Project Report
Department of Computer Science
Faculty of Sciences
University of Porto

On using Deep Learning for Automatic Taxonomic Identification of Butterflies

Tomás M. S. Mamede

Supervisors:
Luís M. B. Lopes, Associate Professor
Eduardo R. B. Marques, Assistant Professor

2019/2020

Contents

Abstract

In this project we derived and deployed a Machine Learning model in a Web application to perform taxonomic identification of butterflies. This involved creating a data set with thousands of images, the use of Deep Learning and Computer Vision tools to derive different models and the evaluation of the results using different metrics, such as precision and recall. We found that some models performed better for some genus, while others did not present relevant changes or even performed worse. Here we demonstrate how to perform this experiment and show the results we obtained. We proved that today Artificial Intelligence and more specifically, Computer Vision through Deep Learning, is accessible and can create impact in a great number of areas and scientific fields.

1 Introduction

In the last two decades, Artificial Intelligence (AI) has established itself as one of the technologies with the largest social impact. Examples of major breakthroughs are self-driving cars, natural language translation and expert systems for medical diagnostics. The major challenge in AI is trying to make computers perform tasks that are very straightforward for people to do, but very hard to describe formally. In others words, computers have difficulty learning how to do simple things that are intuitive for human beings, e.g., recognising spoken words or faces in images. To solve this problem, one of the solutions suggests that AI systems need the ability to acquire their own knowledge, by extracting patterns from data. This capability is known as Machine Learning.

Here we report the use of Machine Learning techniques, namely Deep Learning, via Google's AutoML Vision framework, to perform automatic taxonomic identification of Portuguese species of butterflies from images. The model was later integrated in a Web application so that users could identify different species of butterflies using just a photograph. The project was divided in three different phases. First, we built a data set with thousands of images of butterflies identified by specialists to train the Machine Learning model. After that, we trained the model, we evaluated it using a data set of test images and further optimised it. Finally, we deployed the resulting model in an interactive Web application so that anyone can upload an image of a butterfly and receive back its taxonomic classification.

A demonstration video of the Web application is available on Youtube and can be accessed using this link. In the Zip file containing this report, the *url* for the Web application and the access credentials are also available in the *readme.txt* file. We also included images of butterflies for testing of the Web application.

2 Background

Here we introduce the reader to some of the subjects under consideration in this work to improve readability of the document.

2.1 Lepidoptera

Butterflies (Rhopalocera) and moths (Heterocera) are the adult stage of insects belonging to a group called Lepidoptera. The word itself has its origins in Ancient Greek from *lepidos* (scale) and *ptera* (wing), meaning "wings with scales", a reference to the fact that the wings of butterflies and moths are actually covered with thousands of tiny scales overlapping in rows. As all insects, butterflies have six legs and three main body parts: head, thorax and abdomen. This group also possesses sophisticated sensing organs: the antennae, the palps (modified mouth parts), and the feet. With these, butterflies are able to sense gradients in the concentrations of certain molecular compounds allowing them to find food, sexual mates and laying eggs with high accuracy [8].

Butterflies are holometabulous insects, meaning that they undergo full metamorphosis during their life cycle which involves four different stages: egg, larva, chrysalis and adult butterfly or imago (Figure 1). This complex life cycle was first described by the early entomologist Maria Sybilla Merian [15].

Butterflies are complex insects and a keen observer of their behaviour may see them involved in many activities, like basking, puddling, feeding, mating, chasing rivals and egg laying.

Butterflies play an essential role in the natural ecosystem as pollinators and as food for other animals in all stages of their life cycle. Their intricate relationships with plants and animals means that they are often the first to be endangered if an ecosystem's delicate balance is disturbed. Thus, the occurrence of certain species and their numbers are often used as indicators of ecosystem health [6].

Butterflies are also important research subjects and case studies in areas such as evolution, genetics, population dynamics and conservation, a fact that has been known since the works of naturalists such as Alfred Russel Wallace, Henry Walter Bates and Fritz Müller [17, 2, 11].

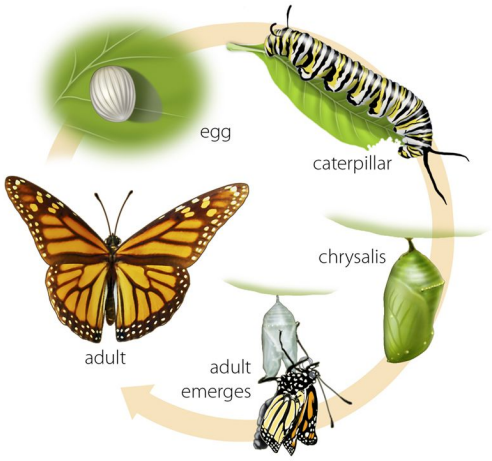


Figure 1: Scheme of the life cycle of a butterfly.

Given their characteristic and often stunning wing patterns and colours butterflies attract attention and therefore there are countless photos available online documenting sightings, mostly from non-experts. This makes butterflies a perfect subject for a project that aims at using Deep Learning techniques to perform automated taxonomic classification. However, while there are thousands of described species of butterflies from around the world, in this work we focus solely on the approximately 150 species that can be observed in Portugal [9, 4].

2.2 Neural Networks and Deep Learning

Neural Networks were first proposed in 1944 by Warren McCullough and Walter Pits, two researchers at the University of Chicago, that later became the founding members of the first cognitive science department at MIT in 1952 [10].

Later, in 1958, the perceptron developed by Frank Rosenblatt [13] emerged as a new model of the neuron and became the simplest Artificial Neural Network capable of linearly separate data in two classes [1]. The simplest type of perceptron has a single layer of weights connecting the inputs and output. Later appeared the multi-layer perceptron, consisting of a sequence of layers each fully connected to the next one. This was the beginning of the development of Neural Networks.

Even as it became a major area of research in both Computer Science and Neuroscience, as years and decades went by, Neural Networks saw good and bad days. They were first dismissed as a promising area of research in 1969. Then, in the 1980's they enjoyed a resurgence just to be forgotten once again in the first decade of the twenty-first century.

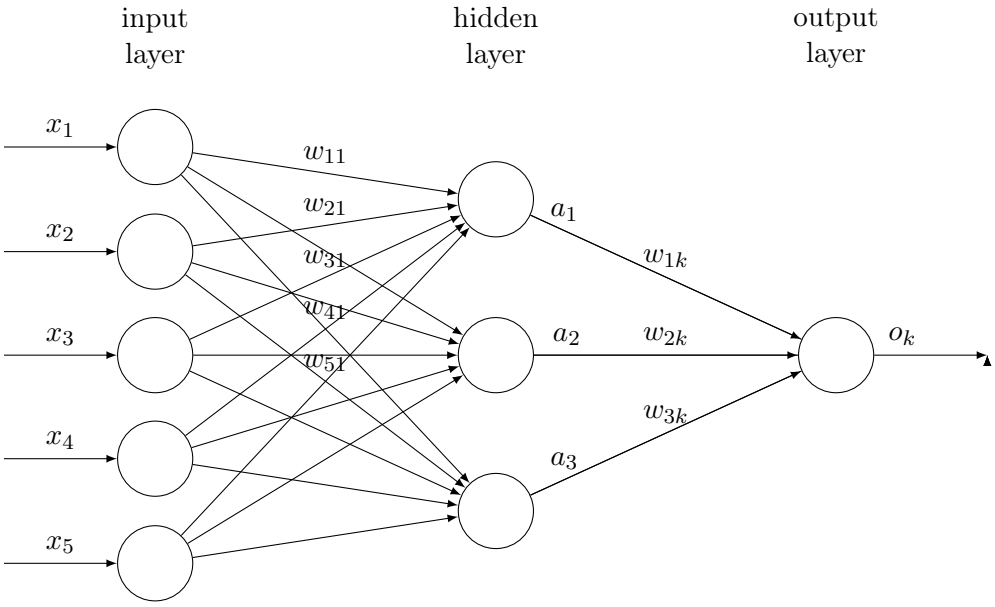


Figure 2: A portion of a 2-layer feed-forward neural network.

Today, Neural Networks are a means of doing Machine Learning, in which a computer learns to perform some tasks by analysing training examples [5]. An object recognition model, like the one used in this project, might be fed thousands of images of the object of study, so it would find visual patterns that characterise the images in the data set.

Neural Networks are composed by thousands or even millions of simple nodes or processing units

that are densely interconnected, mimicking neurons in the brain, hence the name Neural Network. In Figure 2 can be seen an example of a Neural Network. Each edge from a node i to a node j serves to propagate the activation from i to j. Each edge also has a numeric weight w associated with it, which determines the strength and sign of the connection [18]. A node j computes a weighted sum of its inputs (thus a linear transformation) and then applies a non-linear activation function f, g in Figure 2, corresponding to the following outputs for the nodes in the hidden layer and output layer, respectively:

$$a_j = f\left(\sum_{i=1}^{N} w_{ij} x_i\right) \qquad o_k = g\left(\sum_{j=1}^{M} w_{jk} a_j\right)$$

In this example, $N = 5$ and $M = 4$ and the depth of the network is 2 (the input layer does not count). In general, the number of nodes per layer and the number of layers is arbitrary and, in the case of Deep Learning networks, it can be quite large.

When a Neural Network is being trained, all of its weights and thresholds are initially set to random values. Training data is fed to the input layer and it passes through the succeeding layers until it arrives at the output layer. During training, the weights and thresholds are continually adjusted until we get similar outputs from training data with the same labels.

To form a network, nodes need to be connected together, as described in the image above. Most of modern Neural Networks are organised into layers of nodes. An individual node might be connected to several nodes in the previous layer, from which it receives data, and several nodes in the next layer, to which it sends data. In fact, Neural Networks can be single-layer, in which every node connects directly from the network's inputs to its outputs, or multi-layer, which have one or more layers of "hidden" nodes between the input and the output layers of the Neural Network.

The Feed-forward Neural network was the first and simplest type of Artificial Neural Network to be invented. In this kind of Neural Networks, the information moves through the hidden nodes (if they exist), towards the output nodes. The perceptron can be considered the simplest kind of Feed-Forward Neural Network.

The development of the back-propagation algorithm by David Rumelhart, Geoffrey Hinton and Ronald J. Williams began a new generation of Neural Networks[14]. This algorithm is used for tuning the weights of the edges of a Neural Network based on the error rate obtained in the previous iteration. The proper tuning of the weights allows to reduce error rates and make the model more accurate. It allowed to solve the XOR problem, which for many years had made lots of scientist lose faith in this type of approach to Artificial Intelligence.

In the last decade, Neural Networks became one of the most promising and exciting areas of research in science thanks to the conceptual foundations and engineering advancements laid by Geoffrey Hinton, Yoshua Bengio and Yann LeCun, receivers of the ACM A. M. Turing Award in 2018, and the breakthroughs in graphic processing units (GPUs)[3].

Neural Networks allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined in terms of its relation to simpler concepts.

By gathering knowledge from experience, this approach avoids the need for computer scientists to formally specify all the knowledge that the computer needs. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simple ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach Deep Learning [5].

Deep Learning is now one of the most promising and exciting areas of research in science, and led to great advancements in technologies such as Computer Vision, Speech Recognition and Machine Translation, which can be used nowadays by anyone with a modern smartphone.

2.3 Convolutional Neural Networks

Convolutional Neural Networks have played an important role in the history of Deep Learning [7]. They are a key example of a successful application of new insights to Machine Learning applications, obtained through the study of the brain. They were also some of the first working Deep Neural Networks trained with back-propagation.

Convolutional Neural Networks (CNNs) are a specialised kind of Neural Network for processing data that has a known grid-like topology. One of the most common examples of this would be image data, which can be thought of as a grid of pixels.

These networks ingest and process images as tensors, which are matrices of numbers with additional dimensions. In a computer, images are stored as a 2-dimensional array of pixels arranged spatially, where each pixel corresponds to a very small part of the image. Each pixel in the 2-dimensional space contains three numbers from 0-255 (inclusive) corresponding to the red, green and blue channels, hence RGB.

The name "Convolutional Neural Network" indicates that the network employs a mathematical operation called *convolution* [18]. In this context, a convolution is a specialised kind of linear operation that takes as arguments an array of input data and an array of weights that we multiply together. In a Convolutional Neural Network terminology the second argument is known as the kernel or filter. The result of the operation is sometimes referred to as the *feature map*. Lets consider the following matrices representing an image I and kernel K:

$$I = \begin{bmatrix} 17 & 14 & 13 & 9 & \dots \\ 21 & 64 & 62 & 41 & \dots \\ 42 & 54 & 61 & 52 & \dots \\ 41 & 30 & 31 & 34 & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix} \qquad K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In practice, in order to perform a convolution on an image we center the kernel at each pixel location, multiply the weights of the kernel by the corresponding pixels and sum all the obtained results. Finally, we override the pixel in the center of the kernel with the output of the kernel convolution. The convolution function with a kernel K over a matrix I, denoted $C = \text{Conv}(I, K)$, where K has dimensions $(2M + 1) \times (2N + 1)$, is defined as follows for every pixel (i, j) in I as:

$$C(i,j) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} I(i-m, j-m)K(n,m)$$

As an example, lets consider the image above image I and kernel K. Depending on the values of each element, the kernel can cause a wide range of effects. The evaluation of this convolution for pixel $(1,1)$ is given by:

$$\begin{bmatrix} 17 & 14 & 13 \\ 21 & 64 & 62 \\ 42 & 54 & 61 \end{bmatrix} \times \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{9} (17 + 14 + 13 + 21 + 64 + 62 + 42 + 54 + 61) = 39$$

As we just saw, convolutions are used to apply different effects to images. One of the most important effects used in Computer Vision, to extract useful features from images, is edge detection (Figure 3). Edges in images come from a variety of factors, as, for example, surface normal discontinuity, depth discontinuity and illumination discontinuity, just to name a few; and they allow us to understand the essential features that characterise an object in an image. For the computer to be able to learn and make sense of them all, we use Convolutional Neural Networks that hopefully, after having been trained, will learn how to classify different objects [12].

Figure 3: Edge detection convolution applied to an image of a butterfly.

In a rough sense we can think of a Convolutional Neural Network as consisting of two parts: a block of various convolutional layers and block of fully connected layers that make up a multi-layer Neural Network, as depicted in Figure 4. A typical layer of a Convolutional Neural Network, used to recognise patterns in the input images, consists of three different stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations, which in the second stage are run through an activation function. In the last stage, a pooling function is used to modify the output of the layer further, by grouping up the pixels in the image and filtering them down to a subset. The image is reduced, but the features are still maintained.

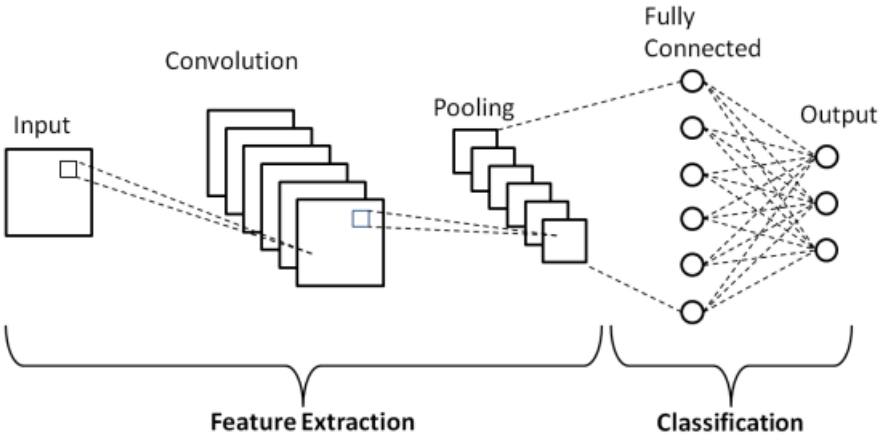


Figure 4: Representation of Convolutional Neural Network.

Like the convolution operation, pooling consists of a fixed-shaped window that is slid all over the input, computing a single output for each location. At each location that the pooling window hits, it computes the maximum or average value of the input subarray in the window. The result of applying 2×2 maximum pooling to matrix I is given by:

$$\text{maxPooling}_{2\times 2}\left(\begin{bmatrix} 17 & 14 & 13 & 9 & \dots \\ 21 & 64 & 62 & 41 & \dots \\ 42 & 54 & 61 & 52 & \dots \\ 41 & 30 & 31 & 34 & \dots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}\right) = \begin{bmatrix} 64 & 64 & 62 & \dots \\ 64 & 64 & 62 & \dots \\ 54 & 61 & 61 & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

Pooling can also change the output shape by padding the input and adjusting the stride. Each layer uses a kernel and processes each output with an activation function. Pooling is used to alleviate the excessive sensitivity of the convolutional layer and summarise the presence of features after the convolutional operations. As the images are fed into the Convolutional Layers, a number of randomly initialised kernels, will pass over the image. The results are then fed to the next layers and matching is then performed by the Neural Network. And over time, as more images go through the Convolutional Neural Network, the kernels that return the outputs that give the best match will be learned. This process is called Feature Extraction.

2.4 AutoML Vision

AutoML Vision is a Machine Learning model builder for image classification that is part of the Google Cloud Platform. Building a new model for Computer Vision and image recognition can be a herculean task and programming it by hand is both hard and time consuming.

Auto ML Vision enabled us to perform supervised learning. This involved training a Convolutional Neural Network to recognise different species of Portuguese butterflies from labelled data. In order

to train the model we had to provide images of butterflies labelled by species and the categories or labels we wanted the model to predict.

The data set (see Figure 5) provided to AutoML Vision contained training, validation and testing sets. We didn't really have to worry about making this distinction because the tool can handle it automatically. The only thing we had to make sure was that for each butterfly species we provided at least ten images. For each of the species, AutoML Vision automatically used 80% of the images for training, 10% for validation and 10% for testing.

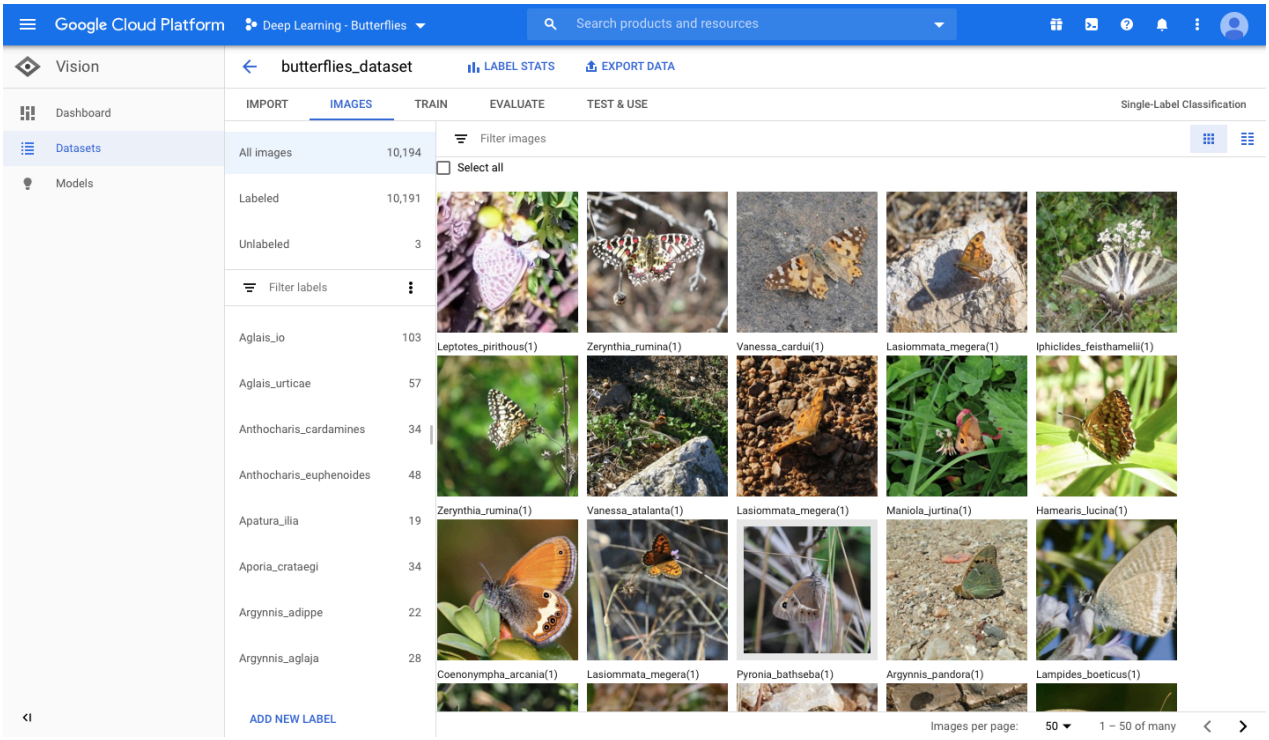


Figure 5: Screenshot of the Google's AutoML Vision.

Thus, the vast majority of the data goes into the training set. This corresponds to the data that the Convolutional Neural Network actually sees during training and uses to learn the weights of the connections between the nodes in the network. On the other hand, the validation set, also used in the training process, is used to tune the variables that specify the structure of the model, so that it can generalise better. Finally, the testing set is not involved in the training process at all. Once the model has completed its training entirely the test set is used to measure its accuracy.

The AutoML Vision tool provides a section named *Evaluate*, as show in Figure 6 so that we can assess the performance of the model using its output on test examples and common Machine Learning metrics. For each example, the models output a series of numbers that communicate how strongly it associates each label with a given image. These numbers represent how certain the model is that a specific label is the correct one.

AutoML Vision also allows the user to change the score threshold using a visual tool, which refers

to the level of confidence the model must have to assign a category to an image. If the score threshold is low the model will classify more images, but with the risk of misclassifying a few of them in the process. On the other hand, if the score threshold is high, the model will classify fewer images, but it will have a lower risk of misclassification.

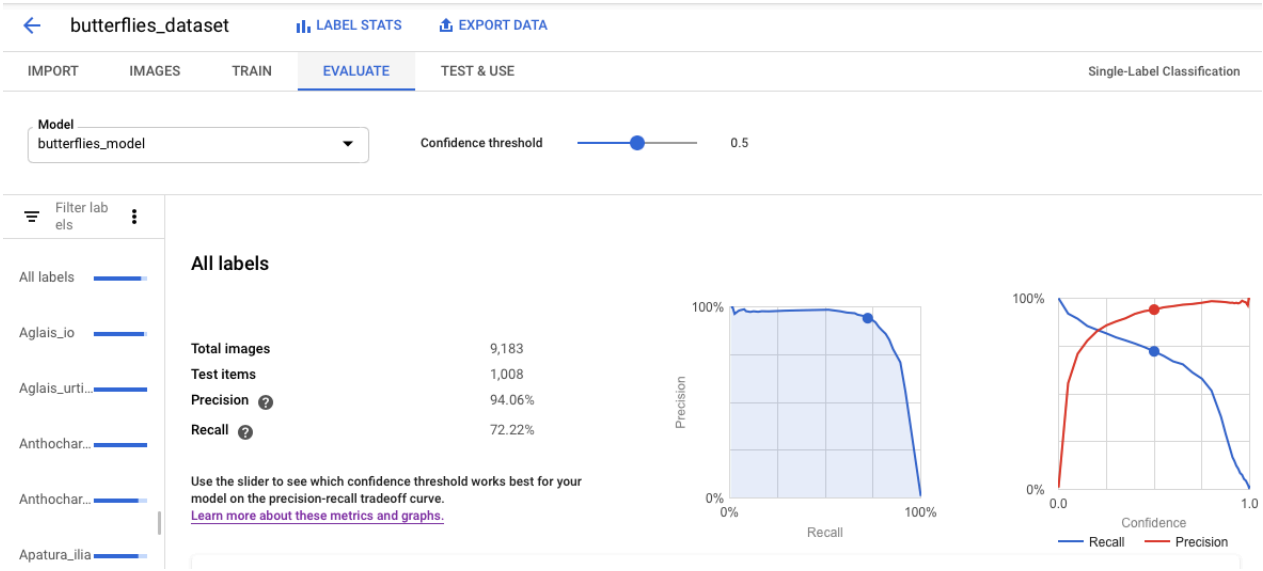


Figure 6: Screenshot of the Google AutoML Vision - Evaluation.

After applying the score threshold, the predictions made by the model will fall in one of four categories: true positives, true negatives, false positives and false negatives. The categories are then used to calculate the *precision* (positive predictive value) and *recall* (sensitivity) of the model. These metrics help to evaluate the effectiveness of the model.

From all the test examples that were assigned to a label, precision tells us how many were correctly assigned to it. In other words, precision is the number of correct results divided by the total number of results. Higher precision means less false positives. On the other hand, recall tells us, from all the test examples that should have been assigned to that label, how many were actually assigned the label, i.e, the number of correct results divided by the number of results that should have been returned. A higher recall means that the model produces less false negatives. So, basically, precision and recall helped us understand how well the model is capturing the information.

Another important evaluation metric provided by the AutoML Vision tool is the Confusion Matrix. Part of it can be seen in Figure 7. With it we can compare the models performance on each label. In an ideal model, all the values on the main diagonal will be high and the other values will be low. When values outside of the diagonal are higher, that means the model is misclassifying categories.

Auto ML allowed us to focus on results and speed of progress, without worrying to much about programming and the specifications of the Neural Network. We were able to take advantage of the evaluation metrics provided by Google to know what to expect from the model we trained and then make improvements to get better results when trying to classify different species of butterflies.

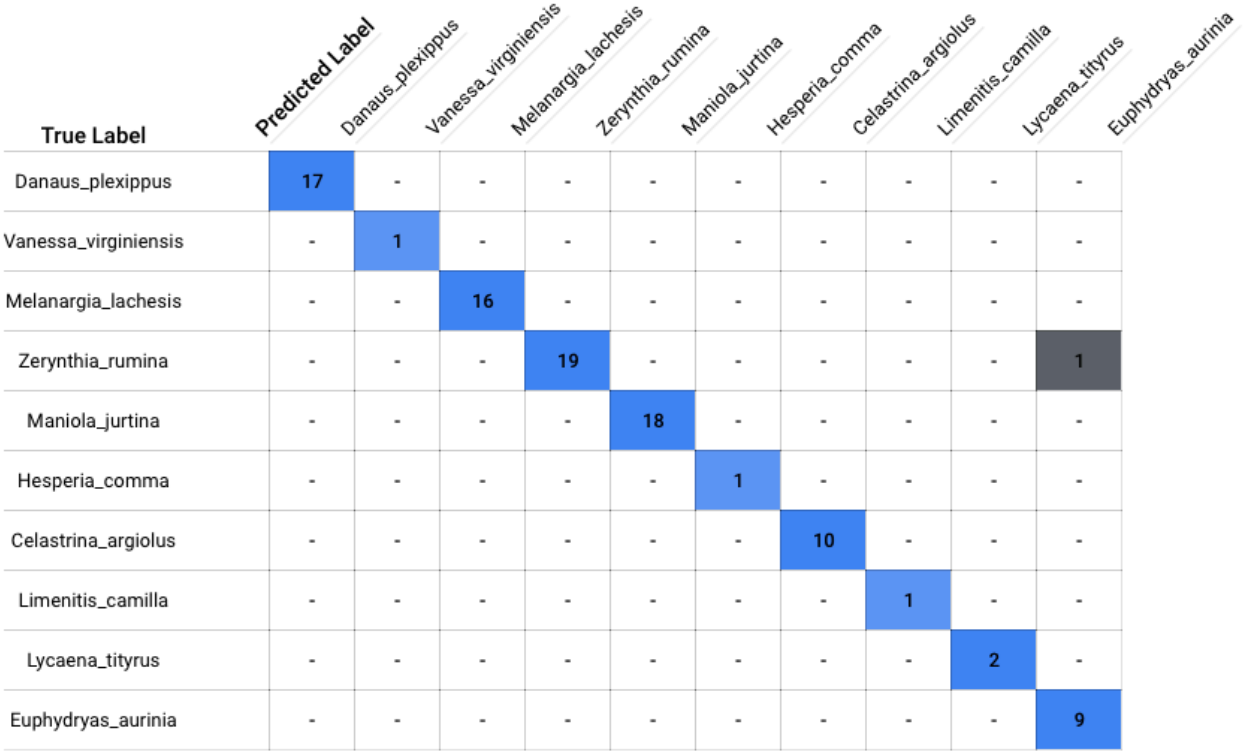


Figure 7: Fraction of the confusion matrix provided by AutoML Vision.

2.5 TensorFlow

TensorFlow is an Open-Source software library for numerical computation of mathematical expressions that works with data flow graphs. It was developed by Google Brain for internal Google use and then released for the public under the Apache License 2.0 in 2015. Currently, TensorFlow is one of the most used software libraries for Machine Learning and it has a large variety of applications.

TensorFlow allows programmers to create a graph of computations to perform. Each node in the graph represents a mathematical operation and each connection represents data. This creates a comfortable level of abstraction that allows the user to focus on the overall logic of the application. All computations associated with TensorFlow involve the use of tensors. A tensor is a n-dimensional vector or matrix that is able to represent various types of data. Values stored in a tensor have identical data types. Vectors are one-dimensional tensors, matrices are two-dimensional tensor; a three-dimensional tensor is a matrix of matrices and so on and so forth. In the graph, computations are made possible through interconnections of tensors. TensorFlow takes an input in the form of an n-dimensional matrix that flows through a series of operations producing an output at the end. Hence the name TensorFlow.

In the context of this project we used TensorFlow Lite to run inference with the model we obtained from Google AutoML Vision. Inference is the process of running data through the model to

obtain predictions. This tool was used later in the project to evaluate the produced model. We used TensorFlow Lite Interpreter library. This library takes a model file, executes the operations it defines on input data and produces an output. In this case the input is an image of a butterfly and the output is the string representing the species of the butterfly in the image. The TensorFlow Lite Interpreter works across multiple platforms and provides a simple API for running TensorFlow models from Java, C++, Python, Swift and Objective-C.

3 The Data Set

In this section we describe how we built the data sets used for training and testing the model.

3.1 GBIF and iNaturalist

The first and one of the most important steps of building a Machine Learning model for Computer Vision is the construction of a good data set. One that, not only contains lots of examples to train the Neural Network, but that is also diverse and well balanced so that each output label gets enough images. The decisions made early when creating the data set would later influence the final results of the model.

In Portugal there are around 150 different species of butterflies that can be seen during the day. There isn't a clear consensus and not even specialists know for sure how many there are, as records differ from source to source. This can be seen as something quite normal in this area, in great part due to the possibility of DNA analysis.

Having this in mind, we ended up working with a data set of 135 species of butterflies. To get the images for all the species we used iNaturalist and GBIF. The first one is one of the most famous nature applications (for Web and Mobile) in the world that helps anyone identify different animals and plants while, at the same time, also creates quality data for research. On the other hand, GBIF is a network and research infrastructure that provides open access to data about all types of life on Earth.

In this project, GBIF was an important tool that allowed us to download the images. Using the website (see Figure 8) we were able to specify exactly the species and the countries from where we wanted results. We were also able to select the main data set from where GBIF would select images. For this last query field we selected iNaturalist.

We started out by downloading the data set files for all the images of butterflies from Portugal and Spain available through GBIF, separately. These files included .txt files with metadata about the data set for each of the countries, e.g., the ID of the observation on GBIF, the *url* of the image on iNaturalist, the date of the observation and location of the observation and the scientific name of each species In GBIF, the GBIF ID corresponds to one observation. Each observation can have more than one image associated to it.

From Portugal we obtained references for 5028 different images. On the other hand, from Spain this number was much larger and we ended up with 14059 additional images, for a total of 19087

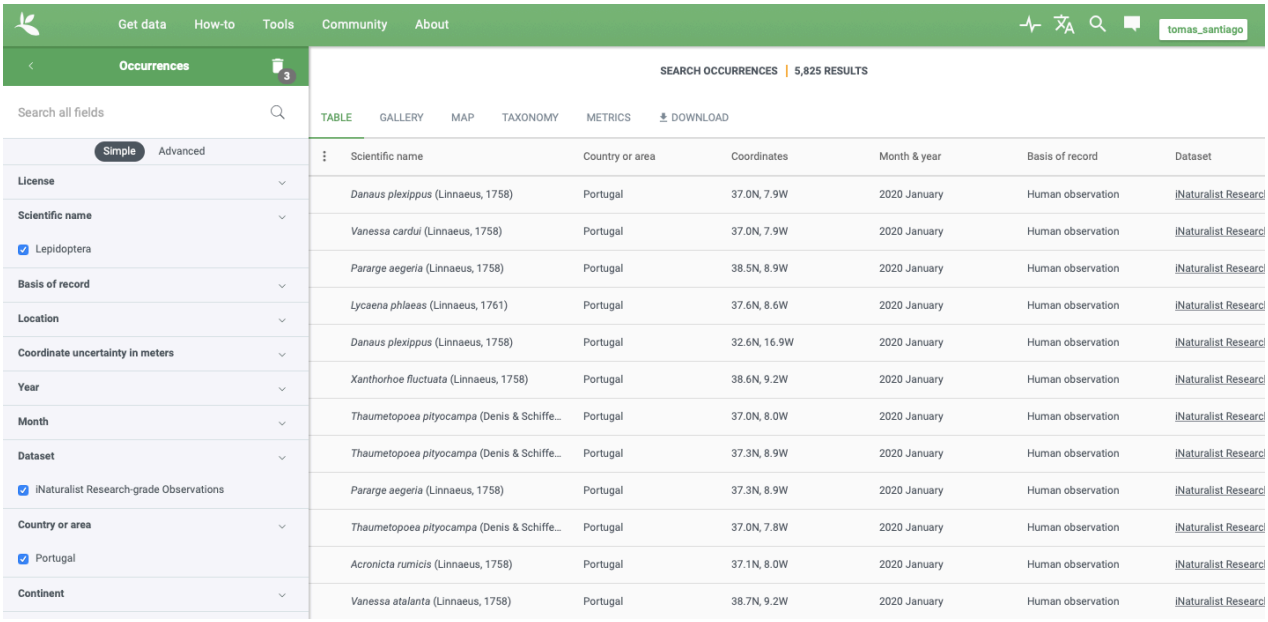


Figure 8: Screenshot of the GBIF web interface.

images. We decided to include images from Spain in the data set in order to complement the ones we obtained from Portugal. We figured out that using images from Portugal would not be enough to train the model and get good results. Because Spain is a neighbour country of Portugal there are many species that can be found in both countries.

3.2 Filtering the results and obtaining the images

We started this next stage of the construction of the data set with data from the aforementioned countries. For each of this countries, we got two different *.txt* files from GBIF: *occurence.txt* and *multimedia.txt*. These two files combined provided all the information we needed to be able to filter the observations recorded in GBIF and later obtaining the images of the butterflies.

The file *occurence.txt* provided information about each of the images of butterflies from Portugal and Spain that we would get from iNaturalist. In this file an occurrence corresponds to an observation of a butterfly, which has extra information related to it, e.g., date, references in iNaturalist, rights holder for the images related to the observation, country, district, latitude, longitude and scientific name of the species recorded in the observation.

On the other hand, the file *multimedia.txt* is more specific and provides information for each photograph associated with an observation. This includes GBIF ID, type and format of the image, *url* to the image and rights holder. Differently from what we had in the *occurrences.txt* file, where each line had a different GBIF ID, in this one, most often that not, we find that different lines have the same GBIF ID. This happens because each observation can have more than one image associated to it.

To make sense of all the information in the two files we started by converting each of the files to the *CSV* format, both for readability and ease of use. Next, we used Python and the Pandas library for data analysis and manipulation [16]. The goal was to merge the *occurrences.txt* and *multimedia.txt* into a single *CSV* file. We did this using the GBIF ID key so that we could combine all the information from the two files and add the columns that mattered the most for each image of a butterfly: family and genus of the butterfly, species, decimal latitude and longitude and the *url* of the photo. Each line in these two files represented a different image of a butterfly we would later add to the data set.

However, not all the images were relevant to the project. We only worked with species of butterflies that can be observed in Portugal. To filter the results by species we created a new file where we listed all the species we where interested in and, once again, we used Python and the Pandas library to obtain a new CSV file, ordered by GBIF, with the references to all the images we would later download.

After filtering the results from GBIF regarding Portugal, we ended up with with references to 3063 different images. We repeated the process described in the last few paragraphs with the data from Spain that was provided by GBIF. As a result, we added more 7722 references, which sums up to a total of 10785 different references to images.

Table 1: Final CSV layout with *urls* for images.

gbifID	species	photoURL
2269308307	Aglais io	https://static.inaturalist.org/photos/42395386/...
2311381868	Argynnis paphia	https://static.inaturalist.org/photos/46542761/...
2573849394	Colias croceus	https://static.inaturalist.org/photos/61093825/...
...	...	...
...	...	...

To download and save the images we used Python combined with two different libraries: PySpark and the Google Drive API. The first one allowed us to read CSV files and infer its structure to be able to read the necessary information stored in each column. To download each image we needed access to the *url* and to name each of the downloaded images, we used the species name combined with the GBIF ID. We also used a counter to keep track of the number of images for each observation and we concatenated that to the end of the image name. So, for example, the name of a image in the data set might be `Zizeeria_knysna_2445096096_1.jpg`.

As we dealt with a considerable amount of data we had to pay attention to storage management. We wrote our script to be able to upload each downloaded image directly to Google Drive so that we didn't have to save the images in our personal machines. We used the Google Drive API to handle this task. After an image was uploaded to Google Drive it was deleted from our computers.

With this first part done and all the images available though Google Drive we had to go through one more stage of filtering. As we mention in 2.1, butterflies undergo full metamorphosis in their life cycle, which has four different stages. GBIF doesn't make a distinction between them so there were cases where images of eggs and chrysalis could be found in the downloaded images. For this task there was not an automated way that could be used, so we went through all the images and checked one by one.

In the end of the process described in this Section we had 2863 images of butterflies from Portugal and 7473 more images from Spain, which makes a total of 10336 images of butterflies. With the data set now complete we were ready to train the Machine Learning model to identify different species of butterflies using AutoML Vision.

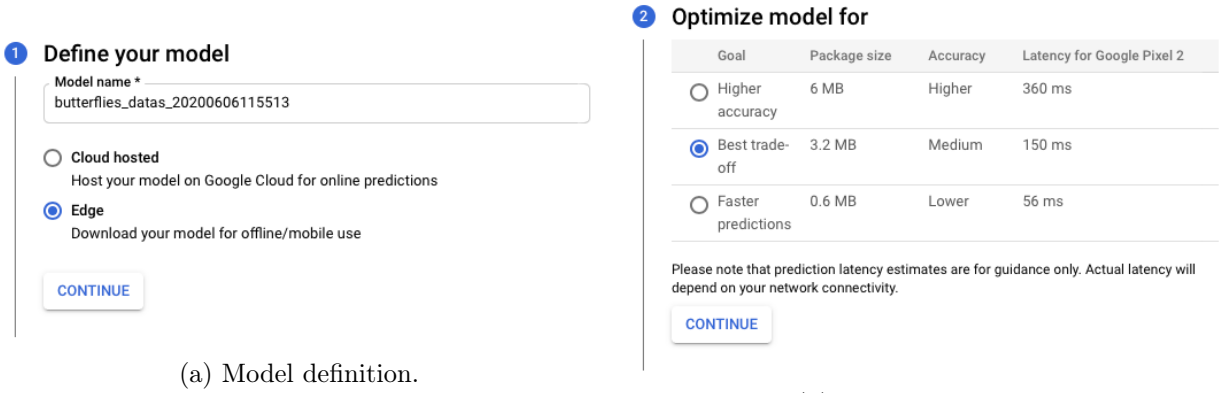
4 The Model

The core of this project was the creation of a Machine Learning model and its deployment in a Web application that would allow us to classify different species of butterflies. In order to achieve this goal we used Google's AutoML Vision tool. The first step when using this tool is to specify the objective of the model. AutoML Vision allows to train Machine Learning models that perform Single-Label Classification, Multi-Label Classification and Object Detection. For the purpose of this project we chose Single-Label Classification because each one of the species of butterflies we worked with were mutually exclusive and each image was assigned to one category only.

Next we had to import the data set of the butterflies into AutoML Vision. Google's Machine Learning tools uses buckets. These are very flexible storage objects identified by a unique, user-assigned key that can be created in Google Cloud Platform. To create a bucket we had to specify the name for the bucket and the country where the servers would be. Here we had to be careful to choose a country that had servers capable of training the Machine Learning model. In the first attempt we created the bucket in a region that was not prepared to train Machine Learning models and, as a result, we had to delete the bucket we had and create a new one. Once the bucket was created we moved the images of the butterflies from Google Drive to the bucket in Google Cloud Platform using Google Colaboratory Notebooks that allows to create quick Python scripts and run the code on Google's servers. Using this tool we performed the mount operation on Google Drive to be able to transfer the entire data set to Google Cloud Platform.

With the images in the bucket we had to make them accessible to the AutoML Vision tool that would make use of them to train the model. For each image in the bucket, AutoML Vision required the location of each image in Google Cloud Platform and its correct label. Once again we wrote a Python script to create a CSV file that had only two columns and no headers. In the first column we had the URI (Uniform Resource Identifier) for each image, which identified the resource in the Google Cloud Platform. In the second column, for each URI we put the correct label for that image. When everything was ready we imported the CSV file to the bucket and placed it at the root.

Finally, with everything set on the part of the bucket, we imported the images to AutoML Vision by selecting the CSV file in the bucket. Once the import process finished we defined the specifications of the model and started training. It would be an Edge model so that we could download it and use it offline and we selected "Higher Accuracy" for the optimisation option. We could also have selected "Faster Predictions" or "Best Trade-Off". We followed the recommended training hours for the data set that were provided by Google and after 10 hours of training the model was ready to be evaluated and tested.



(a) Model definition.

(b) Model optimisation.

Figure 9: Final steps before start training the Convolutional Neural Network.

5 Evaluation

In this section we introduce the results obtained from the models derived throughout the project and the methodology we followed in order to do so. All the results reported here were obtained with a MacBook Air laptop running macOS 10.15.5, with 8GB of RAM and an Intel Core i5 CPU.

5.1 Methodology

To obtain classifications for each of the models derived throughout the project using AutoML Vision, we wrote a Python script that took a .tflite file and ran inference on the model. This script generated a CSV file presenting the first five predictions returned by the model, as well as, the score of each prediction. The output of the model is a floating point value between 0 and 1 and represents the accuracy of a prediction, with, naturally, higher values associated with more confident predictions.

Table 2: Layout of the CSV file where we saved the prediction for each testing image.

expected	prediction 1	score 1	prediction 2	score 2	...
Aglais io	Aglais io	0.72	Pieris Rapae	0.016	...
Gonepteryx rhamni	Gonepteryx cleopatra	0.36	Gonepteryx rhamni	0.24	...
...	...	...	...	...	...
Zerynthia rumina	Zerynthia rumina	0.66	Aricia cramera	0.0079	...

5.1.1 Evaluation Metrics

From the CSV file with the classifications we obtained top-1 and top-5 classification results and built the corresponding confusion matrix. A confusion matrix is a specific table layout the allows for the visualisation of the performance of an algorithm, typically in the context of supervised learning.

The confusion matrix enabled us to assess the number of true positives, true negatives and false positives for each of the species and genus generated by each model.

We briefly describe the meaning of these classifications:

- **true positive**: the model correctly predicts the positive class. In a well built confusion matrix the number of true positives for each class can be found in the main diagonal.
- **false positive**: the model incorrectly predicts the positive class. To obtain the False Positives for a class in the confusion matrix we sum all the values in the column representing that class (except for the element in the diagonal).
- **false negative**: the model incorrectly predicts the negative class. To obtain the False Negatives for a class in the confusion matrix we sum all the values in the row representing that class (except for the element in the diagonal).
- **true negatives**: the model correctly predicts the negative class. In the confusion matrix, the True Negative values for a class are the results presented in every cell that is not the value in the main diagonal, the row or the column representing that class.

From the true positives, false positives and false negatives we where able to obtain precision and recall values for different species and genus. Thus, the analysis of the CSV file allowed us to extract the four metrics used to evaluate the models:

- **top-1 classifications**: fraction of test images for which the correct label was the first one predicted by the model;
- **top-5 classifications**: fraction of test images for which the correct label appeared in the first five predictions made by the model;
- **precision**: the frequency with which a model was correct when predicting the actual class. In other words, the percentage of positive identifications that were actually correct. A models that produces no false positives has a precision of 1.0.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

- **recall**: the percentage of total relevant results correctly classified by the model. In other words, the percentage of correctly identified positive labels. A model that produces no false negatives has a recall of 1.0.

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

For the remainder of this document values reported for precision, recall, top-1 and top-5 classifications are always in percentage.

When evaluating the model for each of these metrics we varied the confidence threshold. We chose four different values: 10%, 25%, 50% and 75%. The confidence threshold affects precision and recall because we only consider predictions with a score above that threshold. When the threshold is higher we produce more false negatives and less false positives. Thus, as the confidence threshold increases, precision improves but recall worsens.

5.1.2 Model Generation

As the project evolved we found it useful to derive different models from the original training data set in order to understand how the results varied in terms of classification capabilities as opposed to having different models specialised in parts of the domain. Despite having the same training data set these models are different in the labels they are able to classify.

We derived the first model to be able to perform automatic taxonomic identification of butterflies. This model was able to identify 100 different species of butterflies. However, there are species of the same genus with very subtle differences between them. Thus, we wanted to understand how the results would change if we aimed only at identifying the genus rather than going at the species level.

Table 3: Information about each of the models derived in the project.

model	training images	number of labels
species model	10194	100
genus model	10194	58
Argynnis model	331	4
Euchloe model	83	2
Hipparchia model	238	4
Melitaea model	232	7
Pieris model	661	3

Also, we used the AutoML Vision to derive different models from the initial one. We derived one model specialised only in genus and it could classify 58 different genus. We also derived five different models that could classify species of certain genera of interest: Argynnis, Euchloe, Hipparchia, Melitaea and Pieris. This allowed us to compare the capabilites of each model and understand how the success of classification varied with the number of labels each model could classify. Table 3 provides a digest of the aforementioned models.

For each model the corresponding Convolutional Neural Network had 173 layers and each image took between 0.13 seconds and 0.16 seconds to be classified.

5.1.3 Test Data Sets

To evaluate the models during the project we created two different test data sets. These data sets have different images and each is different from the ones used to train the model. The first test

data focused only on 50 species (half of the species the model can classify) and is composed of 391 images. The second one is more robust, being able to perform tests on 100 species (all the species the model can identify) and includes 1007 images. Table 4 provides a digest of the test data sets we used.

Table 4: Information about each of the test data sets used to evaluate the models.

image source	species	number of images
specialist websites	50	391
iNaturalist	100	1007

For the first test data set we used websites specialised in Lepidoptera so we could get images classified by specialists. These images are all different from the ones obtained from iNaturalist. On the other hand, the images from the second test data set, taken from iNaturalist, were used by AutoML to provide the first evaluation results.

5.2 Results

We now proceed to report the performance metrics obtained for all models. In 5.2.1 we present the obtained results for the species model, in 5.2.2 we present the results for the genus model and finally, in 5.2.3 we present the results for the five specific genus models.

5.2.1 Species

The two bar plots presented in Figure 10 show the values obtained for precision and recall for the model capable of classifying species. This was the first one derived in the project. We ran inference on this model using the test data sets mentioned in Section 5.1.3.

For both data sets, as the confidence threshold increases, precision increases and recall decreases. This behaviour is expected. A higher confidence threshold means that the model only considers predictions with a score greater or equal than the threshold. As a result, the score for each prediction is higher, which translates in better accuracy. Overall, the model produces less False Positives and thus precision increases.

The dual situation happens for recall. As the confidence threshold increases and the score for considered predictions increases, more images are left out. This results in the model producing more false negatives and the value for recall decreases. The results are similar for both test data sets. However, it is worth pointing out the higher values for precision and recall for the AutoML test data set. This data set includes not only more images but also features more species. As more images are classified and more species are contemplated in the test the results approximate to its true value.

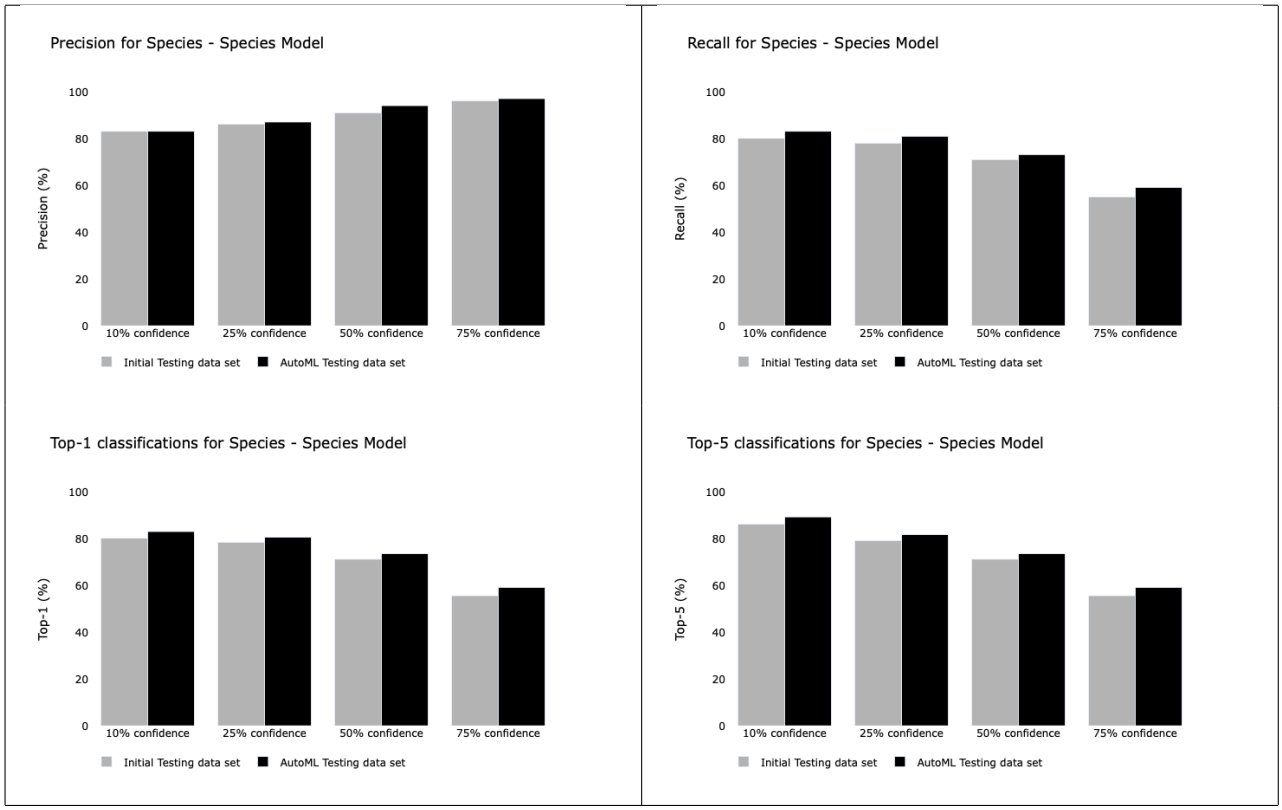


Figure 10: Results for species model using two data sets.

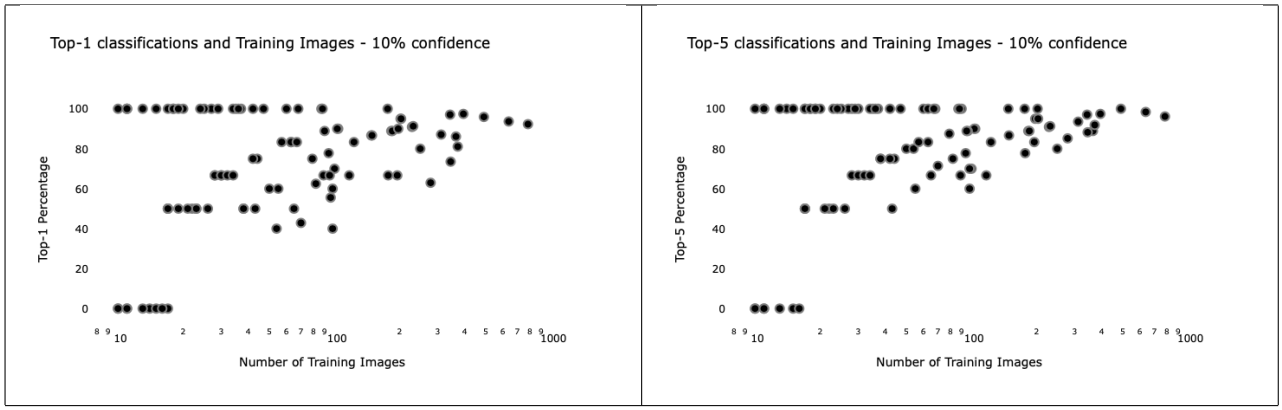


Figure 11: Relation between the number of training images and metrics.

When it comes to top-1 classifications, we can see that, in general, the model presented better top-1 results for species with more training images (Figure 11). For species with less training images we noticed the number of top-5 classifications increasing, as is depicted by the greater concentration

of black dots on the top left side of scatter plot on the right. Also, we noticed that genus that include more than one species tend to present more top-5 classifications. This means that when there are more subtle differences the model struggles to distinguish the differences in these species.

It is also worth pointing out, as it can be observed in Figure 10, that as the confidence threshold increases the number of top-1 classifications and top-5 classifications also decreases. Also, for confidence thresholds of 50% there is no difference between the total number of top-1 classifications and top-5 classifications.

5.2.2 Genus

In another phase of the evaluation we tried to understand how the species model would perform if it only classified for genus. Naturally, we no longer had to consider species of the same genus that the model would naturally struggle to distinguish. These were grouped together in the new model in post-processing before computing the metrics. We also used AutoML Vision to train a specific Convolutional Neural Network with images labeled only for with the genus. By doing this we were able to compare the performance of the species model to predict genus and the performance of one model focused only on genus. As we did previously in the case of species classification, for genus we also considered the two test data sets presented in Section 5.1.3.

By analysing the bar plots in Figure 12 we can see that the species model presents slightly better numbers in terms of precision and recall when compared to when it classifies for species (c.f., Figure 10). Overall, precision and recall values are better as was intuitively expected.

Both models present similar results for precision. However, when we focus on recall we can see a difference in behaviour: the specific genus model shows a very clear improvement over the species model. This is visible across all confidence levels.

Finally, when we consider the scatter plots in Figure 13 we can observe that the species model has a higher top-1 classification with fewer images, when compared to the genus model. This can be seen by the higher density of black dots in the left hand side of the scatter plot in the genus model. An explanation for this could be the fact that by having to distinguish species of the same genus the species model was able to capture and learn more subtle features, where the genus model took a more broad approach and was not able to capture the necessary level of detail.

5.2.3 Genus Specific Models

In the last phase of the project we derived five different models specialised in classifying species within certain genera. Here we obtained mixed results as in some cases the specific model presented better results and in other cases it performed worse. Of the five specific models, the Hipparchia model presented better results, as showed in Table 5. We can see a significant rise in both precision and recall for this genus when compared to the results obtained with the general model for this genus.

Overall, the Argynnis (actually a complex of 3 very close genera - Argynnis, Fabriciana and Speyeria), Euchloe and Melitaea models performed worse when compared to the general model. We could not find a relation between these results and the number of training images or the number

of different labels. For example, the Argynnis model included four different labels, was the second model with more training images and still performed worst. On the other hand, the Hipparchia model

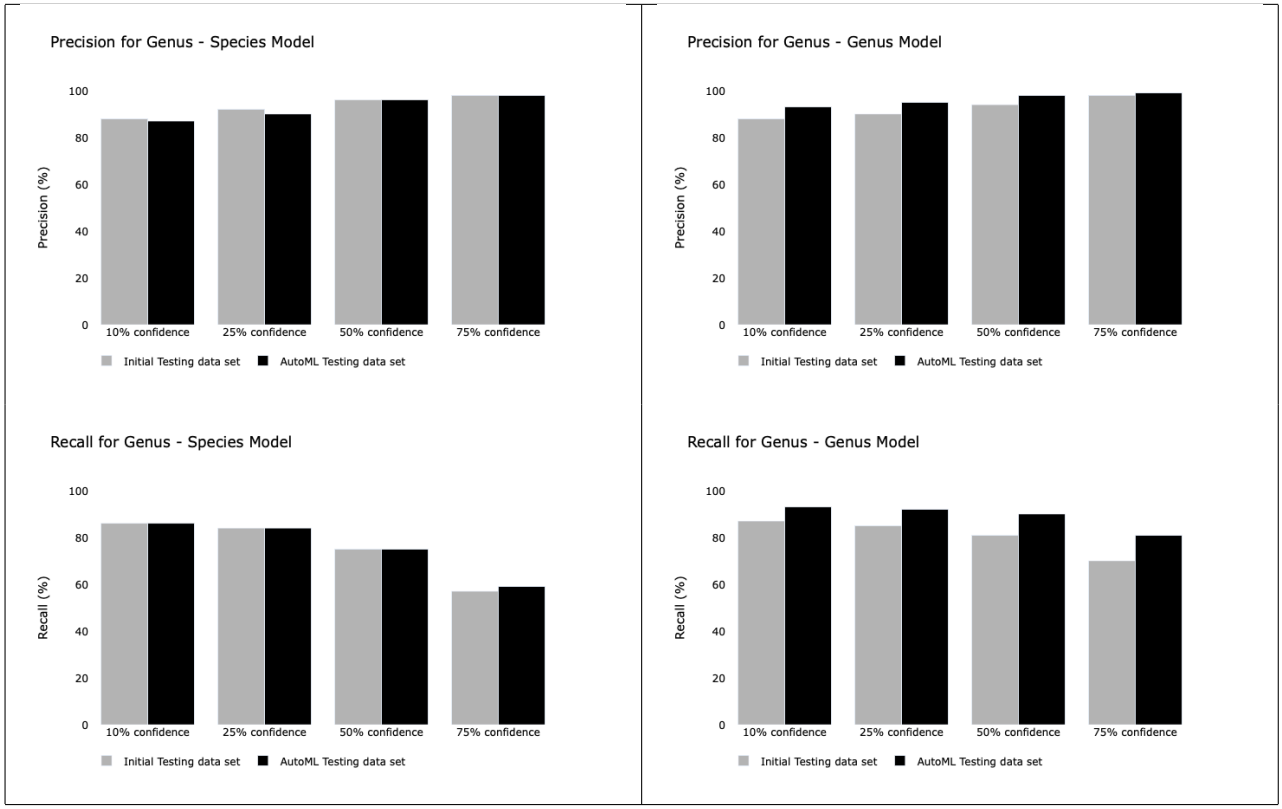


Figure 12: Precision and recall for both genus models using both test data sets.

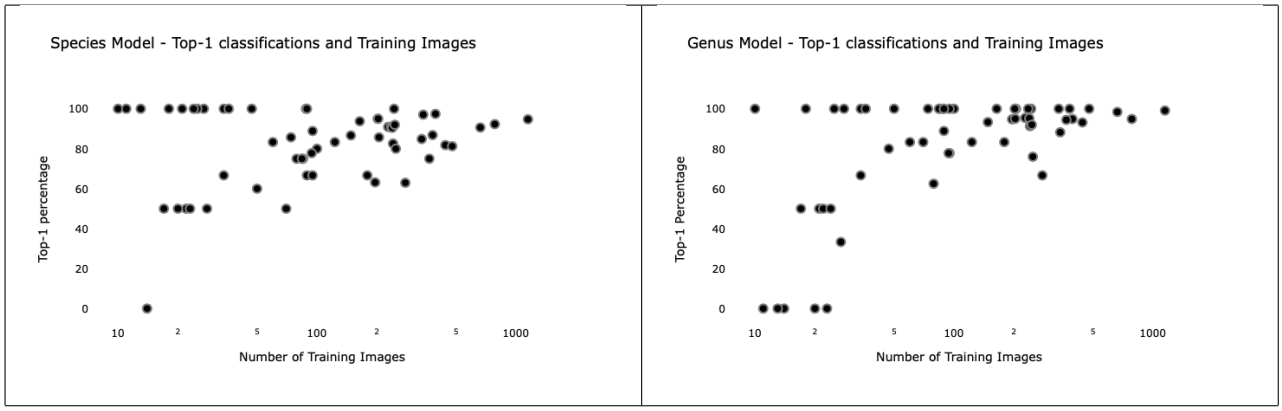


Figure 13: Relation between number of training images and top-1 classifications - 10% confidence.

had less training images and the same number of output labels. It not only performed better when compared to the Argynnis model, but also presented better results than the general model. The Melitaea model performed better than the general model in terms of precision but worse in terms of recall. This was the model with more output labels so, somewhat not surprisingly, it produced more false negatives.

Table 5: Metrics for the genus specific models.

genus	precision general/specific (%)	recall general/specific (%)
Argynnis	83/74	78/74
Euchloe	85/62	75/62
Hipparchia	75/89	65/89
Melitaea	55/61	71/61
Pieris	77/76	73/76

Finally, the Pieris model was the one with more training images and only three labels. We found the values between the Pieris model and the general model to be identical. However, with respect to recall the Pieris model performed slightly better.

Table 6: Metrics for species included in the specific models - 10% confidence.

genus	species	images	top-1 (%)	top-5 (%)	precision (%)	recall (%)
Argynnis	Argynnis pandora	183	100	100	64	100
	Argynnis paphia	100	80	90	89	80
	Fabriciana adippe	22	25	83	100	25
	Speyeria aglaja	28	100	100	100	100
Euchloe	Euchloe belemia	26	40	60	67	40
	Euchloe crameri	57	82	100	60	82
Hipparchia	Hipparchia fidia	103	93	7	88	93
	Hipparchia hermione	13	0	100	0	0
	Hipparchia semele	55	80	80	100	80
	Hipparchia statilinus	70	100	100	88	100
Melitaea	Melitaea cinxia	16	0	0	0	0
	Melitaea deione	41	50	100	100	50
	Melitaea didyma	32	100	100	62	100
	Melitaea nevadensis	36	33	100	100	33
	Melitaea parthenoides	10	0	0	0	0
	Melitaea phoebe	82	75	100	50	75
	Melitaea trivia	15	0	0	0	0
Pieris	Pieris brassicae	195	50	94	69	50
	Pieris napi	371	44	78	80	44
	Pieris rapae	96	92	100	78	92

6 Web application

The deployment of the AutoML model can be regarded as the culmination of all the work done throughout the project. We designed a web page where the user can submit an image of a butterfly and receive back the corresponding species name. We used several frameworks and programming languages. The basis of the Web application is written in Python. The Flask web framework was used to build the server side of the application. It does not have a big learning curve and it easily handles GET and POST requests without making the programmer write a lot of code. On the server side we have another Python file that uses TensorFlow to run inference on the AutoML model. The model returns all the results that have a higher probability compared to the confidence level chosen by the user.

The Flask application interacts with the HTML page that displays the results. We used a form to allow the user to submit the images of the butterflies. Initially the user could only submit one image at a time, but in the final version, with a simple update, the user started to be able to submit several images at a time and get the classification for all of them. We also used a slider to allow the user to choose the confidence level used by TensorFlow when running inference on the model. We also used buttons and images.

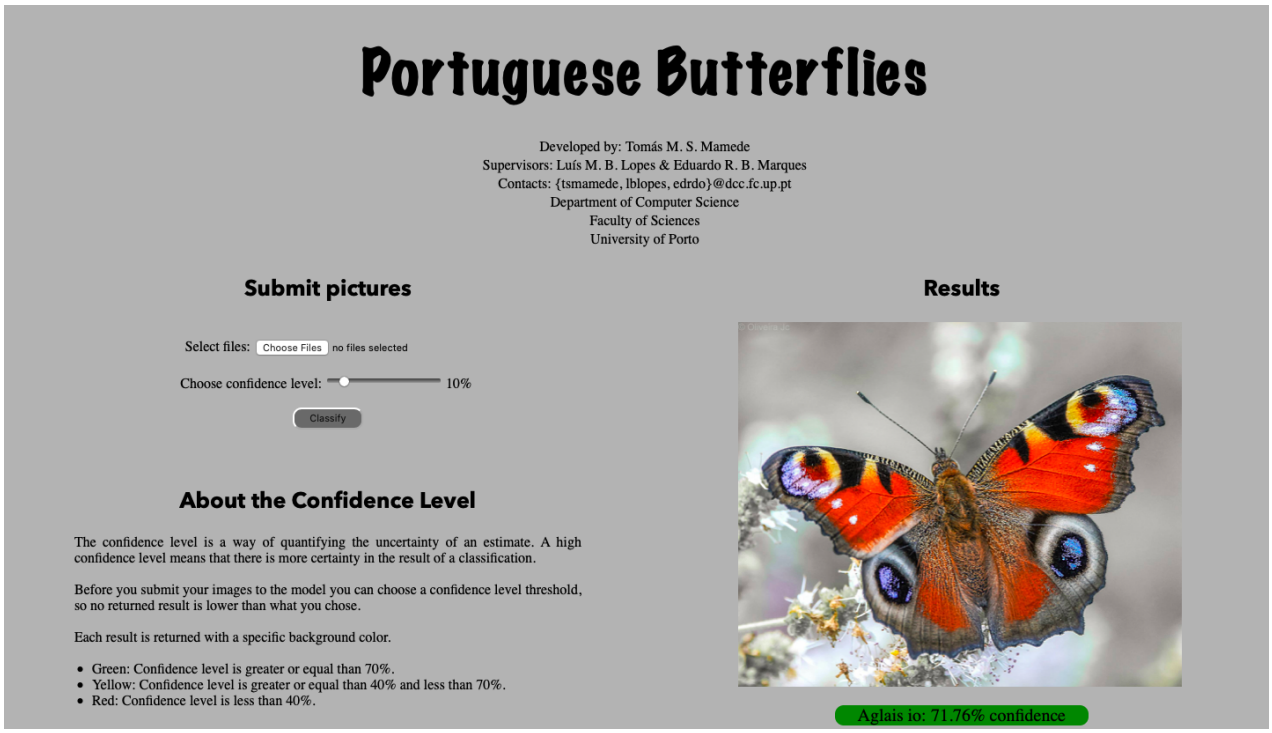


Figure 14: Screenshot of the final Web application.

When the user clicks the "Classify" button the Flask application runs inference on the model for the images submitted by the user. The results are displayed on the HTML page. On the right-hand side of the page we display every image submitted with the possible classifications for that image right below it. The first prediction for each classification is colour coded, as shown in the image below. For a probability above 70% the text background is green and for probabilities between 40% and 70% the background is yellow. Finally, for probabilities below 40% the text background turns red.

To style the Web application we used CSS. We also used JavaScript to create the dynamic slider and make the value change as the user drags across the element in the web page. The web page was divided in two vertical sections. On the left the user is presented with the form to submit the images and choose the confidence level. On the right hand side there is the Results section where the user sees the submitted images with the corresponding result right below it.

7 Conclusion

This report showed the process of deriving a Machine Learning model that uses Deep Learning to classify different species of butterflies that can be found in Portugal, and its deployment in the form a Web application, where users can submit their own images of butterflies to obtain predictions regarding its species.

At the end of this project the model was capable of identifying 100 species of butterflies with good results in terms of precision and recall. However, there is space to improve on the current model. One first step to improve it would be to train the model with more images. This would help the model dealing with species that have lots of similarities between them. On the other hand, we could also improve on the number of species the model can currently identify. Some species are not spotted very often because they exist only at very specific locations in the Portuguese territory.

In the future our intentions are to continue working on this project. From here it can go in different directions such as:

- Make the Web application available publicly for use by the general public. Currently, this is being considered in articulation with the Lusoborboletas website and the Lepidoptera Portugal Facebook page;
- Generalisation of the model to also cover moths and nocturnal species;
- Combining model outputs, e.g., for the genus and species model or even genus specific models, to improve predictive performance.
- The development of a mobile application for iOS and Android;

References

[1] Ethem Alpaydin. *Machine Learning: The New AI*. MIT Press Essential Knowledge series. The MIT Press, 2016.

[2] Henry Walter Bates. *The Naturalist on the River Amazons*. The Narrative Press, 2001.

[3] *Fathers of the Deep Learning revolution receive the 2018 ACM A.M. Turing Award*. URL: https://www.acm.org/media-center/2019/march/turing-award-2018.

[4] Carlos Franquinho and Eduardo Marabuto. *Listagem das Borboletas Diurnas de Portugal Continental e Ilhas*. URL: https://preview.tinyurl.com/y9oyh3es.

[5] Ian Goodfellow, Yoshua Bengin, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. The MIT Press, 2017.

[6] Adrian Hoskins. *Butterflies Of The World*. New Holland Publishing, 2015.

[7] John D. Kelleher. *Deep Learning*. MIT Press Essential Knowledge series. The MIT Press, 2019.

[8] Patrice Leraut. *Butterflies of Europe and Neighbouring Regions*. NAP Editions, 2016.

[9] Ernestino Maravalhas. *Borboletas de Portugal*. Edição do Autor, 2003.

[10] Warren S. McCulloch and Walter Pitts. «A logical calculus of the Ideas Immanent in Nervous Activity». In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259.

[11] Fritz Müller. «Über die Vortheile der Mimicry bei Schmetterlingen». In: *Zoologischer Anzeiger* Volume 1 (1878), pp. 54–55.

[12] Simon J. D. Prince. *Computer Vision: Models, Learning, and Inference*. Cambridge University Press, 2012.

[13] Frank Rosenblatt. «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain». In: *Psychological Review* (1958), pp. 386–408. DOI: 10.1038/323533a0.

[14] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. «Learning Representations by Back-Propagating Errors». In: *Nature* 323.6088 (Oct. 1986), pp. 533–536.

[15] Kim Todd. *Chrysalis: Maria Sibylla Merian and the Secrets of Metamorphosis*. I.B. Tauris, 2007.

[16] Jake VanderPlas. *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, 2016.

[17] Alfred Russel Wallace. *The Malay Archipelago*. Penguin Classics, 2014.

[18] Aston Zhang et al. *Dive into Deep Learning*. Available at https://d2l.ai. 2020.