gamma \log(p_j) + (1 - y_j) p_j^\gamma \log(1 - p_j). \tag{2.2}$$

The cost function J, which is derived to update the weights in back-propagation, is defined as a mean over the losses of the images in the batch of size B:

$$J(\mathbf{Y}, \mathbf{P}) = \frac{1}{B} \sum_{i=1}^{B} L(\mathbf{y}_i, \mathbf{p}_i), \tag{2.3}$$

where $\mathbf{Y}$ and $\mathbf{P}$ are matrices with the ground truth values and probabilities of the images in the batch, respectively:

$$\mathbf{Y}_{K \times B} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_B \\ | & | & & | \end{bmatrix}, \tag{2.4}$$

$$\mathbf{P}_{K \times B} = \begin{bmatrix} | & | & & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_B \\ | & | & & | \end{bmatrix}. \tag{2.5}$$

2.1.8 Class weights

In real-world applications, the distribution of classes is often imbalanced. An algorithm-level technique to mitigate this imbalance is weighing the loss of the classes inversely proportionally to the class frequency. This way, rarer classes get greater importance in the loss function forcing the model to learn from it [10]. Using the binary cross-entropy loss function, this is as below:

$$L(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} w_j (y_j \log(p_j) + (1 - y_j) \log(1 - p_j)), \tag{2.6}$$

where K is the number of classes, p_j is the probability of the j^{th} class, y_j is the binary ground truth of the j^{th} class, and w_j is the weight applied to the loss term of the j^{th} class:

$$w_j = \frac{\text{total number of images}}{\text{number of instances of class } j \cdot \text{number of classes}}. \tag{2.7}$$

Consider the case $K = 3$, with the three classes being: A containing $1K$ images, B containing 500 images, and C containing 10 images. Then the total of images is 1510, and the weights are:

$$w_A = \frac{1510}{1000 \cdot 3} = 0.50; \quad w_B = \frac{1510}{500 \cdot 3} = 1.01; \quad w_C = \frac{1510}{10 \cdot 3} = 50.33.$$

2.1.9 Sample-weights

In certain classification problems, there is a great imbalance between the number of positive and negative labels [42]. In this case, the binary cross-entropy loss function is adapted to give a reweighting factor to the minority positive classes. The positive sample of each class is given a weight equal to the ratio between the number of negative samples and positive samples of the class across the dataset. In summary, the loss function is:

$$L(\mathbf{y}, \mathbf{p}) = -\sum_{j=1}^{K} (\alpha_j y_j \log(p_j) + (1 - y_j) \log(1 - p_j)), \tag{2.8}$$

where the weight α_j is defined as

$$\alpha_j = \frac{\text{number of negative samples of class } j}{\text{number of positive samples of class } j}. \tag{2.9}$$

2.1.10 Resampling

Under-sampling and oversampling are data-level techniques that reduce imbalance by removing instances from the most common classes or duplicating instances from the rarer classes. In a binary-classification setup, removal or duplication is done randomly, in what is known as random undersampling (RUS) and random oversampling (ROS) [19]. However, in a multilabel setting, each image may have several labels; hence, a more sophisticated solution is needed.

Charte et al. [7] introduce two algorithms, ML-RUS and ML-ROS, that perform undersampling and oversampling of a multilabel dataset. The same paper also introduces a metric to measure the imbalance in multilabel datasets, the imbalance ratio (IR) of a single label, introduced in 2.10, and the mean imbalance ratio (*meanIR*), in 2.11. In the formulas, L is the set of labels, $|D|$ is the total set of images, and $y' \in Y_i$ means that image y' has the label Y_i:

$$IR(y) = \frac{\max_{y' \in L}(\sum_{i=1}^{|D|} [[y' \in Y_i]])}{\sum_{i=1}^{|D|} [[y \in Y_i]]}, \tag{2.10}$$

$$meanIR = \frac{1}{L} \sum_{y \in L} IR(y). \tag{2.11}$$

The logic behind the IR metric is that a label's IR is the number of images the most common label contains divided by the number of images the label contains. The closer the *meanIR* is to 1, the more balanced the dataset is.

Another important metric introduced in [7] is the metric called *SCUMBLE* (*Score of ConcUrrence among iMBalanced LabEls*), ranging from 0 to 1. The hypothesis, proved in the paper, is that the lower the *SCUMBLE* of a dataset, the better the resampling algorithms work. Moreover, the paper states that resampling algorithms, such as under-sampling and oversampling, are ineffective when the *SCUMBLE* metric is well over 0.1.

2.2 Related work

This section presents related work done in the areas of machine learning research in Wikipedia, classification of imbalanced data, and hierarchical multilabel classification.

2.2.1 Wikimedia research

Researchers from the Wikimedia Foundation have already tested a couple of possible solutions to address the problem of better structuring the visual data of Wikipedia. In [35], Redi sets out to develop an image classifier trained on Commons categories. The researcher first develops a taxonomy by pairing the 6.7M Commons categories with the 160 COCO [5] categories by computing FastText [3] vectors for the Commons and COCO categories and then matching them using the Cosine distance. After that, the Inceptionv3 [40] network pre-trained on ImageNet [36] is used with two extra layers trained on the 715K target images. The results from this work are promising. Still, the obtained labels are not suitable for the current problem of identifying topics. From here, we get inspired to use a network pre-trained on ImageNet and then fine-tune it with our data.

In another work, Redi and Onyshchak [30] develops a multimodal prototype recommending images to a Wikipedia article. In more detail, the work embeds text and images in a shared space using a coordinated representation – implementing a model based on Word2VisualVec [9] – and then trains the classifier using these embeddings. The image representation is the output of the last hidden fully-connected layer of a ResNet [17] pre-trained on ImageNet, and the text representation is obtained using a combination of GRU [8], bag-of-words [16], and word2vec [29]. Note that this work is complementary to ours as it aims to solve the problem of recommending images to an article. This thesis, in contrast, aims at the more elementary task of structuring images in Wikipedia. Valuable ideas regarding multimodality and information extraction from the image title are considered during our work.

An essential part of Wikipedia content moderation is ORES [2][15], a participatory machine learning system developed by Wikimedia Foundation to help automate vandalism detection, edit removal, and article topic classification. It is *participatory* because the classifiers constantly evolve and adjust after the volunteers' feedback. Regarding topic routing, ORES models use a top-down taxonomy of 42 classes to classify the articles. The models are trained using word embeddings of the words in the articles. Since different languages do not share the same content or the exact words, the topics of the same article might differ between languages. We get inspiration from the way it automates tasks involving editing and auditing textual data and the way it classifies article topics using a decision tree.

2.2.2 Classification of imbalanced data

Effective classification of imbalanced data is still a fundamental challenge in machine learning. First, real-world data often is imbalanced, such as in cancer or fraud detection problems where relatively few samples will be positive. Second, if no special measures are taken, the model parameters will exhibit bias towards the majority class proportional to the data imbalance. When performing classification with neural networks, the techniques to combat imbalanced data are divided into data-level and algorithm-level techniques.

Data-level methods reduce the imbalance through data operations such as undersampling, oversampling, or augmentation. Estabrooks et al. [13] performs experiments using undersampling and oversampling on several imbalanced datasets to study which of those is better and at what rate. The authors concluded that one of the techniques is not always better than the other; rather, it depends on the used dataset. Perez et al. [33] applies augmentation with the same goal of mitigating the effects of class imbalance.

[2] https://www.mediawiki.org/wiki/ORES

Algorithm-level methods – also called cost-sensitive in some literature – involve techniques that change the loss function with a cost schema or modify the output to privilege rarer classes. The focal cross-entropy loss function [24], reviewed in Section 2.1.7, is a successful approach for mitigating imbalance in dense object detection. It is based on the standard binary cross-entropy loss function, but it has weights to reduce the relative loss for well-classified examples, forcing model parameters to learn more from hard examples. Another cost-sensitive technique that can be used in multilabel problems is weighting the loss of each class by the inverse class frequency – so-called class-weighting [19]. This technique allows the model to focus on rarer classes – rather than on harder examples in the focal loss case. Finally, threshold-moving in multilabel classification finds per-class thresholds by optimizing a chosen metric. The version described by Esposito et al. [12] is adapted and used in this thesis as described in Section 3.2.1.

2.2.3 Hierarchical multilabel classification

Hierarchical multilabel classification (HMC) is a generalization of the hierarchical single-label classification (HSC) problem. Both setups structure the classes in a tree-like or direct-acyclic graph structure. What differs is that HSC maps each data point to a single path, while HMC maps each data point to multiple paths in the hierarchy.

Hierarchical classification methods are divided into local (top-down) and global (one-shot). A local approach known as *local classifier per level* has one classifier per hierarchy level and is used in Cerri et al. [6] when developing HMC-LMLP, where a separated neural network for each hierarchical level is trained and whose outputs are sent to the next level. Such local methods are unscalable and incompatible with deep learning due to the high memory consumption and computation costs. Global methods consist of networks able to map objects to any paths in the tree. Vens et al. [43] introduces the well-known model CLUS-HMC, a decision tree that predicts all classes and evaluates it on functional genomics datasets. This model kept the state-of-the-art results for over ten years. Masera and Blanzieri [28] develops AMX, a block that can be connected to the output layer of a neural network whenever a hierarchy among the classes exists. AMX shows comparable results to CLUS-HMC and significantly better results than HMC-LMLP while assuming that the parent class is the union of its children.

More recently, the method C-HMCNN(h) for performing HMC by adding a block on top of an existing neural network h – similarly to AMX – is developed in Giunchiglia et al. [14]. It is made of two main blocks: one that imposes the hierarchy constraint when in prediction mode, and a loss function based on binary cross-entropy that teaches the model when to exploit the predictions in the leaf classes to update the predictions in the parent classes. It outperforms current state-of-the-art models on 20 commonly used real-work HMC benchmarks on Papers With Code [32]. C-HMCNN(h) is thus the architecture we try to replicate in this thesis when constructing an HMC model. Figure 3.7 shows the implemented network architecture.

3 Data and Methodology

In this chapter, we introduce the data and the methods used in the experiments of Chapter 4.

3.1 Data

Here, we introduce the data used in the thesis. The data, composed of images and their corresponding labels, comes from two sources. The images are taken from a dataset from Wikipedia. At the same time, the label taxonomy (i.e., the pre-defined set of labels) and the ground truth image-label pairs are developed in a parallel project by Salvi [1].

3.1.1 Images

We use images from the Wikipedia-based Image Text (WIT) Dataset [38], a multimodal multilingual dataset made of a set of 37.6M image-text instances with 11.5M unique images across 108 languages. Note that the textual component of the image-text instances is not used in the first stage, where classification is done with only the images as input. The reason for using the WIT dataset – instead of the complete set of images in Commons [44] – is that the images in Wikipedia have better quality, greater relevance, and have undergone human validation since they are part of human-validated articles.

Due to limited computational resources, only the 3.9M images of articles written in English are kept for training the developed models. Further pre-processing eliminates images with encoding problems and file formats such as *.gif* and *.webp*, leaving us with 3.1M images. See instances of the images in Figure 3.1.

3.1.2 Labels

The images in WIT are associated with their *Common categories* [1], around 11M in total. As it is uninteresting to use this many classes in an image classification context, an extra step to link these categories to a smaller set of manually picked 31 *topical* labels was necessary. The attribution of labels to the images in WIT was implemented in a parallel project by Francesco

[1] Every image in Wikimedia Commons is associated with a set of *categories*, where a *category* is a tag that gathers related pages and media.

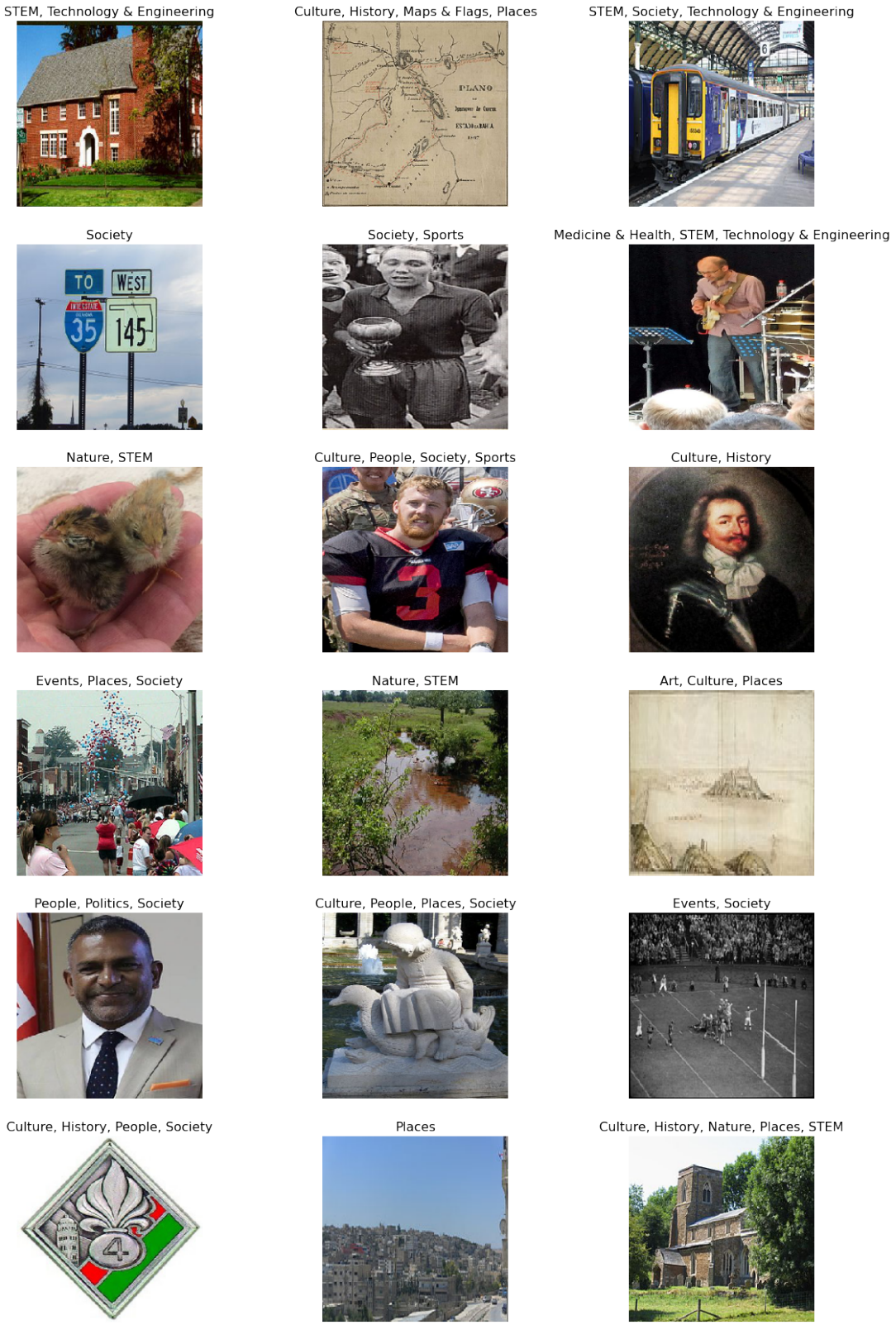


Figure 3.1: Images from WIT [38], and the ground truth labels attributed by the heuristics in Salvi [1].

Dataset	*Ground truth labels*
Flat	Maps & Flags, History
Hierarchical	Maps & Flags, History, Culture, Places

Dataset	*Ground truth labels*
Flat	Sports
Hierarchical	Sports, Society

Figure 3.2: Difference between flat and hierarchical datasets. In the first image, the parent of Maps Flags (Places) and the parent of History (Culture) are assigned to the image *a posteriori*. In the second image, the parent of Sports (Society) is also assigned to the image.

Salvi [1], under the supervision of the same supervisors from EPFL and the Wikimedia Foundation. The image-label pairs are retrieved by parsing the category network of Wikimedia Commons [44] and attributing a label to each image through a set of heuristics. Note then that these labels are not human ground truths but predictions since they change if the heuristics change.

Flat and hierarchical labels

The dataset can be either *hierarchical* or *flat* using the heuristics in Salvi [1], both having the same number of 31 topical labels. In the case where the data is *hierarchical*, the labels are organized in a tree-like hierarchy encoding the semantical links between the labels, as shown in Figure 3.3. Then, after the heuristic has traversed the Commons category network and assigned some of the pre-defined 31 labels to the image, it will *a posteriori* add the labels of the ancestors of the labels already assigned to the image. On the other hand, if the labels are *flat*, then there are no encoded semantical links between the labels, and no other labels are added after the heuristic is run.

See examples in Figure 3.2 between the flat and hierarchical datasets for the same images. Table 3.1 summarizes the metrics of the flat and hierarchical data. Finally, Figure 3.4 displays the label distribution of a randomly drawn training set of size $760K$ hierarchical data, used in the experiments of Chapter 4.

Table 3.1: Metrics of the data available for training, coming from English articles in Wikipedia. The *labels per image* metric is the average number of labels per image in the ground truth data. The *positive labels ratio* is the percentage of positive labels in the ground truth data. Note that they represent the same metric but are expressed differently: $\frac{1.77}{31} = 5.71\%$, and $\frac{2.34}{31} = 7.55\%$.

Type	# images	# labels	SCUMBLE	Labels per image	Positive labels ratio
Flat	$3.09M$	31	0.43	1.77	5.71%
Hierarchical	$3.09M$	31	0.49	2.34	7.55%

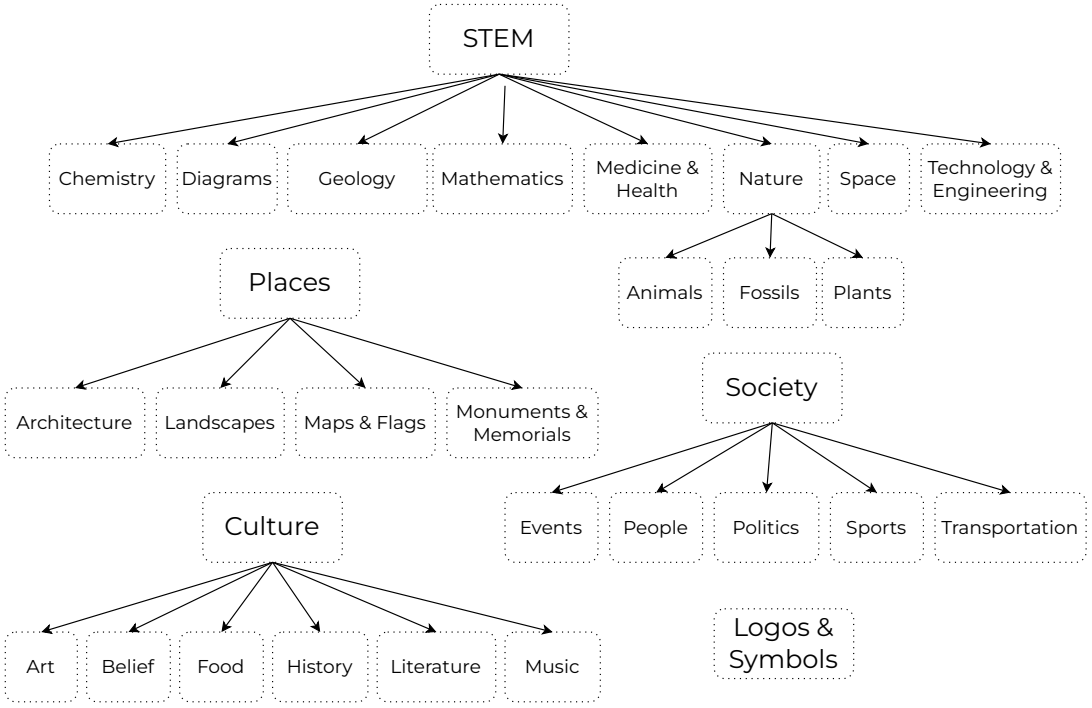


Figure 3.3: Labels organized in a hierarchy with five top categories.

3.1.3 Origins of noise

It is essential to understand the noise present in the images and the labels assigned to them. The first noise source comes from the image upload to Wikimedia Commons and the category assignment: images are uploaded by any registered user who selects the Commons category to which the images belong. The second noise source is the unstructured way users created the Commons category network; it is not a hierarchical tree decided beforehand but a graph that grew gradually into a complex entity with a common root. Finally, the third source of noise is the algorithm by Salvi to link image categories to the pre-determined set of labels; depending on the used heuristics, the labels assigned to the images – which are the labels the model uses as ground truth – will differ.

3.2 Methodology

Here, we introduce the methodology used in the experiments in Chapter 4, classified into the three research questions.

3.2.1 RQ1: Mitigating imbalance

Here we present and justify the techniques explored to mitigate the multilabel dataset imbalance.

Training

Training runs for 15 epochs, with a mini-batch size of 512 samples, and using the Adam optimizer [20] with a fixed learning rate of 0.001, which is the default in the TensorFlow implementation. The motivation for the fixed learning rate is that initial tests showed that this learning rate converged the fastest and with a stably decreasing loss. The weights of the epoch with the smallest validation loss are chosen to avoid overfitting. Furthermore, the

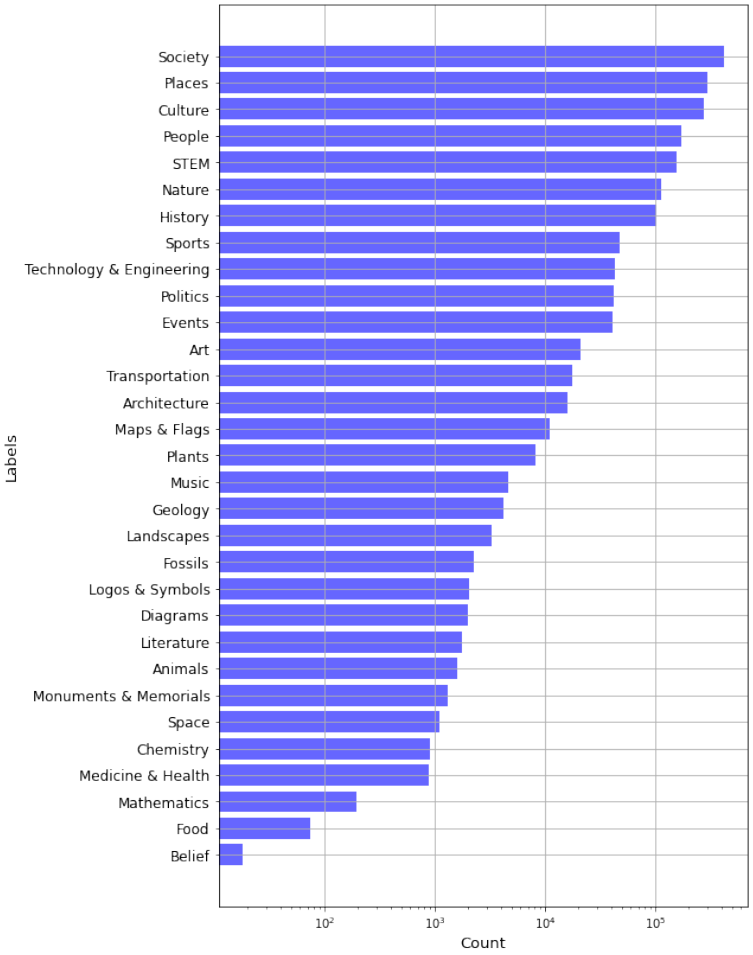


Figure 3.4: Label distribution of the randomly drawn training set of *hierarchical* data with size 760K images.

reason for only using 15 epochs is that the trained models almost always overfitted before the 15^{th} epoch; and due to the high number of experiments and limited time, the number of epochs is thus limited to 15.

Moreover, the loss function in most experiments is the unweighted average of the binary cross-entropy introduced in Section 2.1.7. Notice, however, that in Section 4.2, we experiment with adapting the binary cross-entropy loss function.

Finally, experiments within the same section used the same data split of 760K training samples, 100K validation samples for choosing the best weights, 100K validation samples for finding optimal thresholds, and a test set of 100K samples.

Performance metrics

In the experiments of Chapter 4, we use three main performance metrics: *AUC(PR)*, precision and recall over the most and least frequent classes. The *AUC(PR)* metric is suited to our use case because it is more appropriate for imbalanced data where the number of positives is overwhelmingly smaller than that of negatives; moreover, it summarizes how good the model is in one number. When it comes to precision and recall over the most and least frequent labels, we feel that these two metrics are more intuitive than F1-score. Moreover, giving separate numbers for the most and least frequent classes helps us understand how good the model is at the less common classes.

Transfer learning

Motivation. Transfer learning is desirable when there is a limited supply of training data from the target domain. Such is not our case, as we have a fairly large dataset of around $3.1M$ images.

Our use case. In our specific case, an EfficientNet [41] deep neural network is used as the base model, upon which one fully connected layer and one output layer are added, following the architecture used by Redi [35]. In the input, the network takes a scaled 64×64 image and outputs independent probabilities for each class.

See a schema in Figure 3.5. We choose EfficientNet as the base model thanks to how easy it is to scale it to a more complex network and to the proven effectiveness with which it does so. Compared to other architectures used in transfer learning, such as DenseNet [18], ResNet [17], Inception [39], and NASNet [46], it achieves better accuracy while needing orders of magnitude fewer parameters.

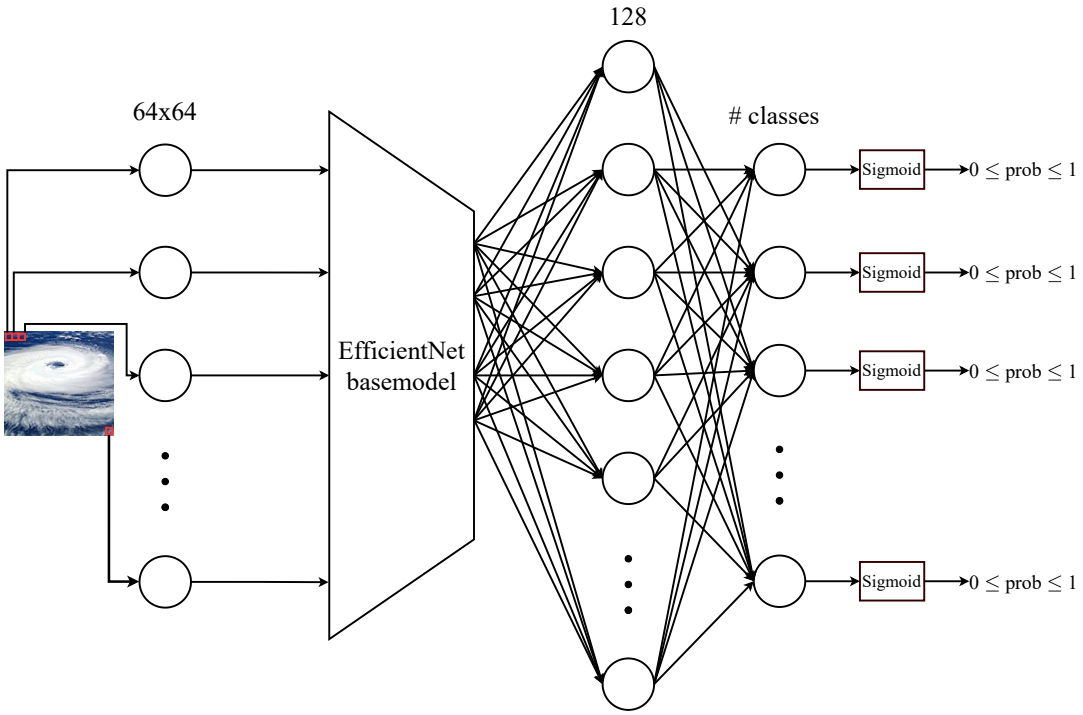


Figure 3.5: Flow chart of the used transfer learning architecture.

Undersampling

Motivation. Heuristic undersampling can reduce the *mean imbalance ratio* (*meanIR*) and improve classification results over the rarer classes.

Our use case. We want to remove images labeled with the most common labels and keep the images with rarer labels. Thus, we compute the cost of each label and define that the cost of an image equals the cost of the labels it contains. At each iteration, we remove the image with the lowest cost. This operation is repeated until a certain percentage of the images are left. See Algorithm 1 for a detailed description. Finally, to quantify the performance of this algorithm, we use the mean imbalance ratio metric (*meanIR*) introduced in Charte et al. [7].

Algorithm 1: Heuristic undersampling.

input : The relative size of the undersampled dataset, $0 < pctg < 1$, and the ground truth labels, y_{true}
output: The indices of the images in y_{true} to be removed from the original dataset

```
// Scale by a big number to avoid numerical problems, e.g., 10^4
label_costs ← BIG_NUMBER / nr_images_per_label;
image_costs ← Σ_label cost(label);
N ← 0;
indices_to_remove ← [];
while N < pctg * len(y_true) do
    N ← N + 1;
    // Find and save the index of the cheapest image
    index ← remove_cheapest_image(image_costs);
    indices_to_remove.append(index);
    // Recompute label and image costs before next iteration
    label_costs ← BIG_NUMBER / nr_images_per_label;
    image_costs ← Σ_label cost(label);
end
return indices_to_remove
```

Oversampling

Motivation. Heuristic oversampling is also a well-known method for mitigating imbalance. It can reduce the *meanIR* metric and improve classification results over the rarer classes.

Our use case. We implemented an algorithm that mirrors the undersampling one, where all images are given rewards instead based on the rewards of the labels it contains. See Algorithm 2 for a more detailed description. As in the under-sampling case, we use the *meanIR* metric to quantify how the balancing has been improved.

Algorithm 2: Heuristic oversampling.

input : The relative size of the oversampled dataset, *pctg*, and the ground truth labels y_{true}
output: The indices of the images in y_{true} to be duplicated from the original dataset, and the number of times they should be duplicated

```
// Scale by a big number to avoid numerical problems, e.g.  10^4
label_rewards ← BIG_NUMBER / nr_images_per_label;
image_rewards ← Σ_label reward(label);
N ← 0;
indices_to_duplicate ← [];
while N < pctg * len(y_true) do
    N ← N + 1;
    // Find and save index of image with greatest reward
    index ← get_best_image(image_rewards);
    indices_to_duplicate.append(index);
    // Recompute label and image rewards before next iteration
    label_rewards ← BIG_NUMBER / nr_images_per_label;
    image_rewards ← Σ_label reward(label);
end
return indices_to_duplicate
```

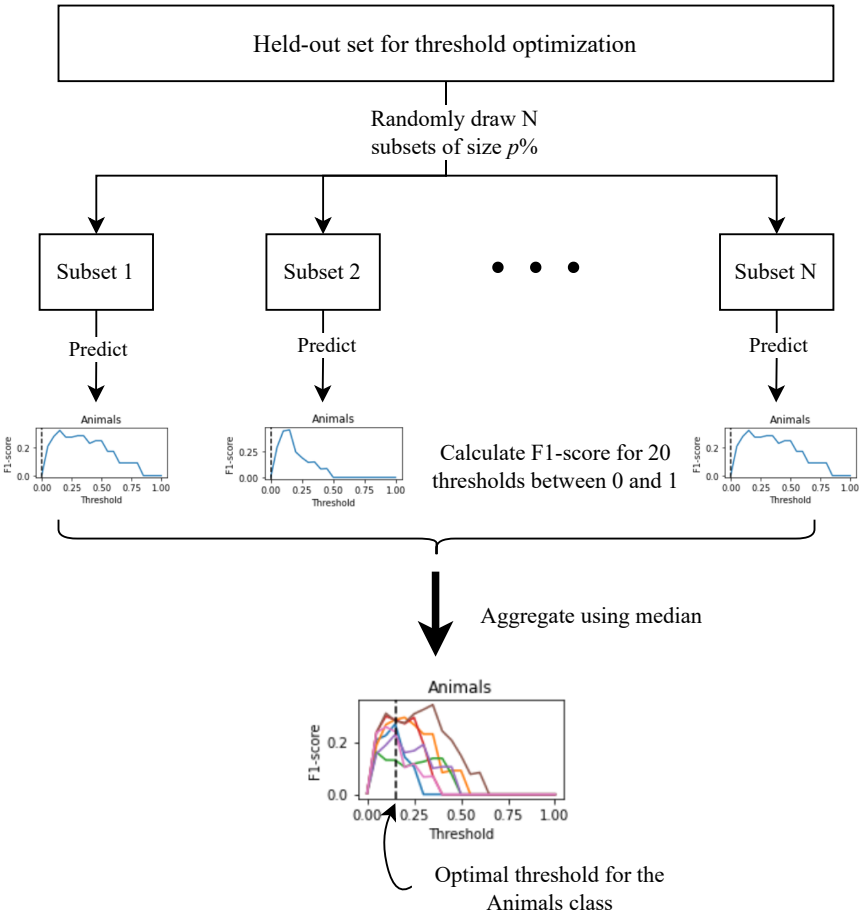


Figure 3.6: Algorithm for deciding the optimal threshold by optimizing over the F1-score. Note that this is done for all 31 classes.

Per-class thresholds

Motivation. It might seem logical to set a hard threshold of 0.5 for the probability output of a label: if a probability is greater than 0.5, then we predict that the label is in the image; otherwise, not. However, this is not adapted to imbalanced datasets, where individual thresholds per class are preferred [4].

Our use case. Thus, to determine better-suited thresholds for each class, we implement the algorithm introduced by Esposito et al. [12] with minor changes. First, the trained model is evaluated on a previously unseen validation set. Then, N sets of a specific percentage size p are randomly drawn from the validation set with replacement. For each of these N sets, the threshold that optimizes the F1-score for each class is computed. Then, the final optimal threshold for each class is defined as the median of the thresholds over the N runs. Image 4.2 shows the optimal thresholds for each of the classes; N is set to 7, and the percentage size $p = 20\%$. Figure 3.6 explains our adaptation of the algorithm by Esposito et al. [12]. The difference is that we decide the optimal threshold for each class using a held-out validation set, and not on the training set.

3.2.2 RQ2: Hierarchical classification

To answer the question of how a *hierarchical* classification model performs compared to a *flat* one (i.e. a hierarchy-agnostic one), we keep the same data and network architecture. While in

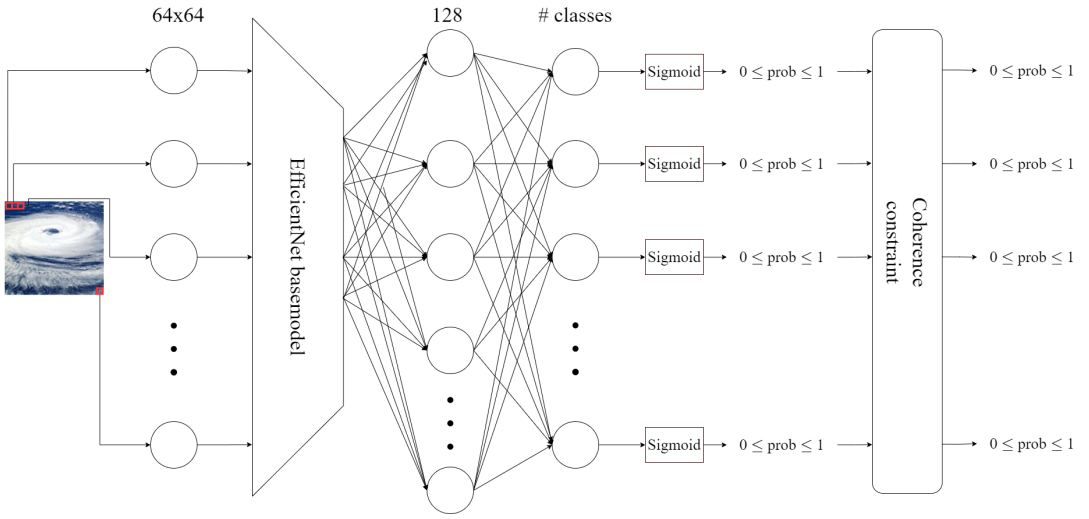


Figure 3.7: The architecture of the used hierarchical multilabel classification network.

the flat classification case, we do not use the hierarchical knowledge embedded in the labels, in the hierarchical model, we implement the coherent hierarchical multilabel classification neural network model, *C-HMCNN(h)*. This model proposed by Giunchiglia et al. [14] contains a block that ensures that the hierarchy constraint is satistified when predicting the labels and a customized loss function that takes the hierarchy into account. Figure 3.7 is a schema of the hierarchical network architecture.

Since the hierarchical model requires a single threshold for all labels, we evaluate the performance with the *AUC(PR)*) metric, usually used in HMC literature [14, 6]. Note that this is independent of the set threshold. We compare the performance of this model with the best version of the flat model.

3.2.3 RQ3: Data study

Using the best-performing model and the best-performing data we have considered in our experiments, we evaluate the model on the training data. We generate diverse plots to better understand the label distribution as predicted by the model and insights from the ground truth labels.

4 Experiments and Results

This section contains experiments performed on the WIT Dataset using the neural networks architectures of Figures 3.5 and 3.7, trained on the data gone through in Section 3.1.

4.1 *Flat* vs. *hierarchical* data

As a first experiment, we test whether using *flat* and *hierarchical* labels leads to different performances. See Section 3.1.2 for an explanation of what we mean by flat or hierarchical labels.

4.1.1 Setup and experiment

Here, we use two different models on two different datasets. As for the models, the *flat* baseline model described in Section 3.2.1, and the *hierarchical* model described in Section 3.2.2 are used. As for the data, the flat and hierarchical data described in Section 3.1.2 are used. In both models, only the last two layers of the models are set as trainable.

Training is run as described in Section 3.2.1. We use the *AUC(PR)* metric to compare the performances of the models for conciseness. Table 4.1 displays the results when evaluating the trained model on a held-out test data.

Table 4.1: *AUC(PR)* when evaluating the trained models on a held-out test data. Training is performed with flat and hierarchical models, with flat or hierarchical data.

		MODEL	
		Flat	**Hierarchical**
DATA	**Flat**	0.271	0.179
	Hierarchical	**0.303**	**0.210**

4.1.2 Conclusions

Hierarchical data yields better performance metrics than flat data. Notice that, whatever the model is (flat or hierarchical), the hierarchical data outperforms the flat data. This result is expected since the hierarchical data mitigates two problems in the flat data. First, the hierarchical data adds labels that were not present in the flat data. Moreover, since it uses the

semantics encoded in the hierarchy pre-defined by humans, it enriches the data with correct labels. Second, as seen in Table 3.1, the hierarchical data contains a greater ratio of positive labels, which is an advantage compared to the flat data.

Although error bars are not computed to show that hierarchical data leads to better performance than flat data, these results are consistent among all runs, and a 10% increase from flat to hierarchical data is significant enough to conclude so.

4.2 Data-level techniques for mitigating imbalance

Here, we describe and present the results of the experiments using the data-level techniques of under-sampling, oversampling, and augmentation for mitigating imbalance. The data used here is *hierarchical*, as Section 4.1 empirically proved that it leads to better performance.

4.2.1 Setup and experiment

The baseline classifier has the structure described in Figure 3.5, where only the final two layers are set as trainable, as is the case for the other classifiers in this experiment. In naïve augmentation, images from all labels are augmented with image transformations (zoom, rotation, shift, and horizontal flip). In under-sampling, the heuristic defined in Algorithm 1 is used, and 80% of the data is kept. In oversampling, the heuristic in Algorithm 2 is used to duplicate 20% of the images with the greatest rewards and augment them using Albumentations. The obtained results are summarized in Table 4.2. Figure 4.1 shows how the imbalance ratio when oversampling with Algorithm 2.

Table 4.2: Performance metrics when using data-level techniques for mitigating imbalance.

Classifier	***AUC(PR)***	**Precision ǀ Recall**		
		All labels	Top 5	Rest
Baseline	**0.254**	**0.408** ǀ 0.142	**0.652** ǀ 0.448	**0.362** ǀ 0.083
Naïve augmentation	0.200	0.345 ǀ 0.098	0.590 ǀ 0.276	0.280 ǀ 0.053
Under-sampling	0.233	0.367 ǀ 0.139	0.646 ǀ 0.457	0.313 ǀ 0.078
Oversampling + augmentation	0.158	0.198 ǀ **0.203**	0.488 ǀ **0.613**	0.142 ǀ **0.124**

4.2.2 Conclusions

Naïve augmentation is bad, especially on the least common labels. This is believed to be due to the data augmentations of the ImageDataGenerator class (zoom, rotation, shift, horizontal flip) being done to all images without discrimination of their labels. As the distribution of the labels is unbalanced, these augmentations will further increase the data imbalance, resulting in the model being worse at recognizing the least common labels.

Undersampling did not mitigate imbalance. The metrics obtained by undersampling and keeping 80% of the data do not show a significant difference compared to the basic setup with all data, although the metrics are systematically lower than in the baseline setup. On the other hand, this technique does not improve the metrics of the less common classes. Thus we conclude it does not effectively mitigate imbalance in this setup with this data.

Oversampling followed by augmentation decreased precision significantly. Find in Table 4.2 that oversampling followed by augmentation of the duplicated data decreases the average precision of all labels by a factor of two. Despite the oversampling algorithm being successful at reducing the *meanIR* metric, as shown in Figure 4.1, the overall performance worsens. A systematically greater recall compensates this lousy result compared to the basic setup. All in all, the quality of the model predictions worsens when applying this technique,

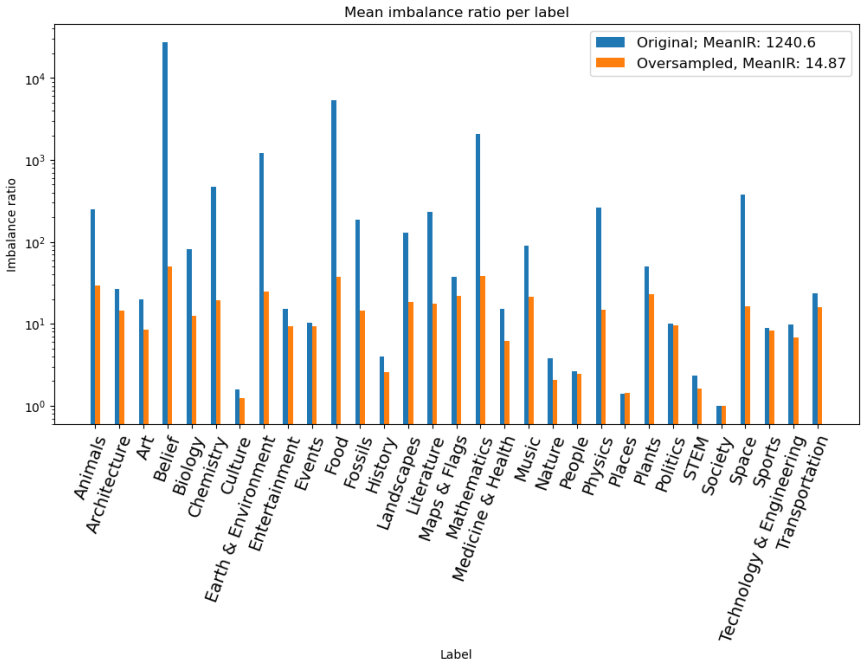


Figure 4.1: Imbalance ratios for all labels when oversampling the data with the heuristics in Algorithm 2. Notice the drastic decrease in the *meanIR*.

as can be seen when analyzing its average *AUC(PR)* value. The worse performance compared to the baseline is rather surprising as much of the imbalance classification literature [19] claims that oversampling has at least equal performance to the baseline model.

The failure of the resampling algorithms (under-sampling and oversampling) in mitigating the effects of data imbalance is explained by the high label concurrence present in the label, which confirms the hypothesis proved in Charte et al. [7]. As shown in Table 3.1, the *SCUMBLE* metric in the data is 0.49, well above 0.1, the limit at which, according to the cited paper, resampling algorithms would stop being effective at mitigating imbalance.

4.3 Algorithm-level techniques for mitigating imbalance

Here, we describe the experiments using methods on the level of the algorithm, leaving the data unchanged. The methods we consider are the focal loss function, individual thresholds per class, class-weighting, and sample-weighting. The data used is *hierarchical*, as Section 4.1 empirically proved that it leads to better performance metrics.

4.3.1 Setup and experiment

The same baseline classifier as in the data-level techniques is also used here. Focal loss is implemented according to Section 2.1.7 using $\gamma = 2$, per-class thresholds according to Section 3.2.1, class-weighting according to Section 2.1.8, and sample-weighting according to Section 2.1.9. Figure 4.2 shows how the optimal thresholds are chosen to be the median over seven different runs for 20 classes. The results are summarized in Table 4.3.

4.3.2 Conclusions

Per-class thresholds increase recall substantially. As we can see from Table 4.3, per-class thresholds significantly improve the recall for all labels, for the top 5 labels, and for the rest

Table 4.3: Performance metrics when using algorithm-level techniques for mitigating imbalance.

Loss	***AUC(PR)***	**Precision ǀ Recall**		
		All labels	Top 5	Rest
Baseline	0.254	0.408 ǀ 0.142	0.668 ǀ 0.487	0.358 ǀ 0.076
Per-class thresholds	0.254	0.233 ǀ **0.527**	0.488 ǀ **0.800**	0.184 ǀ **0.474**
Focal loss	0.252	0.373 ǀ 0.144	0.668 ǀ 0.489	0.316 ǀ 0.078
Class-weights	0.208	0.267 ǀ 0.185	0.556 ǀ 0.486	0.211 ǀ 0.127
Sample-weights	**0.263**	**0.420** ǀ 0.156	**0.674** ǀ 0.517	**0.371** ǀ 0.087

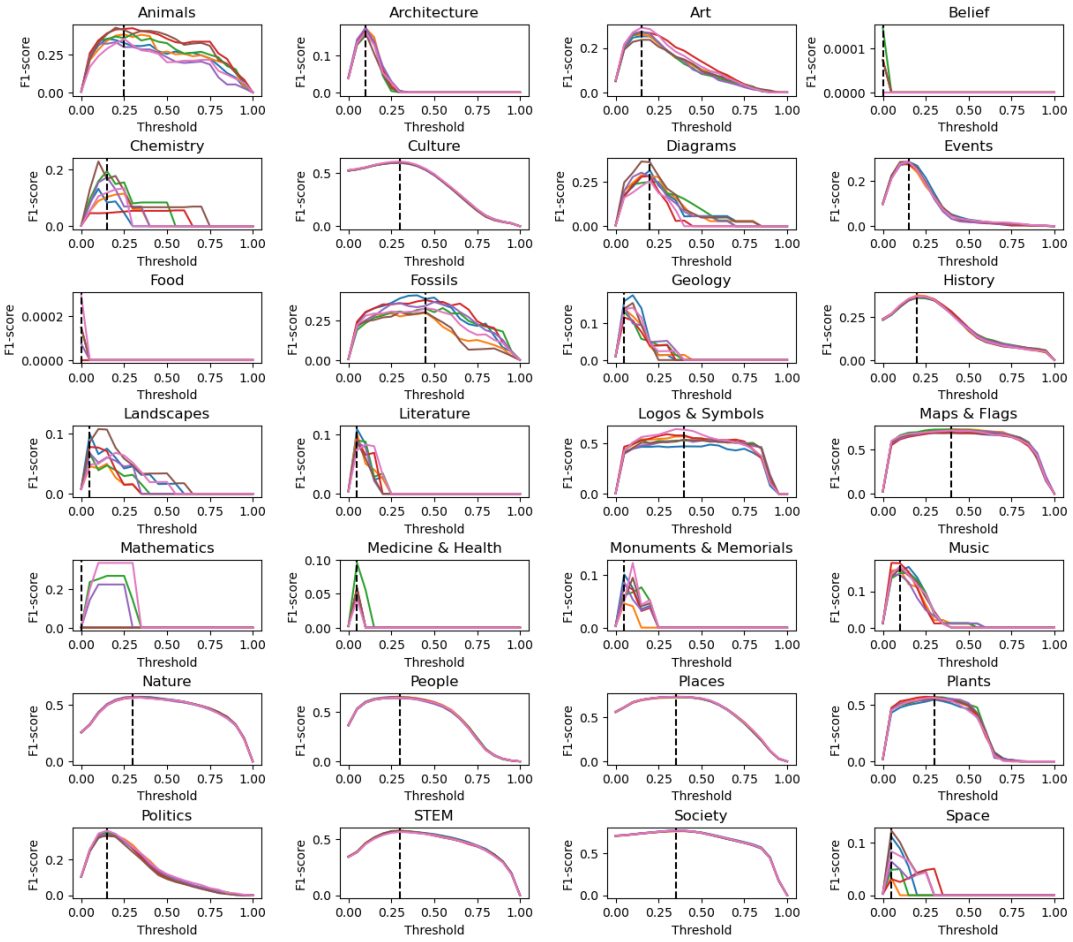


Figure 4.2: The per-label optimal threshold is taken as the median over seven runs with randomly chosen images optimizing the F1-score.

labels. Also, the F1-scores are increased, as shown in Figure 4.4. The number of labels per image increases to 10.322 (not displayed in the table), with the average in the ground truth being 2.396, which explains the increase in recall.

Focal loss does not hurt but does not help either. The results obtained by changing the standard binary cross-entropy loss by the focal loss are strikingly similar. The desired result of mitigating class imbalance by improving the model performance on the less common classes is not obtained. Thus, we conclude that the binary cross-entropy loss is preferred since it is more widely used in multilabel classification problems.

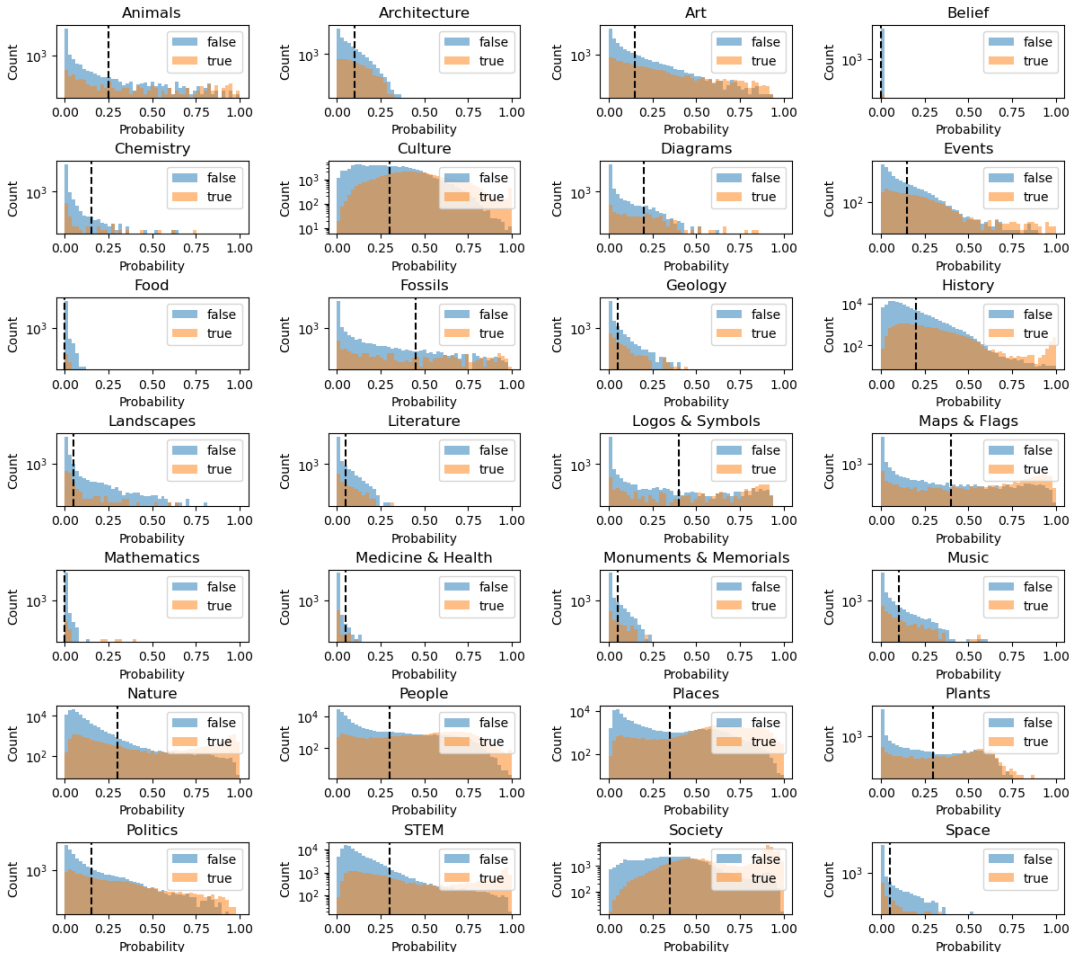


Figure 4.3: The graph shows the count of samples to which the model gives probabilities between 0 and 1. The blue part represents the ground truth positive, and the orange represents the ground truth negative. Also, the dashed line represents the optimal threshold. TP: orange curve to the right of the threshold; TN: blue curve to the left of the threshold; FN: orange curve to the left of the threshold; FP: blue curve to the right of the threshold.

Class-weights improves the metrics of the less common classes slightly but worsens overall metrics. Despite getting a decrease in the *AUC(PR)* metric, the class weight technique provides a 67% increase in the recall of the Rest classes compared to the basic setup. All in all, this technique mitigates the class imbalance by improving the model recall on the less common classes, but with an overall loss of performance on the rest.

Sample-weights improve overall performance. Of all of the techniques used to mitigate the consequences of the data imbalance, giving different weights to positive and negative samples in the loss function, as explained in Section 2.1.9, gave the best results compared to the basic setup. Notice that the *AUC(PR)* increases slightly by one percent. This result is expected since the negative samples heavily dominate the data – as Table 3.1 shows, only 7.55% of the labels are positive.

The label separation is still bad. In an ideal case, we want the ground truth true values to be above the threshold and the false ones below it. In our case, as we can see from Figure 4.3, the network can still not separate well between true and false label predictions. So, even though the per-class threshold increases the F1-score of the classes (metric not shown in the table), we still need better separability of the classes through better labels and a better classifier.

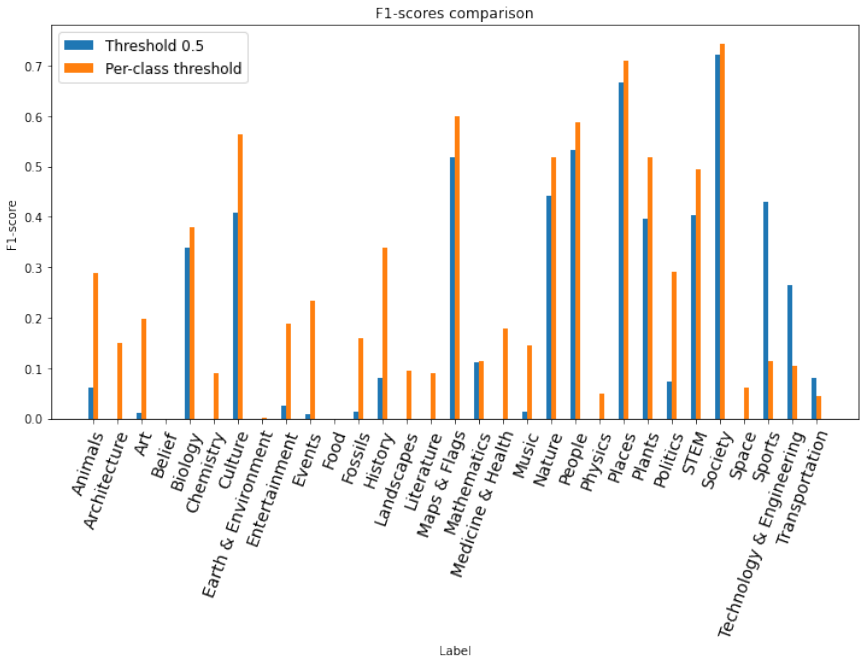


Figure 4.4: Per-class F1-scores before and after threshold moving.

4.4 Weight initialization and fine-tuning for mitigating imbalance

In addition to the data and algorithm-level methods, we also perform two other experiments that do not fit this classification. First, we compared the classification results when initializing the network weights randomly or with weights pre-trained on ImageNet. Second, we study the influence of fine-tuning the network not only on the last two trainable layers but also on more layers of the base model, something cited by Redi [35] as future work.

4.4.1 Setup and experiment: weight initialization

Given the relatively large amount of available data – $3.1M$ images from English Wikipedia – transfer learning is not a requirement. Also, since the source distribution of the pre-training data (ImageNet) is different from that of the target data (Wikipedia images), transfer learning does not seem necessary or desirable. The hypotheses we wanted to test empirically are: first, whether the classification metrics for random initialization are comparable with the case where the weights are taken as pre-trained from ImageNet; second, whether any biases from pre-training on ImageNet exist.

The network is trained on $760K$ images, using $95K$ validation images for deciding the best weights. It is then evaluated on $50K$ images. Training is performed for 30 epochs; however, we keep the model weights of the epoch where the validation loss is the least. Except for the initialization of the network weights – randomly initialized (normal random) or initialized with weights from pre-training on ImageNet –, the training setup is the same in both cases. All layers of the base model are trainable. See the compiled results in Table 4.4 and in Figures 4.5 and 4.6.

4.4.2 Conclusions: weight initialization

Initializing with ImageNet yields better performance, but random weight initialization seems to have the potential to surpass it. Starting the training of the network with weights that have been previously trained on ImageNet data proves to give better performance for

Table 4.4: Average F1-scores for different initialization of weights.

Initial weights	**F1-score**		
	All labels	Top 5	Rest
ImageNet	**0.267**	**0.450**	**0.206**
Random	0.192	0.415	0.118

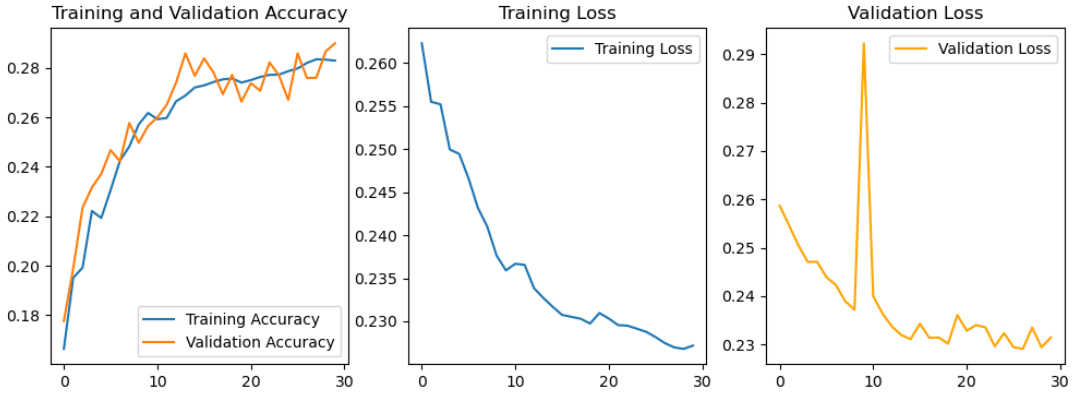


Figure 4.5: Training metrics when weights are initialized as random. The validation loss has a decreasing trend.

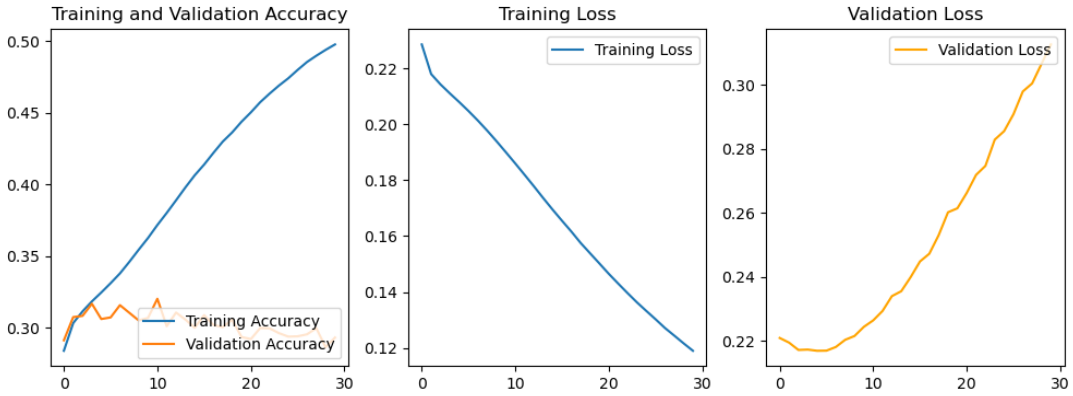


Figure 4.6: Training metrics when weights are initialized after pre-training on ImageNet. Overfitting is already present after epoch 5.

our training setup. On the other hand, by analyzing the training metrics and the numerical results, it seems as if randomly initializing the weights has the potential to yield better results if initialization and training are done in a more adapted way and for more epochs. See in Figure 4.5 that the validation loss has a decreasing trend during all epochs but one. Also, note that the network initialized with pre-trained weights presents early overfitting already around epoch 5, as displayed in Figure 4.6.

There does not seem to be a significative inductive bias. Similar results are obtained for certain classes no matter if the weights are initialized as pre-trained on ImageNet or randomly. This hints that the better metrics achieved by the model is not due to the pre-training on ImageNet. Instead, other factors, such as the number of training samples or the visual coherence of a label, seem to be the actual reasons behind the above-average metrics.

4.4.3 Setup and experiment: fine-tuning of the layers of the base model

In this experiment, we have one hypothesis to test: whether the classification metrics would improve the more layers we train.

The network is trained on 760K images, where 5% of these are the validation set used for early stopping and evaluated on 40K images. Training runs for a maximum of 20 epochs, with a batch size of 512. The weights from the epoch with the lowest validation loss are used. The metrics used for the comparison are the average F1-score and the average number of assigned labels per image.

Table 4.5: Effects of fine-tuning the 339-layers base model.

Trainable layers	**Trainable params**	**Labels per image**	**F1-scores**		
			All labels	Top 5	Rest
0	723,604	0.922	0.186	0.386	0.119
3	1,222,036	0.976	0.209	0.414	0.141
10	2,340,076	0.912	0.195	0.399	0.127
70	5,722,684	1.064	0.241	0.441	0.175
100	6,900,894	1.079	0.242	0.438	0.176
200	8,187,424	1.132	0.252	0.450	0.186
339	8,424,598	**1.165**	**0.262**	**0.469**	**0.193**

4.4.4 Conclusions: fine-tuning of the layers of the base model

Fine-tuning more layers is indeed good. There is indeed a significant increase in the average F1-scores when fine-tuning more of the base model's final layers. We believe this is due to an increase in trainable parameters and, thus, to better adaptation to the data.

4.5 Hierarchical multilabel classification

Here, we compare the metrics achieved by a flat classification model to a hierarchical one, which is given the extra information on how the labels are structured as described in Figure 3.3 in the form of an adjacency matrix. The original implementation of the hierarchical model is written in PyTorch; hence we need to rewrite the original implementation to TensorFlow, the deep learning library used in the thesis.

4.5.1 Setup and experiment

The hypothesis we have, and want to test empirically, is that the hierarchical model performs better as it knows the label hierarchy. The network architectures are presented in Figures 3.5 and 3.7. All of the parameters are set as trainable.

Table 4.6: Flat vs. hierarchical models.

Model	***AUC(PR)***	**Precision / Recall**		
		All labels	Top 5	Rest
Flat	**0.310**	**0.518** | **0.190**	0.686 | **0.535**	**0.486** | **0.124**
Hierarchical	0.237	0.401 | 0.132	**0.723** | 0.421	0.339 | 0.076

4.5.2 Conclusions

Our initial hypothesis that this hierarchical model would perform better than the flat model is wrong, as shown by Table 4.5.1. Despite having the extra information on how the labels are

structured inside the loss function, this model architecture did not achieve better general results, as shown in precision and recall, and summarized by the *AUC(PR)* metric. Surprisingly the hierarchical model presented a performance deterioration on the less common labels.

The reason the state-of-the-art model on HMC [32] did not perform better than the flat model is believed that this hierarchical model is not adapted for large image datasets such as WIT. The original paper by Giunchiglia et al. [14] experiments with two-layer fully connected perceptrons and cites the use of CNNs as a base model for future work, which we do in this thesis.

5 Data study

In this chapter, we use one of the developed models to understand the visual data of Wikipedia. Moreover, we use it to assess the quality of the labels used as ground truth.

5.1 Setup and results

As presented in Section 4.5, the *flat* model performed better than the *hierarchical* model. As for the techniques for mitigating imbalance, sample-weighting was better than the baseline setup, as shown in Section 4.3. Thus we use the flat model, and its architecture is found in Figure 3.5. The threshold for predictions is set to 0.5. The number of trainable parameters is $8.4M$. Furthermore, we use the *hierarchical* data, as it also improved overall performance, as shown in Section 4.1.

Training is conducted as specified in Section 3.2.1, with the sole difference that the model is trained and evaluated on the same data containing $760K$, similarly as done in [26]. Ideally, we would like to train and evaluate all of the available data. However, we still get valuable insights using $760K$ images – around one-fifth of the available data in English Wikipedia. Moreover, weights are initialized as pre-trained on ImageNet, as they give better results for 15 epochs in Table 4.4.

Table 5.1 presents the performance metrics of the model when predicted on the images it is trained on, using hard thresholds of 0.5 for all classes, and when using per-class thresholds calculated as specified in Section 3.2.1 for a matter of comparison.

Table 5.1: Performance metrics on predictions on the training data.

Classifier	***AUC(PR)***	**Precision ǀ Recall**		
		All labels	Top 5	Rest
Flat	0.349	0.525 ǀ 0.229	0.722 ǀ 0.578	0.487 ǀ 0.162
Flat (per-class thresh)	0.349	0.335 ǀ 0.524	0.601 ǀ 0.784	0.283 ǀ 0.474

5.2 Label distribution

See in Figure 5.1 the label distribution of the predicted images.

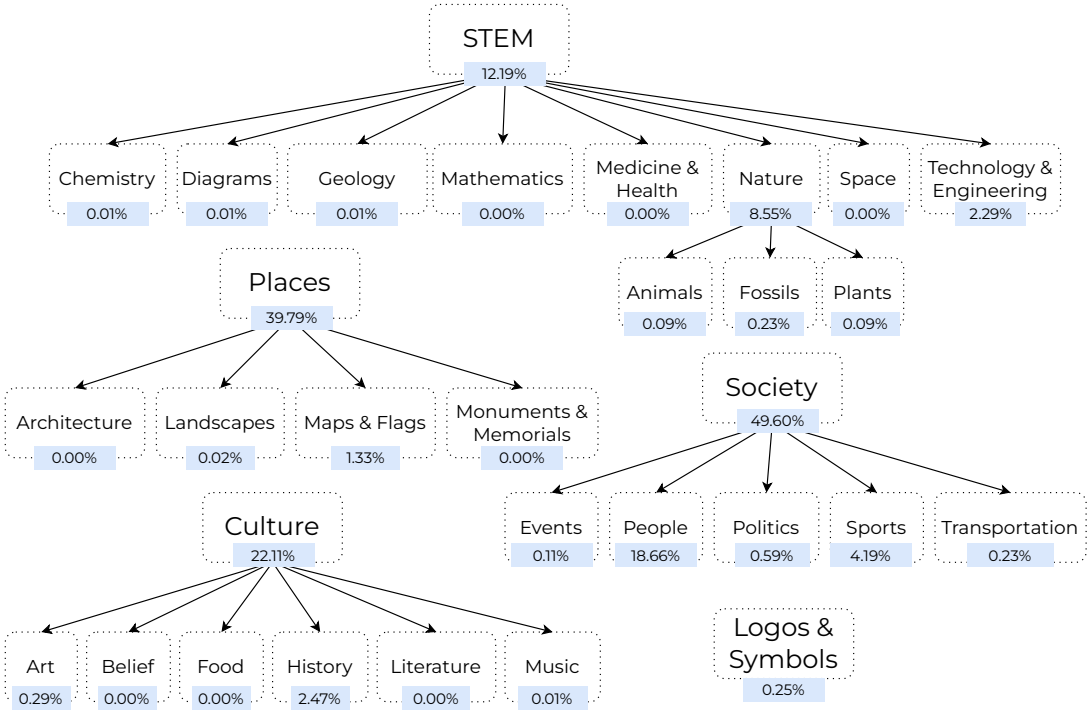


Figure 5.1: Label distribution of the predictions. Note that there is no post-processing in the flat model that enforces respect for the label hierarchy when predicting. So the percentages of the leaf labels do not sum up to the parent labels, nor must they be smaller than the parent label.

5.3 Study of the labels

Here, we want to understand the reasons behind the model's performance in classifying images with a particular label. We hypothesize that at least three factors affect the performance metrics of labels: the number of samples, bias from pre-training on ImageNet, and whether there are learnable signals in the label.

Figure 5.2 we see that there is a clear correlation between the number of samples and *the AUC(PR)* of the labels. However, there are some outliers in the upper left part of the graph, such as the labels: Maps & Flags, Sports, Plants, Logos & Symbols, Fossils, and Nature. These labels have few samples in the training set and, simultaneously, have an above-average *AUC(PR)* metric. Moreover, Figure 5.3 displays the precision-recall curves of all labels and how much improvement they have compared to random.

Finally, let us look at some qualitative examples that prove that the model does well at finding correct labels that are missing in the ground truth labels – remember that these labels are not human ground truth, but predictions generated by Salvi [1]. Figures 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, and 5.11 display these examples for a list of labels.

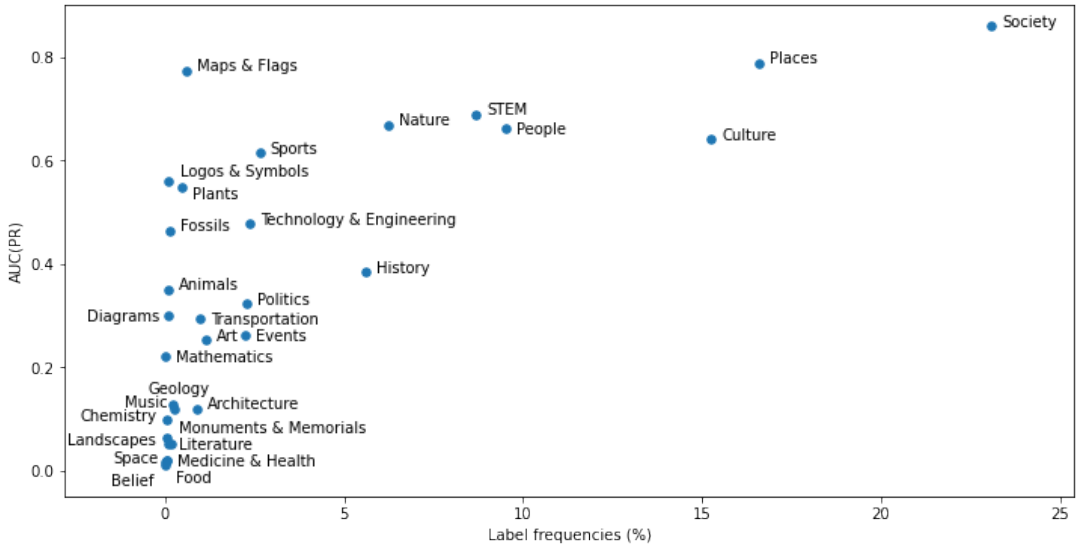


Figure 5.2: Relation between label frequencies and the *AUC(PR)* metric. Note that there is a general correlation between a greater number of samples and the *AUC(PR)* metric. However, there are outliers, such as the topics Maps & Flags, and Sports.

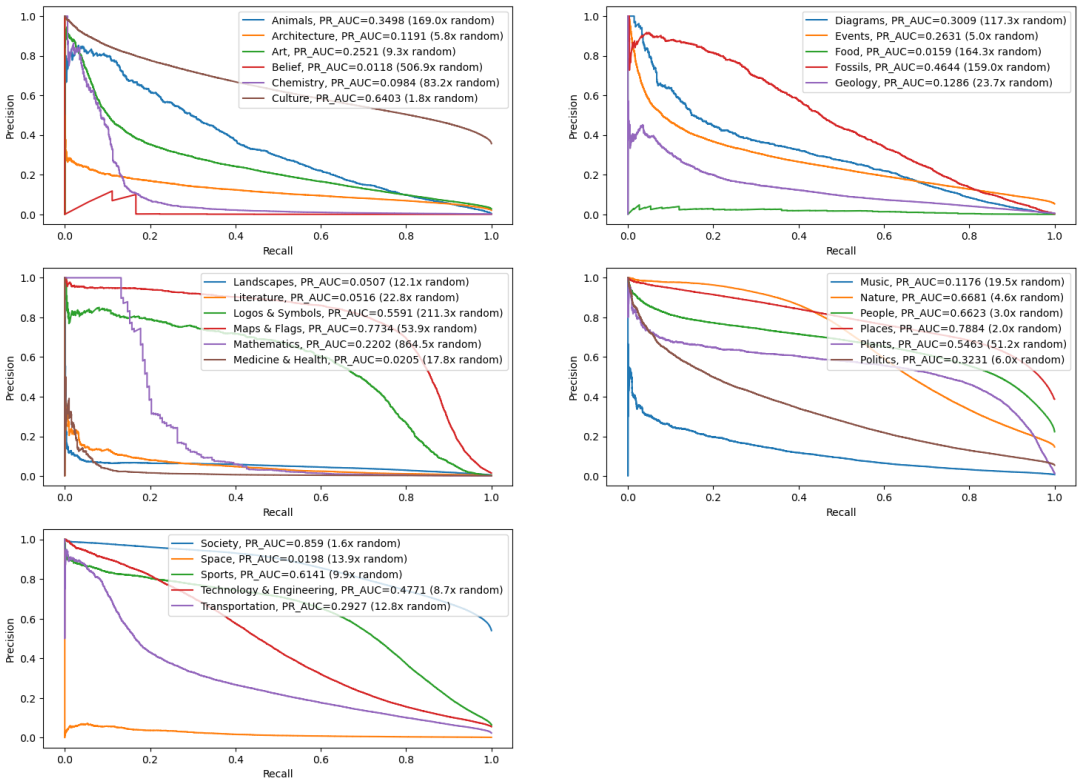


Figure 5.3: Precision-recall curves for all the labels. The *AUC(PR)* value and its improvement compared to random are provided in the legend.

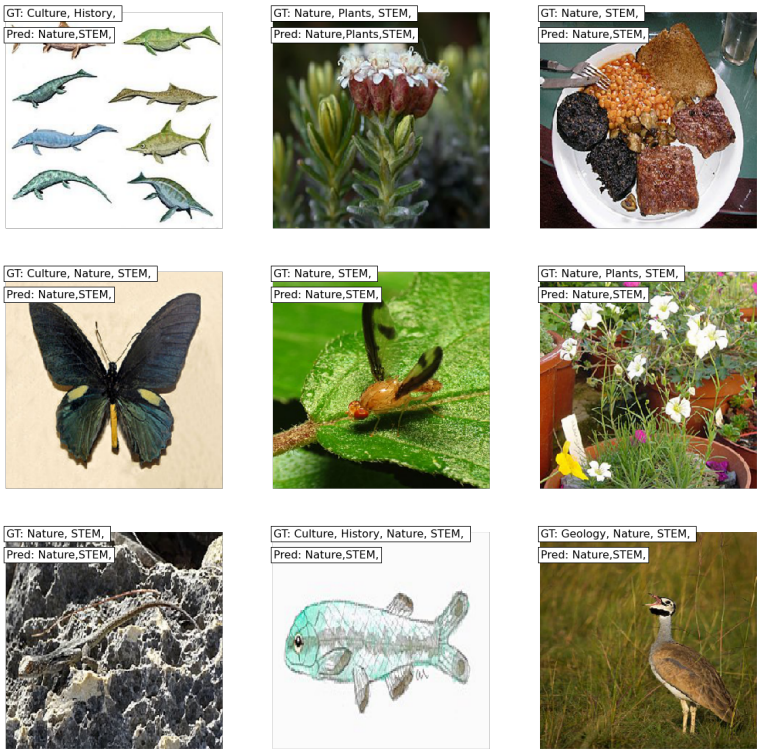


Figure 5.4: Images that the model predicts as *Nature*. Note that the image on the top left does not have *Nature* in the ground truth (GT), while a human would likely classify it so.

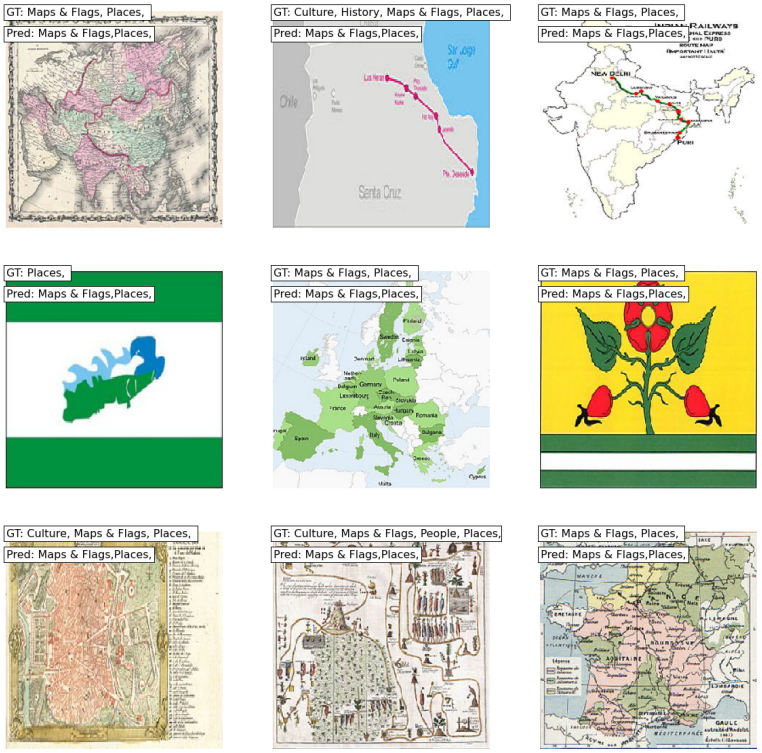


Figure 5.5: Images that the model predicts as *Maps & Flags*. Note that the image on the center-left row does not have *Maps & Flags* in the ground truth (GT), while a human would likely classify it so.

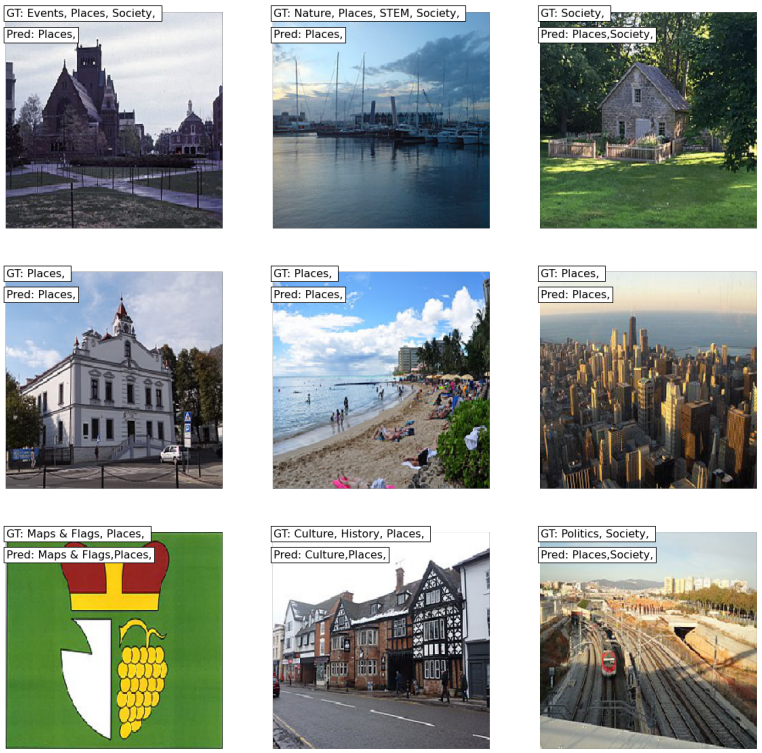


Figure 5.6: Images that the model predicts as *Places*. Note that the images on the top and bottom right do not have *Places* in the ground truth (GT), while a human would likely classify it so.

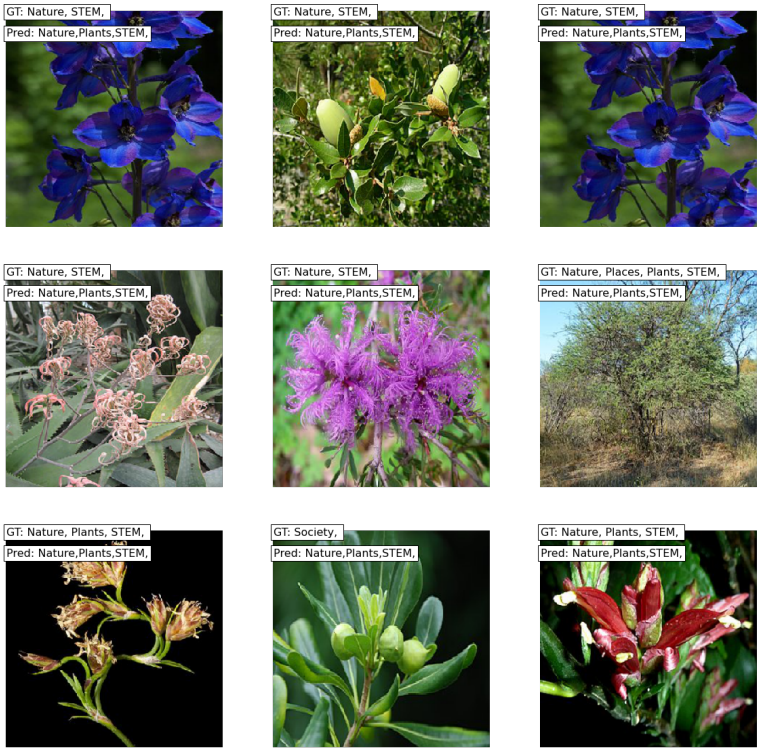


Figure 5.7: Images that the model predicts as *Plants*. Note that many images do not have *Plants* in the ground truth (GT), while a human would likely classify it so.

Figure 5.8: Images that the model predicts as *History*. Note that the images on the center-left do not have *History* in the ground truth (GT), while a human would likely classify it so.

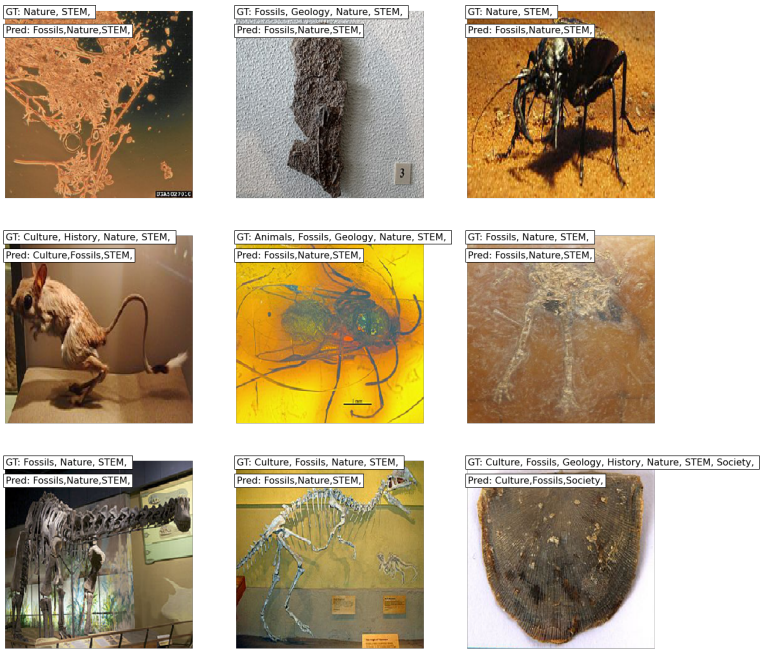


Figure 5.9: Images that the model predicts as *Fossils*. Note that the images on the top right and left and center-left do not have *Fossils* in the ground truth (GT), while a human would likely classify it so.

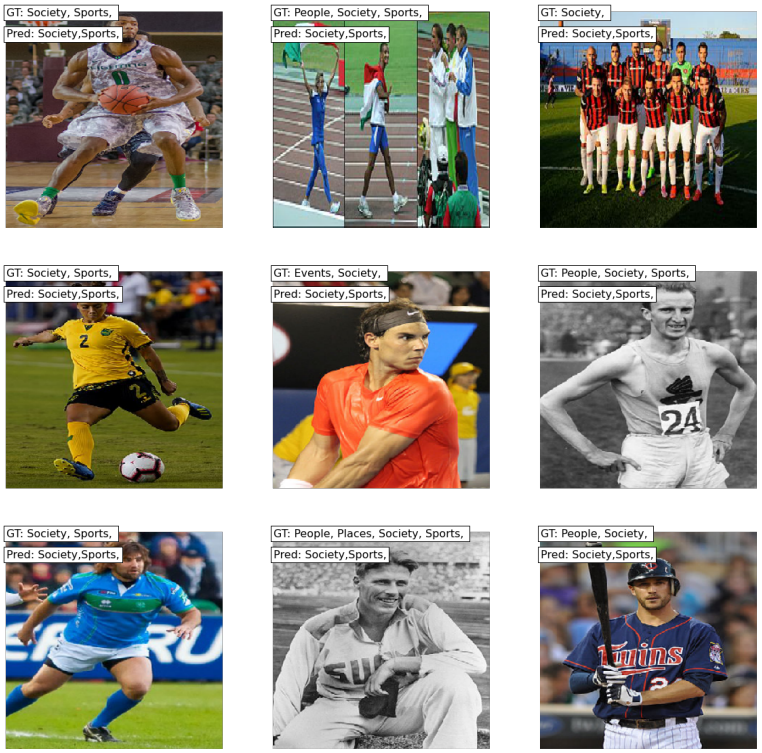


Figure 5.10: Images that the model predicts as *Sports*. Note that the images in the center and bottom right do not have *Sports* in the ground truth (GT), while a human would likely classify it so.

Figure 5.11: Images that the model predicts as *Transportation*. Note that the images in the center-right and bottom left do not have *Transportation* in the ground truth (GT), while a human would likely classify it so.

6 Discussion

This chapter contains a discussion of the methods used and of the obtained results.

6.1 Results

The classification results obtained depended on many factors. To begin with, as presented in Section 3.1.2, the labels are not human ground truth but predictions from the work done by Salvi [1]. This introduces noise in the labels differently, as explained in Section 3.1.3. Later, other important factors affect the *AUC(PR)* of all labels, such as the total number of labels, the number of samples per label, the bias from pre-training on ImageNet, and the signal in the labels. This complexity contributes to the predictions presented in Figure 5.1 not being an entirely correct understanding of the visual distribution of Wikipedia. However, it does give us some starting intuitions that can be used in further developing the label taxonomy of Salvi [1] and the classifier.

Since this is the first time that multilabel classification is performed on WIT using the labels from Salvi [1], no expectations on the results existed in advance.

6.2 Method

Here, we discuss the sources, datasets, and other methods used in this work.

6.2.1 Sources

The primary sources used for the theory chapter were *A survey of transfer learning* [31] by Weiss et al., and *Survey of deep learning with class imbalance* [19] by Johnson et al.. Both have good reviews and thousands of citations on *Google Scholar*. The sources from the related literature section are mainly taken from papers written by researchers on Wikipedia. The rest of the literature was found through *Google Scholar* and *Papers With Code*.

6.2.2 Datasets

The images are taken from the WIT dataset [38], a relevant work published in 2021 by researchers of *Google* that assembles all of the images of Wikipedia and the textual context around them.

Regarding the labels, the use cases by Wikipedia made it necessary to create a customized label set of *topical* labels. The work is not yet published; however, it is done under the supervision of Tiziano Piccardi and Miriam Redi, researchers from the Wikimedia Foundation who are also the main supervisors of this thesis. Thus, we conclude that the use of these labels is justified.

6.2.3 Architecture

Two model architectures were used in this work, *flat* and *hierarchical*. For both architectures, transfer learning with parameters pre-trained on ImageNet is used, something done and recommended in established related work of image classification. The flat classifier is based on EfficientNet, an architecture proven to be efficient and scalable, which are critical qualities in Wikipedia, where scalability is essential. Moreover, the choice of the hierarchical model was based on the page rating models that perform hierarchical multilabel classification on *Papers With Code*. Later in the project, we found out that the proposed hierarchical model had not been tested on more extensive image datasets or with CNNs. If we had been able to restart the project, we would have chosen a more established hierarchical multilabel classification method.

6.3 The work in a wider context

Since Wikipedia is the largest encyclopedia that ever existed, there must be tools to help maintain its quality and credibility. Thus, this work, inspired by its textual counterpart ORES [15], can significantly help the volunteer maintainers of Wikipedia when further developed and scaled.

7 Conclusion

Herein, we answer the research questions and give ideas for future work.

7.1 Research questions

- **RQ1. How to mitigate the consequences of having heavily imbalanced data when performing multilabel classification on Wikipedia images?**

  First of all the data organized in a hierarchy proved to give better results than the hierarchy-agnostic data. The two main reasons for this are believed to be that the ground truth is enriched by correct labels, and that the imbalance between positive and negative samples is decreased.

  Moreover, two main methods to address the classification of long-tailed data were tried out: data-level or algorithm-level. None of the data-level techniques proved effective, something expected given the level of label concurrence in the flat and the hierarchical data. Among algorithm-level techniques, sample-weighting and per-class thresholds effectively improved the *AUC(PR)* metric. Per-class thresholds were good if one wanted to increase recall substantially (especially of the rarer classes); however, precision decreased when using this method.

  Furthermore, initializing weights as random did not perform better than initialing weights pre-trained on ImageNet. However, we suspect better results can be achieved by letting the model train for more epochs and smarter initializing weights.

  Finally, a correlation between fine-tuning more layers and better classification metrics for all classes is verified.

- **RQ2. How does a *hierarchical* topic classification of the images in Wikipedia perform compared to a *flat* classification?**

  The chosen hierarchical model performed worse than the flat model regarding the *AUC(PR)*, precision, and recall metrics. Still, making predictions coherent with the hierarchy through the model itself or through post-processing is recommended, especially if the hierarchy is already encoded in the data.

- **RQ3. Using the developed pipeline, what insights can be drawn from the visual data of Wikipedia regarding its distribution and possible knowledge gaps?**

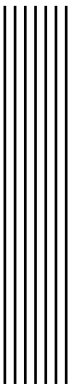
  The visual distribution that the model predicted is presented in Figure 5.1, together with some other figures in Chapter 5. Intuitions that can be drawn from that are, for instance, topics such as Maps & Flags, Sports, Logos & Symbols, Plants, and Nature are easier to classify for reasons such as more signal in the data or pre-training biases from ImageNet. Moreover, we also showed examples that the model is useful for finding correct labels missing in the ground truth labels.

7.2 Future work

An evident approach to continue this work is to continue developing the model, which can be done by, e.g., making the model multi-modal mirroring Lanchantin et al. [22] or Redi et al. [30]. One could also implement other hierarchical multilabel classification models and empirically compare their performances on this dataset. An especially interesting model to implement is XGBoost, a tree-boosting algorithm that is used in ORES [15]. Moreover, experimenting with a gradually decreasing learning rate can be looked further into.

Two baseline models could be implemented to compare the performance of the current model. First, we could use a multilabel image classification model trained on COCO-stuff [5] to detect all the objects on the images of WIT. After that, we would input the model with the objects present in the image, with the ground truth still being the image topic. This way, we would have a link between the visual objects in the image, and the topics that these objects are associated with, increasing the interpretability of the model. For instance, it is logical that an image with vehicles is labeled as Transport, and an image with animals is labeled with topics Animal and Nature. Another interesting baseline model to try out could be parsing out the names in the image file and using semantical similarity between those words and the topic names.

Finally, and most importantly, it would be helpful to study how good the model is at inferring labels missing from the weak labels compared to a *human* ground truth, found through crowd-sourcing at, for instance, Amazon Turk.



Bibliography

[1] *Automated Categorization of Wikipedia Images*. https://meta.wikimedia.org/wiki/Research:Automated_Categorization_of_Wikipedia_Images. Accessed: 2022-12-08.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching Word Vectors with Subword Information". In: (2016). Accepted to TACL. URL: http://arxiv.org/abs/1607.04606.

[4] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural Networks* 106 (2018), pp. 249–259. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2018.07.011. URL: https://www.sciencedirect.com/science/article/pii/S0893608018302107.

[5] Holger Caesar, Jasper R. R. Uijlings, and Vittorio Ferrari. "COCO-Stuff: Thing and Stuff Classes in Context". In: *CVPR 2018* (2016). arXiv: 1612.03716. URL: http://arxiv.org/abs/1612.03716.

[6] Ricardo Cerri, Rodrigo C. Barros, and André C. P. L. F. de Carvalho. "Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks". In: *2011 11th International Conference on Intelligent Systems Design and Applications*. 2011, pp. 337–343. DOI: 10.1109/ISDA.2011.6121678.

[7] Francisco Charte, Antonio Rivera, María José del Jesus, and Francisco Herrera. "A First Approach to Deal with Imbalance in Multi-label Datasets". In: *Hybrid Artificial Intelligent Systems*. Ed. by Jeng-Shyang Pan, Marios M. Polycarpou, Michał Woźniak, André C. P. L. F. de Carvalho, Héctor Quintián, and Emilio Corchado. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 150–160.

[8] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *SST 2014* (2014). arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

[9] Jianfeng Dong, Xirong Li, and Cees G. M. Snoek. "Predicting Visual Features From Text for Image and Video Caption Retrieval". In: *IEEE Transactions on Multimedia* 20.12 (2018), pp. 3377–3388. DOI: 10.1109/TMM.2018.2832602.

[10] Charles Elkan. "The Foundations of Cost-Sensitive Learning". In: IJCAI'01. Seattle, WA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 973–978. ISBN: 1558608125.

[11] Meng Joo Er, Rajasekar Venkatesan, and Ning Wang. *An Online Universal Classifier for Binary, Multi-class and Multi-label Classification*. 2016. DOI: 10.48550/ARXIV.1609.00843. URL: https://arxiv.org/abs/1609.00843.

[12] Carmen Esposito, Gregory A. Landrum, Nadine Schneider, Nikolaus Stiefl, and Sereina Riniker. "GHOST: Adjusting the Decision Threshold to Handle Imbalanced Data in Machine Learning". In: *Journal of Chemical Information and Modeling* 61.6 (2021). PMID: 34100609, pp. 2623–2640. DOI: 10.1021/acs.jcim.1c00160. eprint: https://doi.org/10.1021/acs.jcim.1c00160. URL: https://doi.org/10.1021/acs.jcim.1c00160.

[13] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. "A Multiple Resampling Method for Learning from Imbalanced Data Sets". In: *Computational intelligence* 20.1 (2004-02). ISSN: 0824-7935.

[14] Eleonora Giunchiglia and Thomas Lukasiewicz. "Coherent Hierarchical Multi-Label Classification Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 9662–9673. URL: https://proceedings.neurips.cc/paper/2020/file/6dd4e10e3296fa63738371ec0d5df818-Paper.pdf.

[15] Aaron Halfaker and R. Stuart Geiger. "ORES: Lowering Barriers with Participatory Machine Learning in Wikipedia". In: *Proc. ACM Hum.-Comput. Interact.* 4.CSCW2 (Oct. 2020). DOI: 10.1145/3415219. URL: https://doi.org/10.1145/3415219.

[16] Zellig S. Harris. "Distributional Structure". In: WORD 10.2-3 (1954), pp. 146–162. DOI: 10.1080/00437956.1954.11659520. eprint: https://doi.org/10.1080/00437956.1954.11659520. URL: https://doi.org/10.1080/00437956.1954.11659520.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[18] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CVPR 2017* (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[19] Justin M. Johnson and Taghi M. Khoshgoftaar. "Survey on deep learning with class imbalance". In: *Journal of Big Data* 6.1 (Mar. 2019), p. 27. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0192-5. URL: https://doi.org/10.1186/s40537-019-0192-5.

[20] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *ICLR (Poster)*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14.

[21] Gakuto Kurata, Bing Xiang, and Bowen Zhou. "Improved Neural Network-based Multi-label Classification with Better Initialization Leveraging Label Co-occurrence". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 521–526. DOI: 10.18653/v1/N16-1063. URL: https://aclanthology.org/N16-1063.

[22] Jack Lanchantin, Tianlu Wang, Vicente Ordonez, and Yanjun Qi. "General Multi-label Image Classification with Transformers". In: *CoRR* abs/2011.14027 (2020). arXiv: 2011.14027. URL: https://arxiv.org/abs/2011.14027.

[23] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[24] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. "Focal Loss for Dense Object Detection". In: *ICCVL 2017* (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[25] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022. URL: https://smlbook.org.

[26] Sylvain Lugeon, Tiziano Piccardi, and Robert West. "Language-Agnostic Website Embedding and Classification". In: *ICWSM* (2022). arXiv: 2201.03677. URL: https://arxiv.org/abs/2201.03677.

[27] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[28] Luca Masera and Enrico Blanzieri. "AWX: An Integrated Approach to Hierarchical-Multilabel Classification". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim. Cham: Springer International Publishing, 2019, pp. 322–336. ISBN: 978-3-030-10925-7.

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. DOI: 10.48550/ARXIV.1301.3781. URL: https://arxiv.org/abs/1301.3781.

[30] Oleh Onyshchak and Miriam Redi. "Image Recommendation for Wikipedia Articles". PhD thesis. Jan. 2020. DOI: 10.13140/RG.2.2.17463.27042.

[31] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

[32] *Papers with Code, Hierarchical Multilabel Classification*. https://paperswithcode.com/task/hierarchical-multi-label-classification. Accessed: 2022-11-16.

[33] Luis Perez and Jason Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning". In: *CoRR* abs/1712.04621 (2017). arXiv: 1712.04621. URL: http://arxiv.org/abs/1712.04621.

[34] Daniele Rama, Tiziano Piccardi, Miriam Redi, and Rossano Schifanella. "A Large Scale Study of Reader Interactions with Images on Wikipedia". In: *EPJ Data Sci* (2021). arXiv: 2112.01868. URL: https://arxiv.org/abs/2112.01868.

[35] Miriam Redi. *Prototypes of Image Classifiers Trained on Commons Categories*. https://meta.wikimedia.org/wiki/Research:Prototypes_of_Image_Classifiers_Trained_on_Commons_Categories. 2020.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[37] Yang Song, Aleksander Kołcz, and C. Lee Giles. "Better Naive Bayes Classification for High-Precision Spam Detection". In: *Softw. Pract. Exper.* 39.11 (Aug. 2009), pp. 1003–1024. ISSN: 0038-0644.

[38] Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. *WIT: Wikipedia-based Image Text Dataset for Multimodal Multilingual Machine Learning*. 2021. DOI: 10.48550/ARXIV.2103.01913. URL: https://arxiv.org/abs/2103.01913.

[39] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *AAAI Press* (2017). arXiv: 1602.07261. URL: http://arxiv.org/abs/1602.07261.

[40] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: (2016), pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[41] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *ICML* (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[42] *Training and evaluation with the built-in methods – sample weights*. https://www.tensorflow.org/guide/keras/train_and_evaluate#sample_weights. Accessed: 2022-12-28.

[43] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. "Decision trees for hierarchical multi-label classification". In: *Machine Learning* 73 (Nov. 2008), pp. 185–214. DOI: 10.1007/s10994-008-5077-3.

[44] *Wikimedia Commons Categories*. https://commons.wikimedia.org/wiki/Commons:Categories. Accessed: 2022-12-16.

[45] Tong Wu, Qingqiu Huang, Ziwei Liu, Yu Wang, and Dahua Lin. "Distribution-Balanced Loss for Multi-label Classification in Long-Tailed Datasets". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Cham: Springer International Publishing, 2020, pp. 162–178. ISBN: 978-3-030-58548-8.

[46] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. "Learning Transferable Architectures for Scalable Image Recognition". In: (2017). arXiv: 1707.07012. URL: http://arxiv.org/abs/1707.07012.