Object Detection in an Image

Bruno de Almeida Silveira

October 2019

Abstract

Here comes an abstract...

1 Definition

1.1 Project Overview

The main challenge in this project is to create a model that identifies many objects in an image. This challenge involves two main tasks. The former identifies in an image where it is the position of an object that could be classified, and around it, we are going to draw a bounding box. The latter classifies those bounding boxes correctly, labeling a chair as a chair, a table as a table, and a human hand as a human hand.

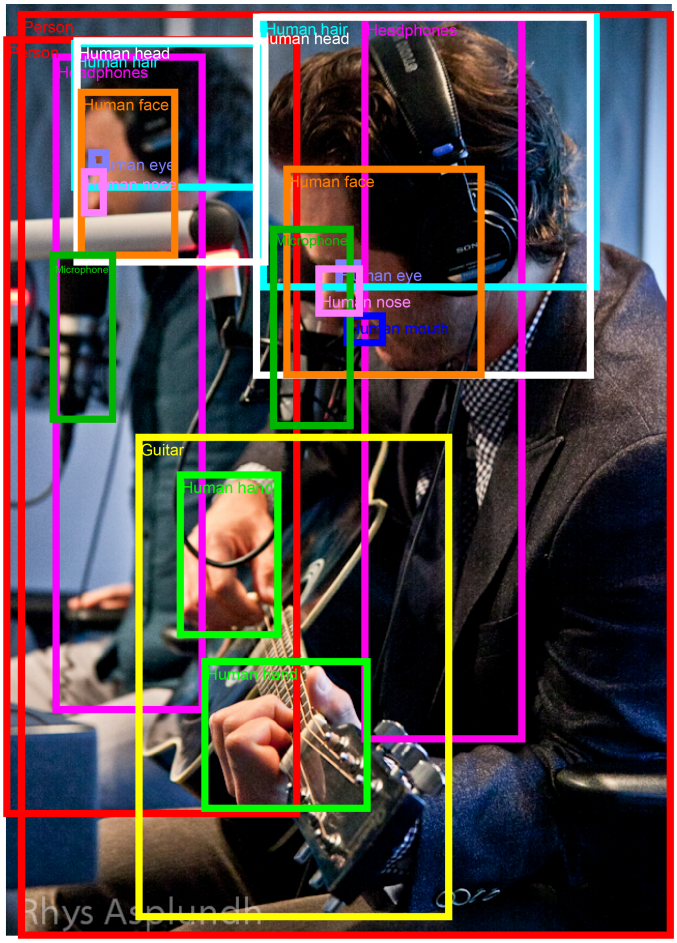


Figure 1: Mark Paul Gosselaar plays the guitar by Rhys A. [1]

The Kaggle challenge created by Google called Open Images 2019 - Object Detection [2] motivates this project. This Kaggle was created using the recent dataset announced, the Open Images Dataset v5 [1]. This project proposes to show some strategies to solve the problem, giving a deep dive into some deep neural network architectures.

1.2 Problem Statement

As described, the problem is going to be divided into two, the problem to create the correct bounding box and the problem to classify it. The idea is to create a simple pipeline with two models concatenated, and each model is going to optimize each technique.

The first model, called Bounding Box Identifier, is responsible for identifying all objects that could be classified by the Image Classifier Model. The loss function of this model is going to be the IoU (Intersection over Union) - described better in the next subsection.

On the other side, for the Image Classifier model, it is going to use the Categorial Crossentropy as a loss function.

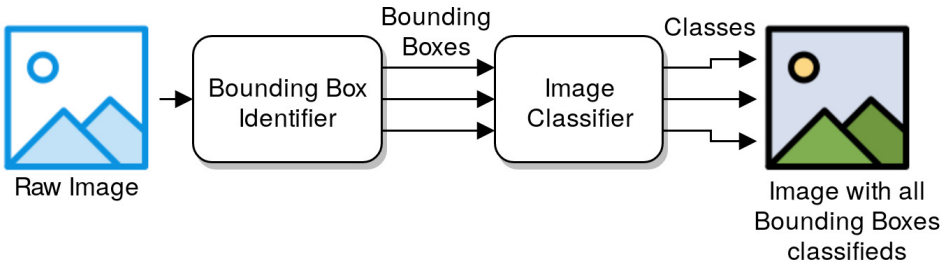


Figure 2: High level Model Pipeline

It is import to explaining more about the loss functions.

The Categorical Crossentropy used in the Classifier model is straightforward since it is a multiclass classifier. The idea is to compare for each class each distribution of the predictions made with the labeled distribution.

The IoU metric used as loss function arrives because mAP derives directly from the IoU. Since the mAP is going to be used as a metric to compare all models, it sounds reasonable to use IoU as a loss function to be optimized in the model responsible for creating the Bounding Boxes.

In the first step, the project starts to use the Transfer Learning technique in many famous networks - such as ResNet-N layers [3], Inception-V4 [4], Xception [5], and Inception-ResNets [4] - for both models, Object Identification, and Classifier. With Transfer Learning, we expect to achieve some reasonable results.

For the second and final step of the project, this work proposes to create, at minimum, one new architecture that could face other architectures seen above.

1.3 Metrics

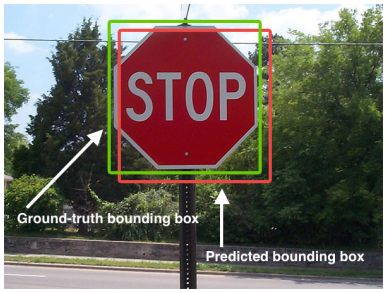
The metric proposed by Google in the competition is the mean Average Precision (mAP) [6], a very didactic explanation about the metric could be found in here [7] and [8].

The mAP metric could be defined as

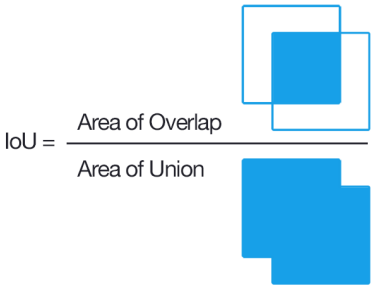
$$mAP = \frac{\sum_{c=1}^{C} AP_c}{C}$$

where C value is the number of all categories (classes).

To understand AP_c, it must comprehend first what is IoU. IoU is the Intersection over Union. It is equal to the ratio of the $Area\ of\ Overlap$ and the $Area\ of\ Union$, considering the Predict bounding box (created by the model) and the Ground-truth bounding box (previously annotated).



(a) Difference between a Predict bounding box with a Ground-truth bouding box [9]



(b) IoU visual represented [9]

Here, it is going to define:

$$\begin{aligned} TruePositive &\doteq IoU > 0.5 \\ FalsePositive &\doteq IoU < 0.5 \\ &\quad or\ DuplicatedPredictBoundingBox \\ FalseNegative &\doteq IoU > 0.5 \\ &\quad and\ WrongClassification \end{aligned}$$

With the concept of True Positive (TP), True Negatives (TN), and False Positives (FN) defined, it is possible to create a Precision-Recall Curve, which defines a function that gives a precision based on the recall.

So by definition, AP_c (Average Precision of some category c), is defined as an Area Under the Curve (AUC) of the Precision-Recall curve.

$$AP_c = \int_0^1 p(r)dr$$

where $p(r)$ is the precision defined in function of recall.

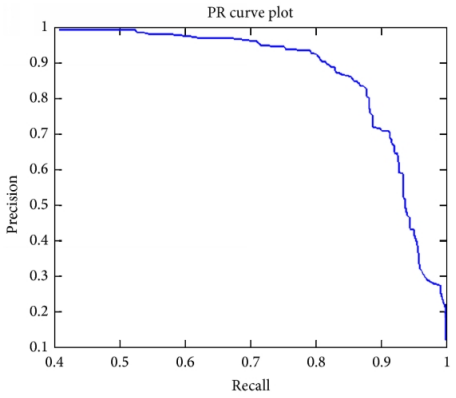


Figure 3: Precision-Recall curve [7]

It is essential to retain that in this project it is going to be used AP_{50} (which uses a threshold of 0.5 in IoU to define TP), but other average precision metrics could be used, like, AP_{75} (with IoU threshold of 0.75) or AP_{90} (with IoU threshold of 0.90).

2 Analysis

2.1 Data Exploration

For the problem itself, there are two datasets (Image-level and bounding boxes), plus two files that describe each class used in the labeling process, and the relation among them.

About the classes, there are 600 unique classes, and they are represented as a graph. So they have inheritances. All classes start from a class that I called "Entity" (it has no name, in fact), and all other classes derive from it or from the classes that derive from it on some level. In the appendix A, I expose all connections at each level (the deepest level is five).

The Train, Cross-Validation and Test set are already given by google. This project pretends to use the same distribution given. However, during the analysis it was possible to notice that there are some classes presented in Train set that are not presented in Test and Cross-Validation sets. The following table shows how much unique classes are present in each data set.

Table 1: Unique classes in each Dataset and amount of Bounding Boxes

	Unique Classes	Count of Boubing Boxes
Train	599	14,609,671
Cross-Validation	570	303,980
Test	583	937,327

Besides that, the distribution of each class is not uniform. 30 classes are responsible for 80% of bounding boxes labels, and 300 classes are responsible for less than 1% of bounding boxes. Of course, this affects how the final model performs in a random image.

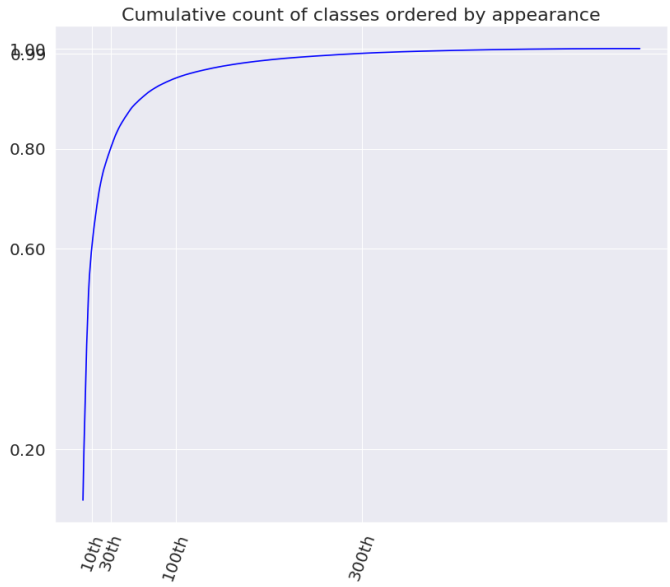


Figure 4: Cumulative count of classes order by appearance

There are two datasets, image-level label and bounding boxes, both divided by three (train, test, cross-validation). Google explains in the blog [10] how they create the Image-level dataset and how, with it, they create the bounding boxes dataset. The label-image set was created, and a little part of it (around 1.7 million images) were chosen to have more granular work. The labels previously created were relabeled and transformed in bounding boxes that identify, not only what is the object, but the position of those objects. This work only uses the bounding boxes dataset.

About Bouding Box Dataset, some points must be concerned. The labels in the original data set are encoded (probably to avoid mismatch and homonyms). In the analysis process they were converted to semantic labels (using an auxiliary dataset provided by google). However, in modeling, it is going to use the encoded version.

Table 2: Amount of each flag in Bounding Box dataset

	IsOccluded	IsTruncated	IsGroupOf	IsDepiction	IsInside
Train	9,629,150	3,643,883	852,641	774,485	13,718
Cross-Validation	134,497	69,698	26,360	14,181	2,200
Test	417,398	211,732	81,037	44,038	6,964

Beyond the label and the bounding boxes position, the dataset also

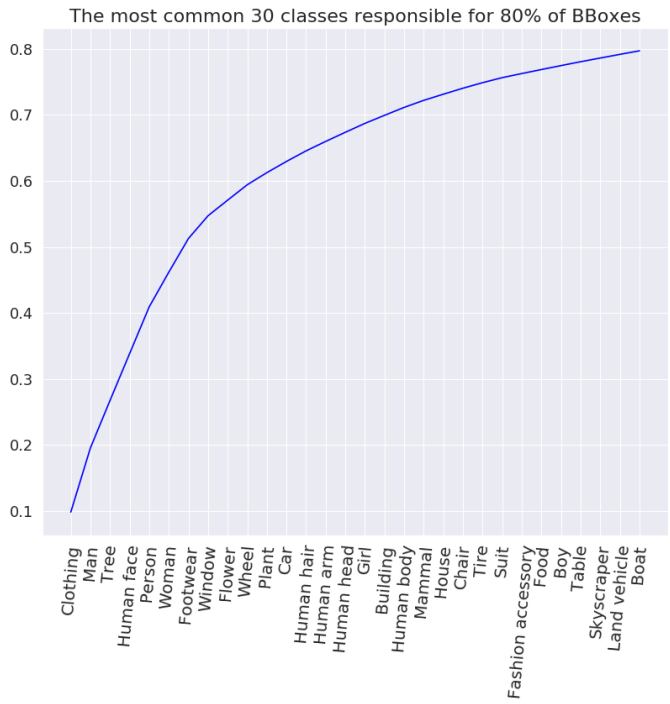


Figure 5: The thirty classes with most appearance

has some flags (IsOccluded, IsTruncated, IsGroupOf, IsDepiction, IsInside). These flags are very helpful to debug the decisions that a model does.

"IsOccluded" points a Bounding Box that has a type of obstruction in it, making it more difficult to be understandable.

"IsTruncated" points a Bounding Box that ends or starts in some edges of the image.

"IsGroupOf" means that the label given is about a group of things, and the bounding box is around all of those things.

"IsDepiction" represents bounding boxes that do not label the object itself, but a representation of the object. Draws, representation, costumes, are some examples.

"IsInside" is about the Bounding Boxes labeled inside rooms or buildings, with artificial light.

As stated, the distributions of classes are not uniform, and this is a big concern about how good the model will be. There is a more accurate and complete analysis, with a study of the correlations between flags and many other topics in all datasets. It can be checked in here [11].

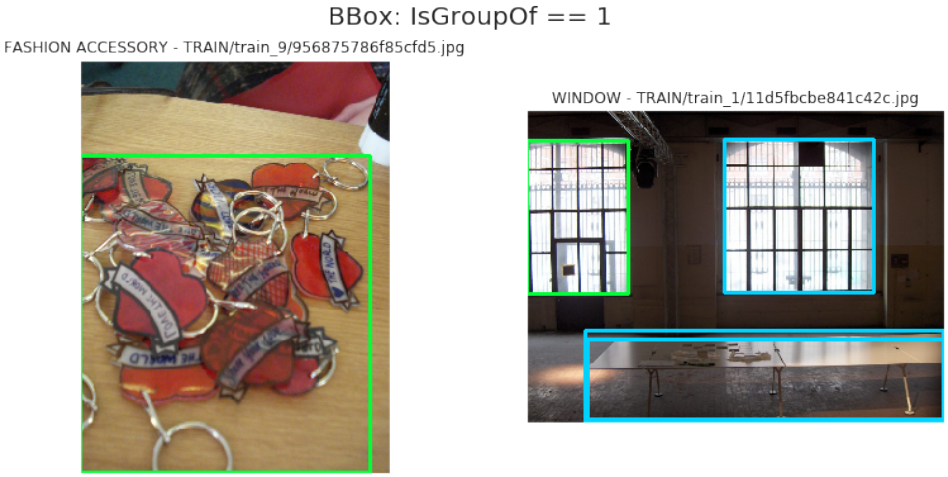


Figure 6: Bounding Box Green represents IsGroupOf True

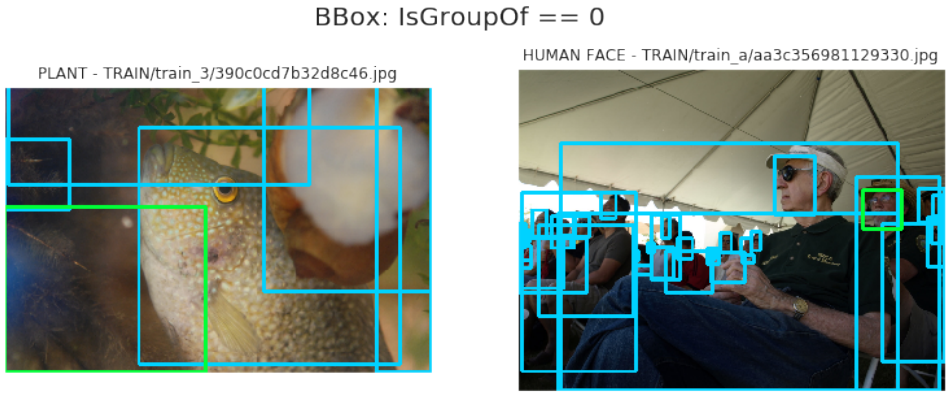


Figure 7: Bounding Box Green represents IsGroupOf False

2.2 Techniques

This project is about how to design and to create deep neural networks. More than that, it is going to deep dive into how convolutional neural networks work and how to use transfer learning to avoid massives trains and architecture designs.

Convolutional neural networks (CNN) are neural networks with convolutional layers. Convolutional layers are layers that identify patterns in the images. In the very low levels, they find the most common patterns, as edges, and in the very deep levels, they identify the most specific patterns, as the form of a cat. These patterns are identified, creating many filters and applying filters over filters to classify more complex patterns. Besides that, we condense the filters information applied in convolutional layers with pooling layers, which have the propose of removing the useless information

and keep the core information for the deep layers. [12]

Unlike conventional neural networks, CNNs tend to do not have layers fully connected, but only some nodes are connected. The main advantage of this technique is to have very fewer weights to calculate, which reduces drastically the computational power required to train.

AlexNet [13] was the first famous CNN architecture to classify images. Many others come later derived from it and newer ones, like VGG [14], ResNET [3], Inception [4] or Xception [5].

This fact is vital to the next topic this project wants to cover, Transfer Learning. Transfer Learning is a technique focused on transfer the knowledge generated in some neural network architecture, trained with some data, for a new context with a similar (could be slightly different) neural network architecture [15] [16].

This project starts the modeling process using transfer learning in some famous architectures. In the very end, it proposes an architecture not revolutionary, but good enough to solve the image detection problem.

2.3 Benchmark

This project was inspired by the Kaggle competition launched by Google, given that, It is going to use as a benchmark the leaderboard itself, [17] and the goal is to achieve at least the top fifteen score.

The Score evaluated is the same discussed in the metric topic of the first part of this project.

#	△pub	Team Name	Notebook	Team Members	Score	En
1	–	MMfruit			0.65887	
2	–	imagesearch		+4	0.65337	
3	–	Prisms		+3	0.64214	
4	–	PFDet		+3	0.62221	
5	▲ 2	Omni-Detection			0.60406	
6	▲ 2	Schwert			0.60231	
7	▼ 1	Team 5			0.60210	
8	▼ 3	pudae			0.59727	
9	–	[ods.ai] n01z3			0.59535	
10	▲ 2	dingwoai			0.58504	
11	▼ 1	J			0.58259	
12	▼ 1	Markup Oracles			0.57304	
13	–	ang TSS method		+4	0.54352	
14	–	tito			0.53885	
15	–	Appian			0.53629	

Figure 8: Leaderboard Image Dection Kaggle Competition - 2019

3 Methodology

3.1 Data Preprocessing

3.2 Implementation

3.3 Refinement

4 Results

4.1 Model Evaluation and Validation

4.2 Justification

5 Conclusion

5.1 Free-Form Visualization

5.2 Reflection

5.3 Improvement

References

[1] Google AI Blog. Introducing the open images dataset. https://ai.googleblog.com/2016/09/introducing-open-images-dataset.html, 2016.

[2] Google Research. Open images 2019 - object detection. https://www.kaggle.com/c/open-images-2019-object-detection/overview, jun 2019.

[3] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. Dec 2015. https://arxiv.org/abs/1512.03385.

[4] Vincent Vanhoucke Alex Alemi Christian Szegedy, Sergey Ioffe. Inception-v4, inception-resnet and the impact of residual connections on learning. Feb 2016. https://arxiv.org/abs/1602.07261.

[5] François Chollet. Xception: Deep learning with depthwise separable convolutions. Oct 2016. https://arxiv.org/abs/1610.02357.

[6] wikipedia. Mean average precision. https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision.

[7] Jonathan Hui. map (mean average precision) for object detection. https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.

[8] Ren Jie Tan. Breaking down mean average precision (map). https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52#d439.

[9] Adrian Rosebrock. Intersection over union (iou) for object detection. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[10] Google Research. Overview of open images v5. https://storage.googleapis.com/openimages/web/factsfigures.html.

[11] Bruno de Almeida Silveira. Eda - object detection - open images v5. http://htmlpreview.github.io/?https://github.com/BAlmeidaS/capstone-udacity-mle/blob/master/EDA.html.

[12] Paramartha Dutta Farhana Sultana, A. Sufian. Advancements in image classification using convolutional neural network. May 2019. https://arxiv.org/pdf/1905.03288.pdf.

[13] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutionalneural networks. 2012. https://arxiv.org/pdf/1905.03288.pdf.

[14] Andrew Zisserman Karen Simonyan. Very deep convolutional networks ffor large-scale image recognition. Apr 2015. https://arxiv.org/pdf/1409.1556.pdf.

[15] Sung Ju Hwang Jinwoo Shin Yunhun Jang, Hankook Lee. Learning what and where to transfer. May 2019. https://arxiv.org/pdf/1905.05901v1.pdf.

[16] Yoshua Bengio Hod Lipson Jason Yosinski, Jeff Clune. How transferable are features in deep neuralnetworks? Nov 2014. https://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

[17] Leaderboard. Leaderboard - open images 2019 - object detection. https://www.kaggle.com/c/open-images-2019-object-detection/leaderboard.

A Class hierarchy

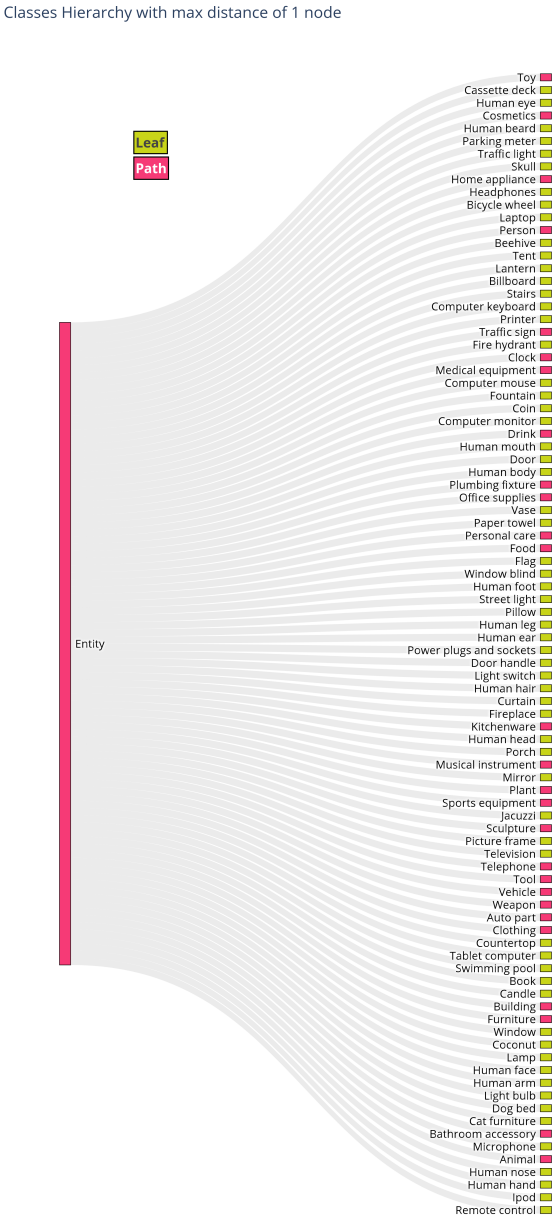


Figure 9: Class Hierarchy - First Level

Classes Hierarchy with max distance of 2 nodes - part 1

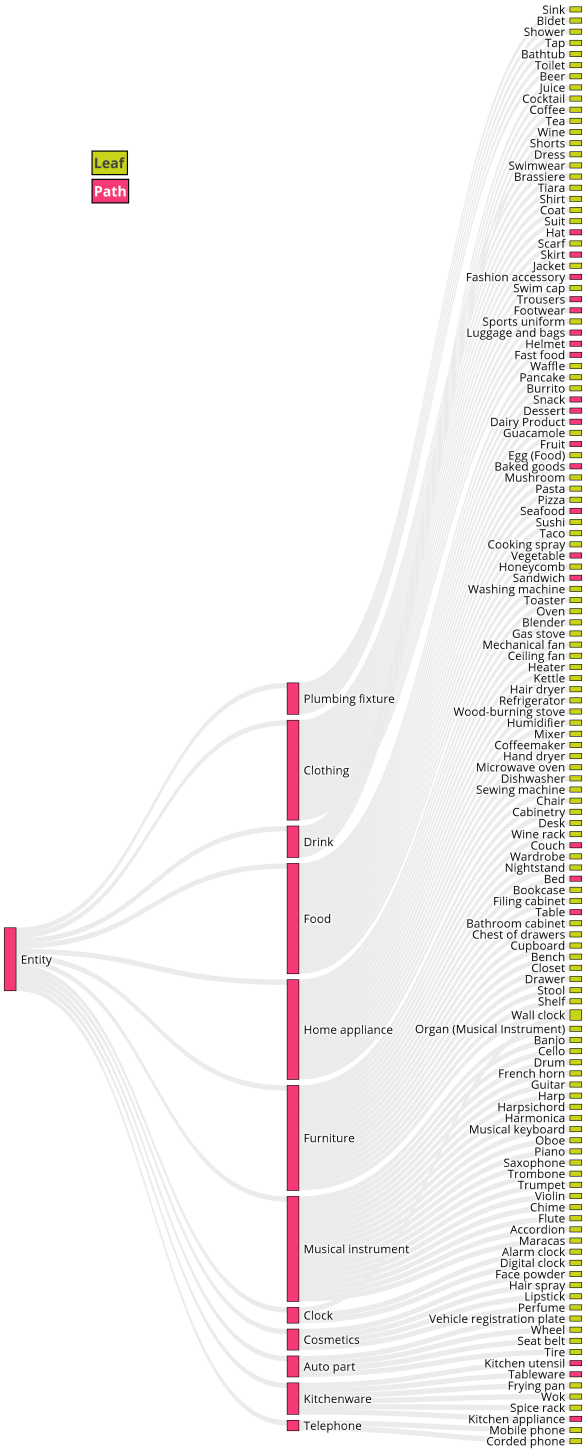


Figure 10: Class Hierarchy - Second Level first part

Classes Hierarchy with max distance of 2 nodes - part 2

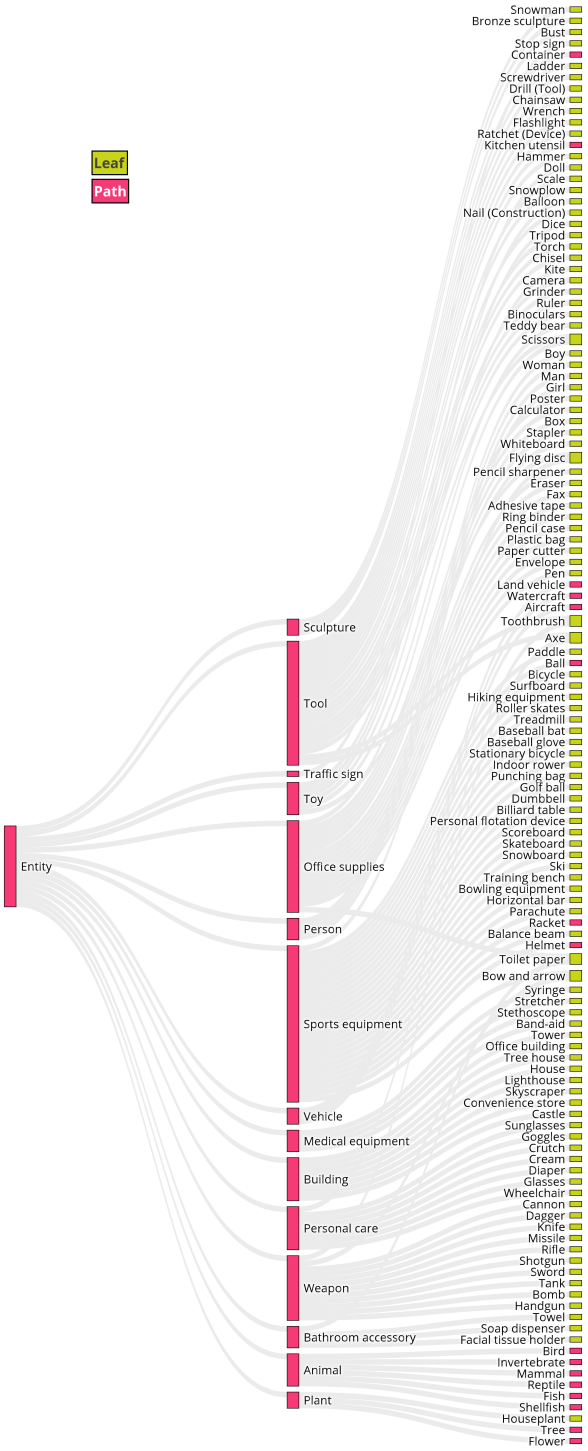


Figure 11: Class Hierarchy - Second Level second part

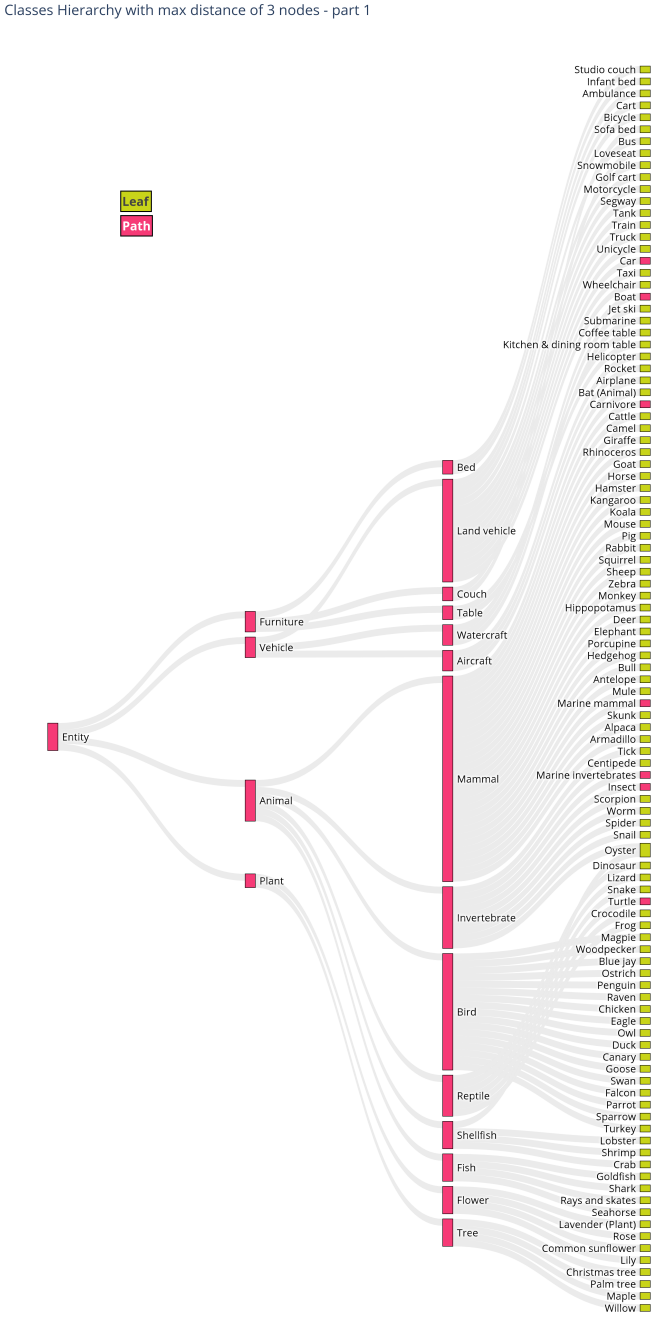


Figure 12: Class Hierarchy - Third Level first part

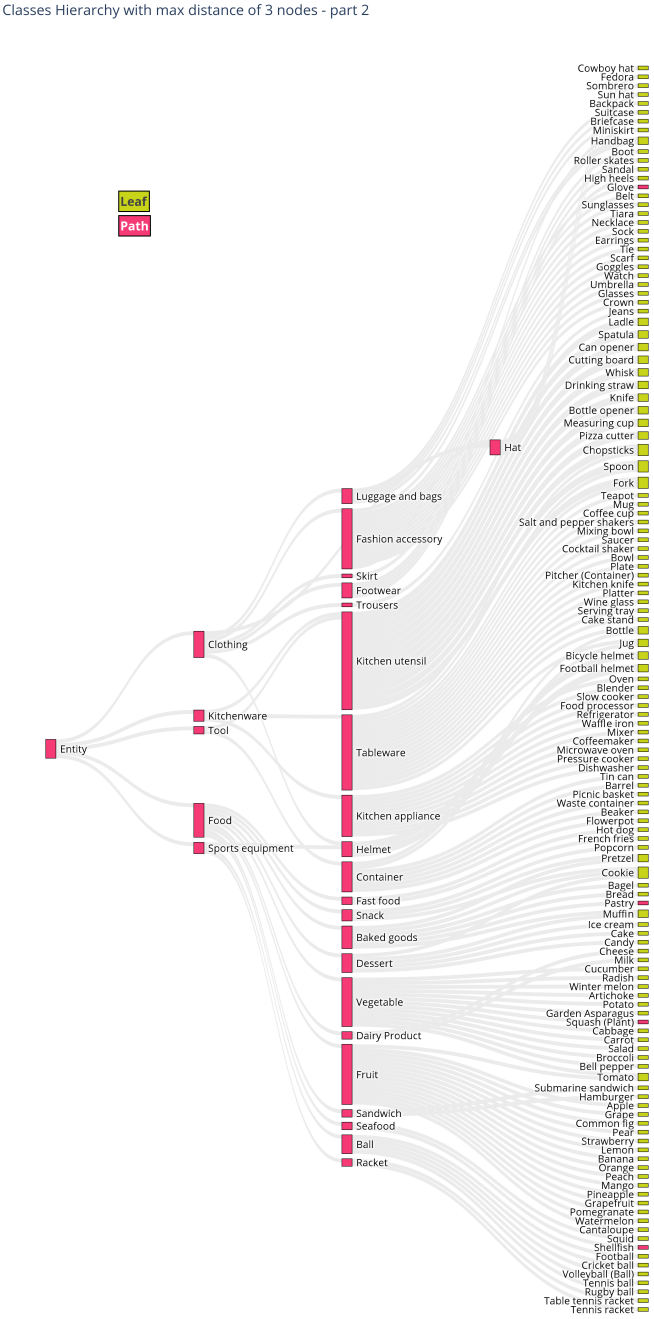


Figure 13: Class Hierarchy - Third Level second part

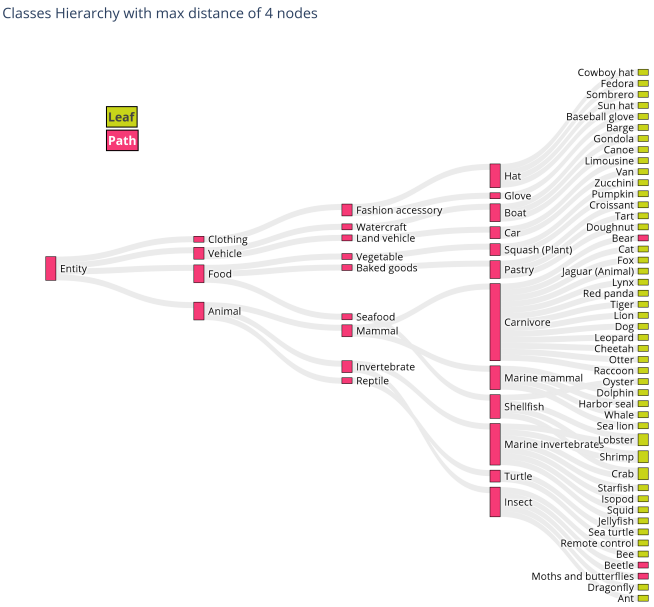


Figure 14: Class Hierarchy - Fourth Level

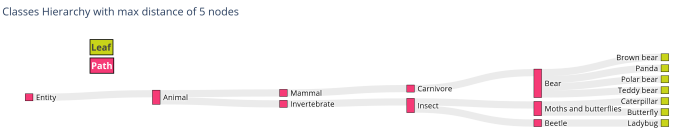


Figure 15: Class Hierarchy - Fifth Level