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 objects in images. This challenge involves two main tasks. The former is to identify object positions in a picture, and the latter is to classify those objects correctly.

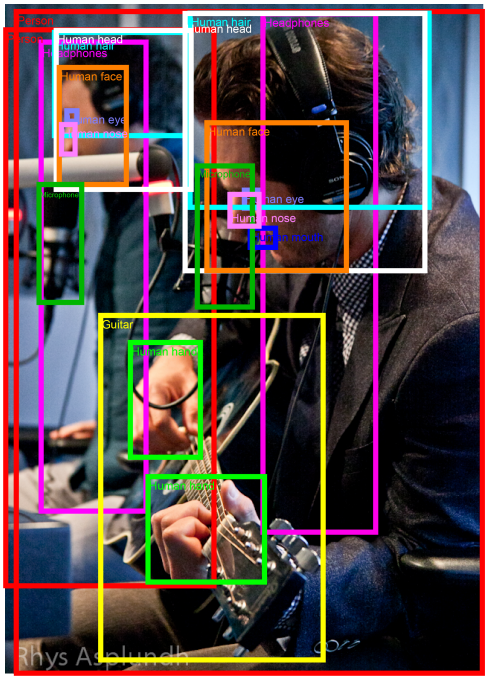


Figure 1: Mark Paul Gosselaar plays the guitar by Rhys A. [1]

The Kaggle competition created by Google [2] using a recently announced dataset is called Open Images 2019 - Object Detection[1], and it motivates this work. This project implements a strategy, using Deep Learning to solve the problem, which is grounded in some researches that this paper is going to present.

This project is a result of 3 months of research and hard work made in the area of computer vision and deep learning architecture. More than to solve the Kaggle problem, the main objective was learning about computer vision and more about Deep Learning.

1.2 Problem Statement

I want to start discussing the main difference between the proposal of the project and the final project delivered. When I wrote the proposal, I did not have the full context of the area, besides that, I proposed a possible and relevant architecture. The idea was to create a simple pipeline with two models concatenated, and each model would optimize one technique (classification and localization of bounding boxes). However, during more studying, it was clear that this architecture was burdensome and inefficient, which is not a problem for some circumstances, but this is not the case. Given that I have limited resources and time, and the dataset has around 1.7 million images, my proposal to use more than one models concatenated, knowing that I had to running the training process a few times, sounds very overwhelming, so I had to change my initial design.

We could define the object detection research area in two big sets about how to predict objects in an image. The first goes to path of pass the image for the model more than once to return one prediction. The computer vision area believed for much time that this was the only way. In the very beginning, there were very complex architectures using SVMs (Support Vector Machines) and regressors to draw bounding boxes. Nowadays, there are more robust architectures using deep learning like RCNNs, FAST RCNNS and FASTER RCNNs. (referencias???) The second goes to a performance path and limits the model to pass the image just once to predict both, the bounding box localization and the classification, at the same time. Examples of this approach are YOLO (You Only Look Once) and SSD (Single Shot Detection). (referencias???)

This project uses SSD structure, which will

be detailed discussed in the next sections. At the very end, I am going to present the changes that were made in the architecture to use new networks (like Resnet and Xception) to extract more accurate data from the images.

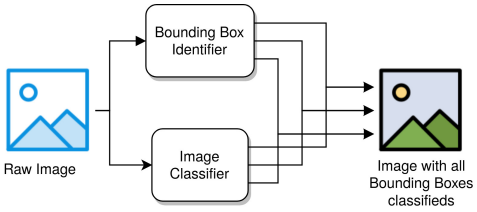


Figure 2: High level Model

1.3 Metrics

The metric proposed by Google in the competition is the mean Average Precision (mAP) [3], a very didactic explanation about the metric could be found in here [4] and [5].

The mAP metric could be defined as

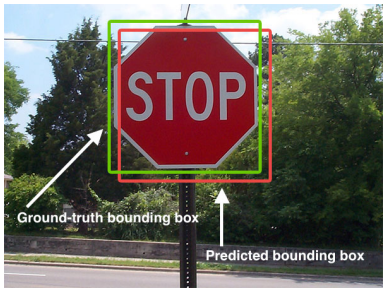
$$mAP = \frac{\sum_{c=1}^{C} AP_c}{C}$$

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

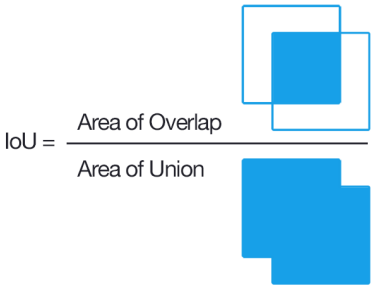
To understand AP_c, it must comprehend first what is IoU. IoU is the Intersection over Union. It is equal to the ratio of the $Area\ of\ Overlap$ and the $Area\ of\ Union$, considering the Predict bounding box (created by the model) and the Ground-truth bounding box (previously annotated).

Here, it is going to define:

$$TruePositive \doteq IoU > 0.5$$
$$FalsePositive \doteq IoU < 0.5$$
$$or\ DuplicatedPredictBoundingBox$$
$$FalseNegative \doteq IoU > 0.5$$
$$and\ WrongClassification$$



(a) Difference between a Predict bounding box with a Ground-truth bouding box [6]



(b) IoU visual represented [6]

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.

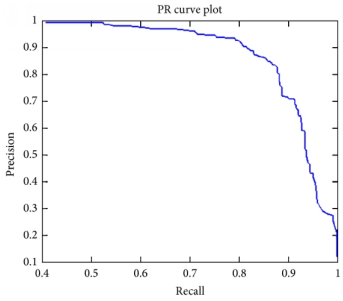


Figure 3: Precision-Recall curve [4]

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.

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 intends 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.

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.

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

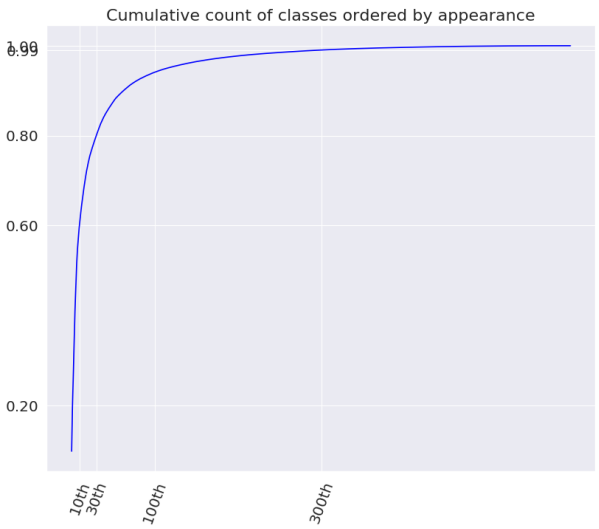


Figure 4: Cumulative count of classes order by appearance

The two datasets, image-level label and bounding boxes, both are divided in three (train, test, cross-validation). Google explains in the blog [7] 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 uses only 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 mis-

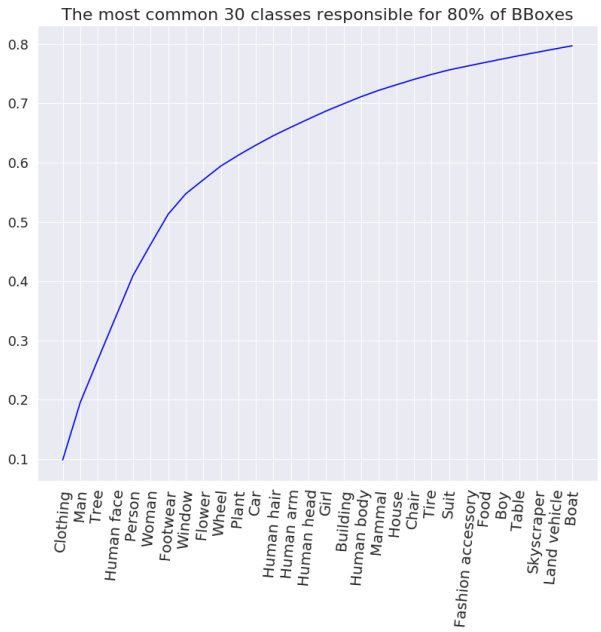


Figure 5: The thirty classes with most appearance

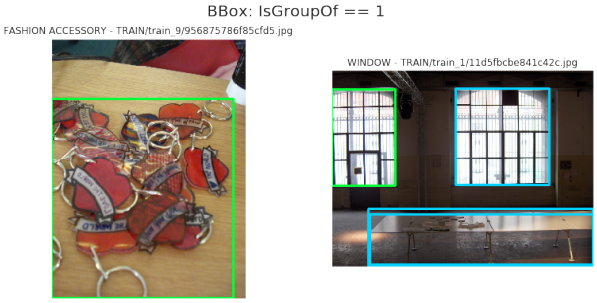


Figure 6: Bounding Box Green represents IsGroupOf True

match and homonyms). In the analysis process they were converted to semantic labels (using a support dataset provided by Google). However, in modeling, it is going to use the encoded version.

Beyond the label and the bounding boxes position, the dataset also 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 made by me, with a study of the correlations between flags and many other topics in all datasets. It can be checked in here [8] on my github.

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 (CNNs) work and how to extract more value of them.

Neural networks could have complex architectures made over a couple of straightforward ideas. First of all, there is the loss function; the loss function is the measure of the error of some neural network, for this project it designs a tailor-made loss function (proposed by SSD paper), which I am going to discuss more further.

Second, in the training process, we have to define a batch-size. Batch size is a number which specifies how many examples we are going to use in the feed-forward process (the predict itself), and how many targets we are going to compare (for obvious reasons these numbers are the same). With the predictions in hand,

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

we compare with targets previous labelled using the loss function. Then, in the process called backpropagation, it is used an optimizer to update the weights of the neural network.

To neural networks to converge, for most of the times, we have to pass the same train set in the model, doing the forward and backpropagation. The Epoch number defines how many times the entire train dataset passes into the network. Besides running many epochs, many times, there is not enough data or data with quality to train a Neural Network. For that reason, we also have to do a process called data augmentation, which consists of augmenting the dataset with more useful examples created by using the original ones.

Convolutional neural networks (CNNs) are neural networks with convolutional layers. These are layers that identify patterns in the images. In the first levels, they find the most common patterns, as edges, and in the 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 purpose of considering only the core information for the deeper layers. [9] AlexNet [10] was the first famous CNN architecture to classify images. Many others come later derived from it and newer ones, like VGG [11], ResNET [12], Inception [13] or Xception [14].

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, [15] and the goal is to achieve the best score that I can do, with the resources and the time that I have available.

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

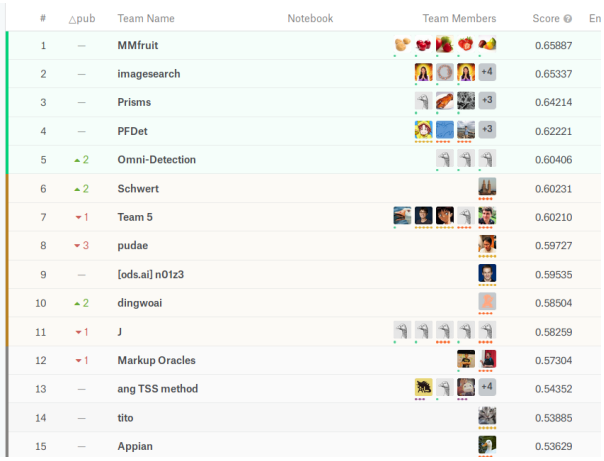


Figure 7: Leaderboard Image Dection Kaggle Competition - 2019

3 Methodology

3.1 Data Preprocessing

There are three processes made in data overall. The first one is a simple One Hot Encode in the categorical feature that defines the class of the bounding box. One Hot Encode is a technique that maps a categorical variable to many boolean features, as many of the num-

bers of the categories. Each class is recorded as True in the category column created. The second one is data augmentation. Because this dataset is hugely unbalanced, this project uses this approach to reduce the problem. It is defined five levels of data augmentations, as described below, and it is applied only one of that in each image of the dataset. Each level creates newly enhanced images.

1st level - Image flip horizontally = 1 extra image 2nd level - 1st level + randomly change the saturation = 2 extra images 3rd level - 2nd level + randomly change the contrast + flip horizontally and randomly change the saturation = 3 extra images 4th level - 3rd level + nine slight zooms in the image = 12 extra images 5th level - 4th level + nine large zooms in the image = 21 extra images

The SSD paper proposes the image flips and the zooms to augment the data. However, the saturation and contrast changes was an approach inspired by much reading on data augmentation process. By transforming the image with the contrast and saturation, the hypothesis was that the neural network would have different aspects to identify the objects.

The SSD paper proposes the image flips to augment the data. However, the saturation and contrast changes was an approach inspired by much reading on data augmentation process. By transforming the image with the contrast and saturation, the hypothesis was that the neural network would have different aspects to identify the objects. Zoom is also an idea that intended to touch more bounding boxes anchors with data.

The third and last process is data selection to train. The initial train dataset contains more than 1.7 million images, with more than 14 million bounding boxes. After data augmentation, this training dataset reaches 2.7 million images with more than 23 million bounding boxes, which is more balanced, but it is enormous. This project created a subset of train dataset trying to preserve the proportions of the more balanced data, but making it possible to train each epoch in the available time. The flow followed in selecting this data is described next. The train augmented dataset was divided into batches (around 50 images each), and by analysing them, the top 650 ones that contained the most balanced dataset were chosen.

All these processes comprehend many steps, and they are available found in the GitHub. (referencia???)

3.2 Implementation

SSD is a technique which incorporates in the VGG16 a capability to find bounding boxes. To do so, it uses the concept of the convolutional layers. In the paper, they propose to remove the fully connected and final layers and to plug a new architecture. This architecture takes the last feature maps and applies into them a sequence of convolutions (actually, a set of two convolutions, which I am going to describe as a convolutional set). Each convolutional set reduces the features map and passes it to the next one. Also, it is used to predict an object in each kernel application of the convolutional layer. Given that each set reduces the feature map, it increases the length of the possible bounding box that some level could predict. So in the first levels of this architecture, the network predicts the smaller bounding boxes, and in the final levels, the network predicts the bigger bounding boxes.

For each kernel application, we also define some forms of bounding boxes, changing the proportion of the width and size based on the kernel. The paper defines these "default bounding boxes" as Anchors.

All kernels are squares, with 3x3 pixels of the feature map, and depending on the level there are 4 or 6 anchors inside this kernel, including two of them being squares with the proportion

of height per width equals 1. In the cases with four anchors, we also consider the proportion of height and width equals two and 1/2, moreover in the conditions with six anchors we add proportions of height and width equals 3 and 1/3. With an input image with 300x300 pixels, the SSD predicts 8732 bounding boxes (one for each Anchor defined).

To evaluate this, they propose a loss function which mixes the classification loss, using log loss function in softmax of each class and a regressor error metric.

The shape of the predicted tensor for each of 8732 outputs is a tensor with n class + 4 position metrics of the bounding boxes (which SSD paper states to use the position x of the centre, the position y of the centre, width and height). In this project, the output tensor of this architecture have 3-rank, and its shape is ¡batch-size, 8732, (600+4)¿. 600 is the number of the classes in the Open Images dataset.

After that, it was designed a custom Keras layer to the network for when it operates in the inference mode. This layer intends to suppress these 8732 bounding boxes in the 200th most important ones, with the higher metrics evaluated, removing using non-max-suppression with IOU threshold of 0.45 the overlap bounding boxes.

After that, it was designed a custom Keras layer to the network for when it operates in the inference mode. This layer intends to suppress these 8732 bounding boxes using non-max-suppression with IOU threshold of 0.45, keeping in the 200th most relevant bounding boxes with the higher evaluated metrics.

The first part of the research I spent designing the SSD as is, using Keras and Tensorflow. The architecture developed is the same proposed in the paper. Many notebooks were written (insert notebooks links referencias???), and all of them are available on Github. (referencias???)

3.3 Refinement

After implementing the SDD with VGG16 as the paper proposes, I decided to change it and plugged three other networks at the start of the SSD structure, substituting VGG. The other ones are Mobilenet-v2, Resnet50 and Xception.

Each network used weights optimized in the training of the Imagenet dataset as an initial state, and the training process iterated over them, instead of entirely random weights.

In the Evaluation Section are the metrics collected in the cross-validation dataset for each architecture. Still, I decided to show and explain more about the winner, that was the Xception with SSD model.

To understand Xception, we have to understand depthwise separable convolution, which is simply the concept of a channel-wise spatial convolution (which is a convolution not only in the feature map but among the layers) concatenated with a 1x1 convolution called pointwise convolution. These 1x1 convolutions could be understood as a type of pooling over the channels, and they intend to reduce the number of layers. The Xception proposes to revert the order of the depthwise convolution with the 1x1 convolution. The main argument is that this reduces the weights of the model considerably with the same accuracy.

The following image illustrates what this project implements, plugging the first architecture proposed in the final three sets of depthwise-pointwise.

4 Results

4.1 Model Evaluation and Validation

To the final results, many models were trained. All of them followed the same pattern, they used the partial train dataset created, the 35k images (650 batches of 50 images),

and they were evaluated in the cross-validation dataset. After some interactions trying to evaluate many models, the amount of epoch chosen was 30, with a batch size of 24. SGD (Stochastic Gradient Descent) as the optimizer, with momentum of 0.9, decay of 0.0005 and a learning rate scheduler that uses 10^-3 in first 15 epochs, 10^-4 in the epochs 16 to 25, and 10^-5 in the last five epochs.

This work presents the best four models created with this evaluation model. The cross-validation dataset has 570 classes, in the following table are the mAP achieved for each model, I also considered the weighted mAP, which is the weighted mean AP, and the weights are the amount of the bounding boxes of each class. The weighted mAP is calculated because the dataset is very unbalanced, and the regular mAP achieved is very low.

[TABELA]

The two best architectures, Resnet50 and Xception, were reloaded with the weights obtained and retrained for more 30 epochs. And the following table shows the final results in the cross-validation.

[TABELA2]

With this evaluation, Xception + SSD is the best one, and the evaluation of the model in the test set is:

4.2 Justification

The final model, considering the weighted mAP tell us that it predicts correctly 17

But after all, this reflects the types of photos that this model is going to evaluate. How many are the images with crabs passing to the model in comparison with pictures of people?

I believe that filtering the dataset and working with fewer classes and balanced occurrences, this model can deliver a much better result.

It is not recommended to use a model with this mAP on a critical system, like self-driving cars. But for a mobile application used to identify objects in user photos is an acceptable MVP (Minimum Viable Product), even more, if the project reduces the classes scope to recognize fewer categories.

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

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

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

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

section-over-union-iou-for-object-detection/.

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

[8] 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.

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

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

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

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

[13] 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.

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

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

A Class hierarchy

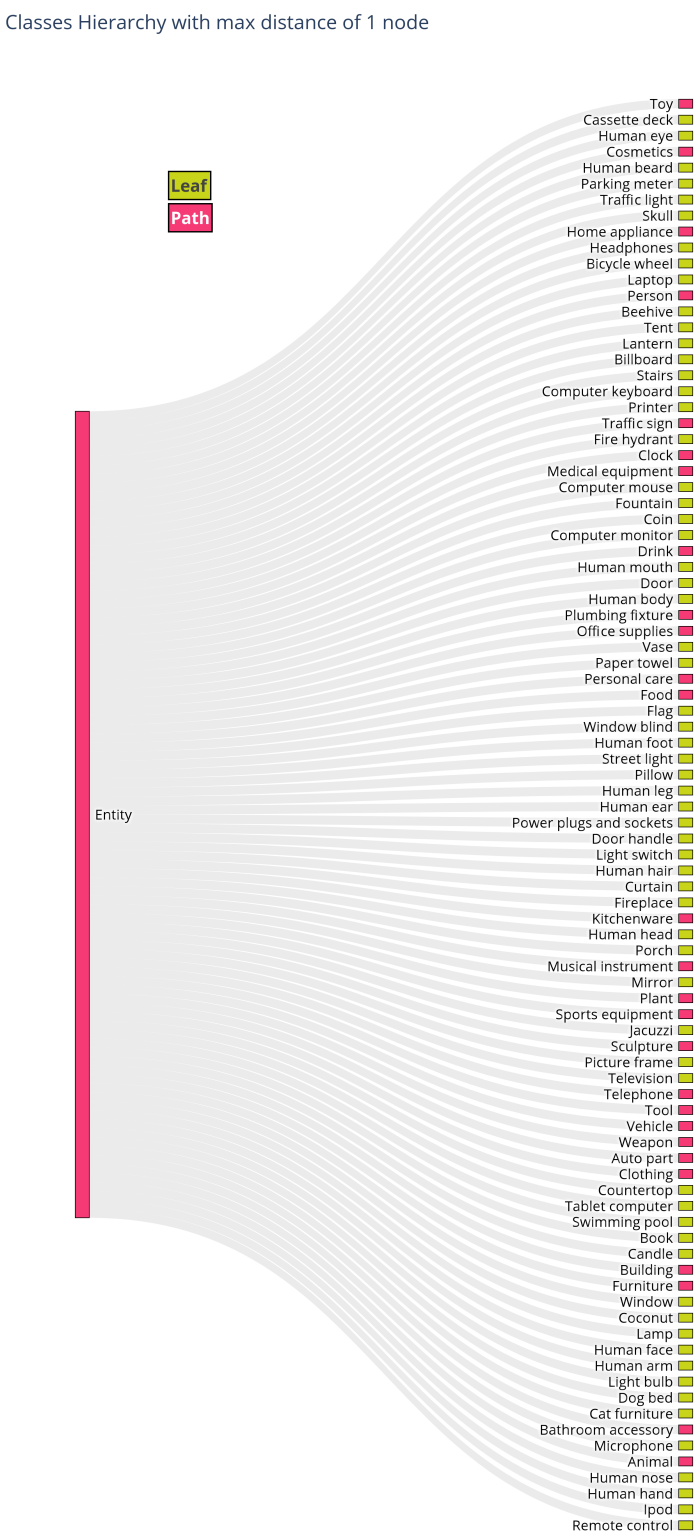
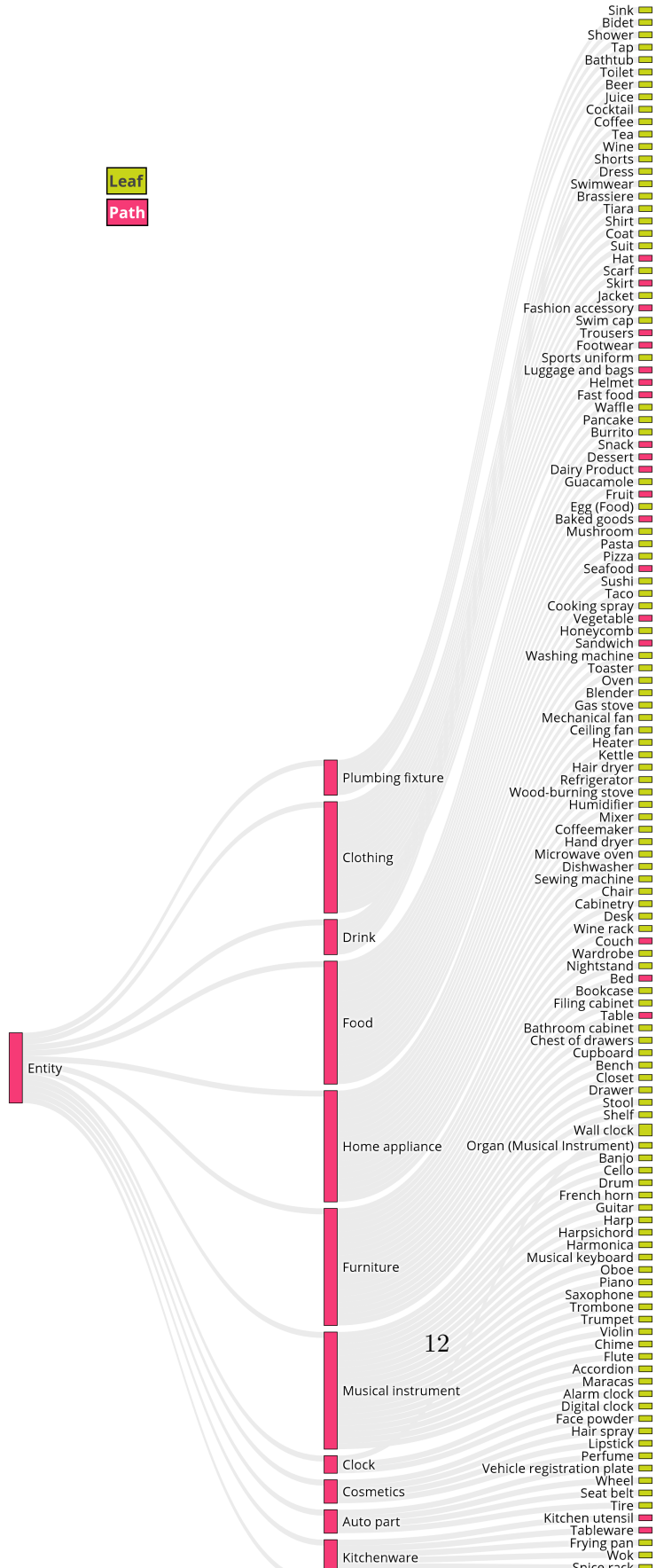
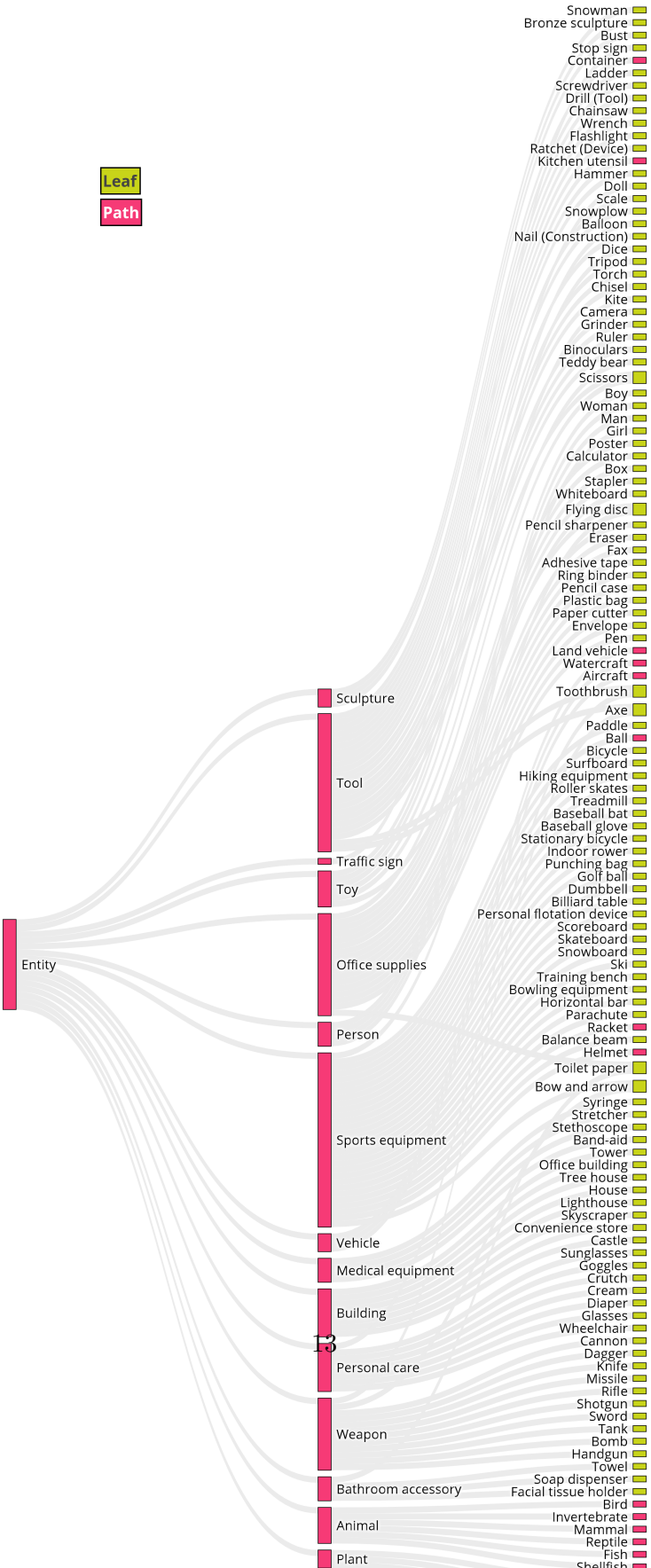


Figure 8: Class Hierarchy - First Level

Classes Hierarchy with max distance of 2 nodes - part 1



Classes Hierarchy with max distance of 2 nodes - part 2



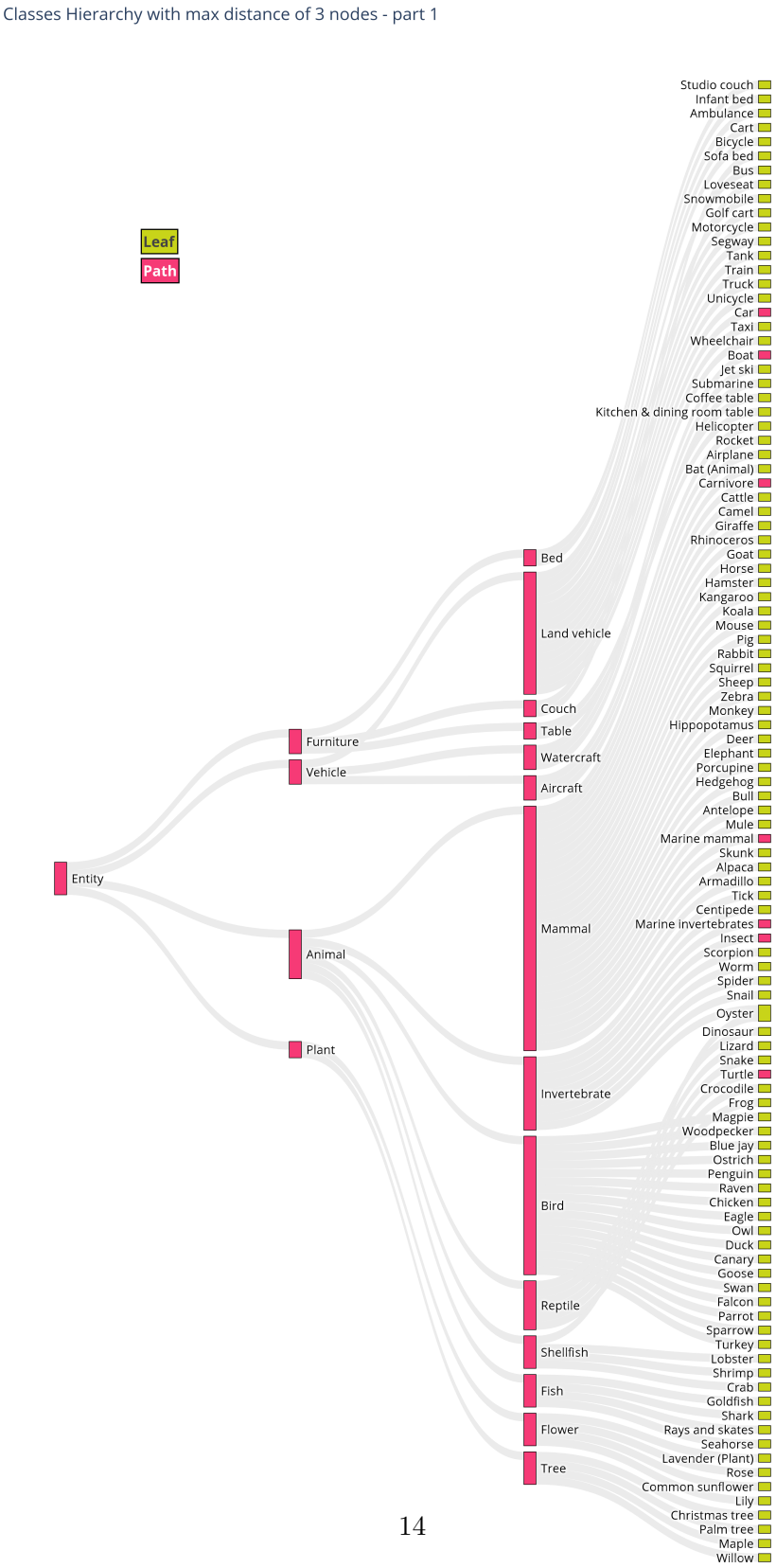


Figure 11: Class Hierarchy - Third Level first part

Classes Hierarchy with max distance of 3 nodes - part 2

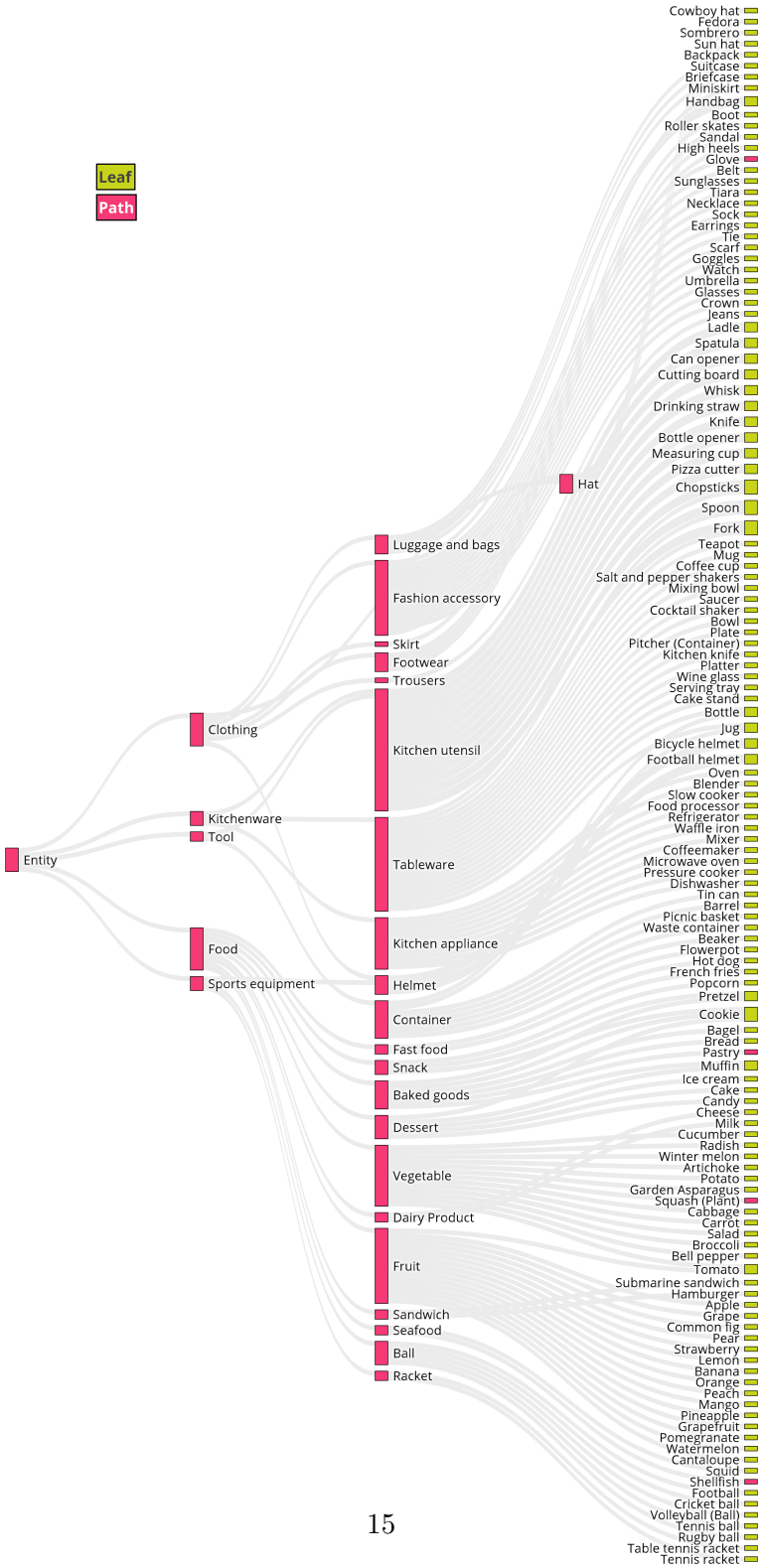


Figure 12: Class Hierarchy - Third Level second part

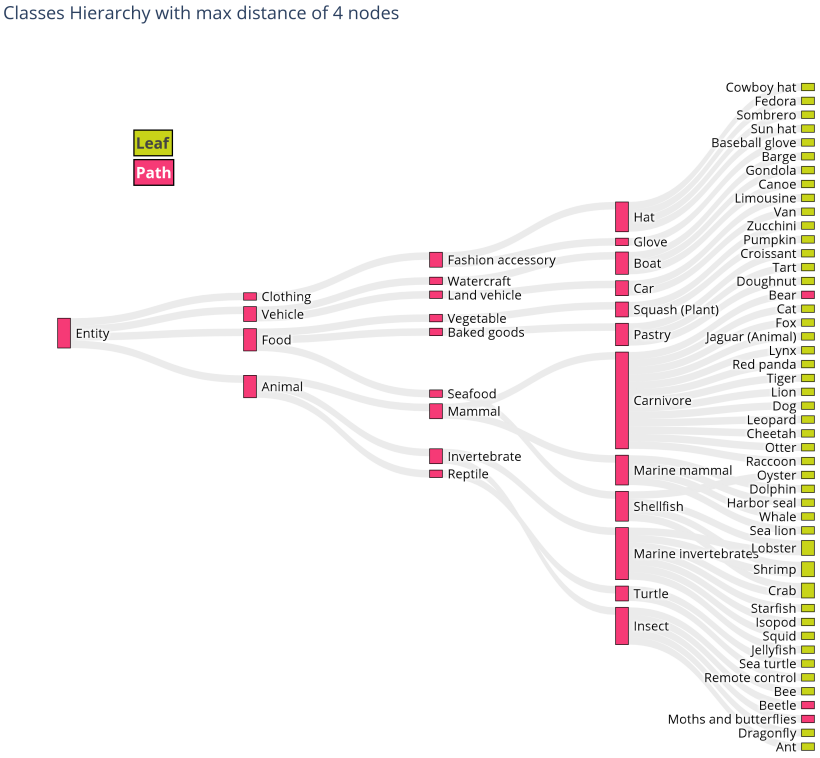


Figure 13: Class Hierarchy - Fourth Level

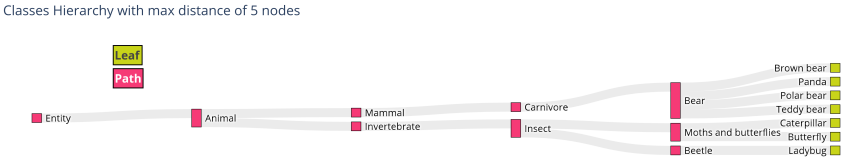


Figure 14: Class Hierarchy - Fifth Level