VRIJE
UNIVERSITEIT
BRUSSEL

ANIMAL CLASSIFICATION AI

Machine Learning

Lennert Bontinck

December 11, 2020

Master Computer Science: AI

Sciences and Bio-Engineering Sciences

Abstract

This intermediate report documents the development of an animal classification AI using a more "old-school" approach of Visual-Bag-of-Words models. The AI should be capable of differentiating 12 different animals. These models, and thus the AI, are developed in Python-based Jupyter Notebooks accompanied with this document. This animal classification AI was developed as a fulfillment of the Machine Learning course requirements and was used to compete in the organised Kaggle competition (Rosseau, 2020).

Part I of this report discusses the accompanied code in general. Section 1.1 explains which files are accompanied and which are the most important. Section 1.2 describes the ideology used to created the code. To make testing multiple models easier, a form of *pipeline* was created and is discussed in section 1.3.

In part II the data analysis part of this project is discussed. Section 2.2 talks about the unbalanced data distribution. In the next section, section 2.3, a deeper look is taken into the data and possible preprocessing is discussed. The last 2 sections of this part, 2.4 and 2.5, discuss how the feature extraction is dealt with and what the numerical representation looks like.

TODO XXX linear baseline

Finally, part IV discusses future plans for this project. Section 4.1 lists possible topics that can be explored to create a better animal classification AI. In the last section, section 4.2, some open issues are discussed.

Contents

Part I

About the code

1.1 Files included with this report

TODO XXX

1.2 Ideology of the developed code

TODO XXX

1.3 A typical model exploration

TODO XXX

1.4 Technical remarks

This report was created in LaTeX by modifying the excellent and well-known VUB themed template from Ruben De Smet (2020). BibLaTeX was used for reference management and natbib was used for more citation control.

Most source files, for this report and the created models, are available on GitHub (Bontinck, 2020). Some files, like the used training images, were not included in this GitHub repository. Details about this can be found on the GitHub page (README file). The supplied code is written in Python-based Jupyter Notebooks. Rights to this GitHub repository can be asked from the author.

Part II

Data analysis

2.1 About this part

Before rigorously testing different models available, it's important to take a look at the data that is supplied. The supplied data consists of 2 main groups of images, labeled training images and unlabeled test images. As per requirement of the Kaggle competition the test images should only be used for evaluating the model on the Kaggle page by submitting a CSV of the prediction results. Thus the test images can't be used for creating the model in any shape or form. This means that only the labeled training images can be used to create and validate the model in development. To avoid altering the model to perform well on the supplied test data and not in general, only the training data will be analysed. All code used for this part is available under the developed code folder on GitHub, in the Jupyter Notebook *data_analysis.ipynb*. This part describes how data can be analysed with the given code. This code can be easily changed to analyse other variations of the data, e.g. using another descriptor.

2.2 Data distribution

The provided labeled training data consists of 12 different classes. There is a total of 4042 labeled training images supplied, the distribution of which is shown in figure 1. As visible in this figure, the distribution between between classes is not balanced. This has to be taken into account when fitting a model, since some models will show unwanted behaviour when fitted with unbalanced data. Luckily many solutions exist to minify the impact of this unbalance.

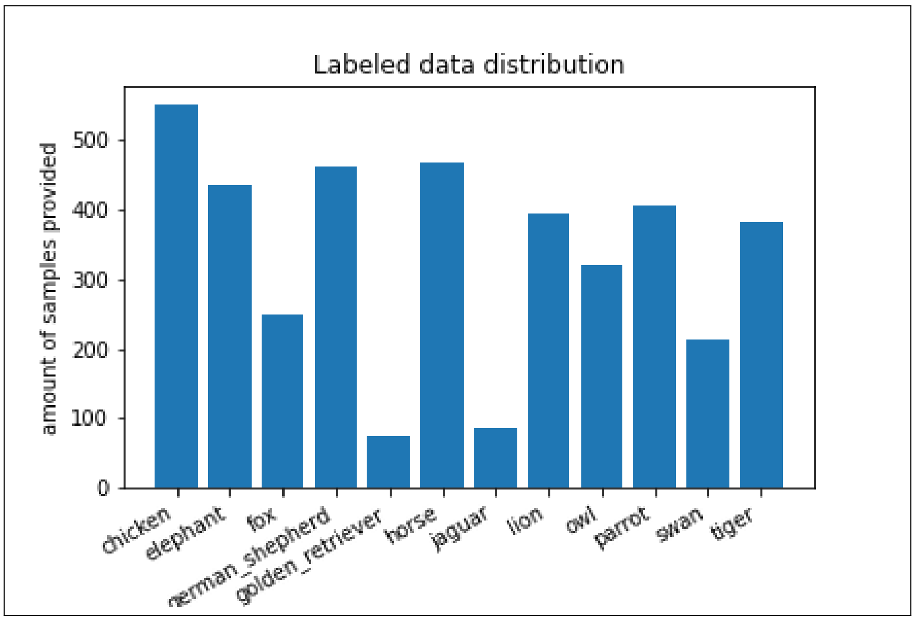

Figure 1: The data distribution of the supplied training set.

2.3 Deeper look at the training data

Whilst noting that the available data isn't balanced over all the classes is very important, there are also different aspects of the data that need analysing. An overview of the supplied training data is given in figure 5, available in the figures list at the end of this report. This figure shows the first five images of each class. From this, it becomes apparent that multiple factors of the data aren't *optimal*. This knowledge is important since it can aid in better prepossessing and in finding a better model in general. The most noteworthy findings are listed here:

- Images vary in shapes, some are taken in portrait, others in landscape.
- Images vary in size, some are high resolution whilst others are relatively low resolution.
- The framing of the subject(s) varies a lot. Sometimes the labeled animal is completely visible and centered in the frame. In some images there are multiple animals spread across the image, others show a close-up of the animal.
- Some images have a detailed background that makes up for a lot of the image, in others the background is blurry and it's impact is presumably less.
- Some images have very vibrant colors in broad daylight, others are black and white in a dimly lit environments.

This diversity in the provided training set is expected since it has been scraped from the web. This also means that *noise* can be expected, another important factor to keep in mind when choosing and optimizing models. Many of the listed things can be minified by doing some clever prepossessing of the images.

2.4 Feature extraction

Since the focus of this competition is on developing great models and not necessarily on data prepossessing and feature extraction, some feature extraction has already been provided. More info on the prepossessing and feature extraction provided is available in the provided notebook *creating_vbow.ipynb*. In short, images are converted from there typical RGB representation to a numerical representation of interesting points, which can be used as input for our model. How this is done will briefly be discussed here.

Instead of using the whole image as data, only a select few of *interesting points* if the image are taken into consideration. These interesting points of an image are found by using the *Shi-Tomasi corner detector*. The following important parameters for the *features.extractShiTomasiCorners* function call are used for the supplied features:

- number of features = 500
- minimum distance between features = 20

Shown in figure 2 is an example output of interesting points found by the Shi-Tomasi corner detector. It's clear that this is far from optimal, but finding interesting points isn't an easy task and thus the results are better then they might seem on first sight.

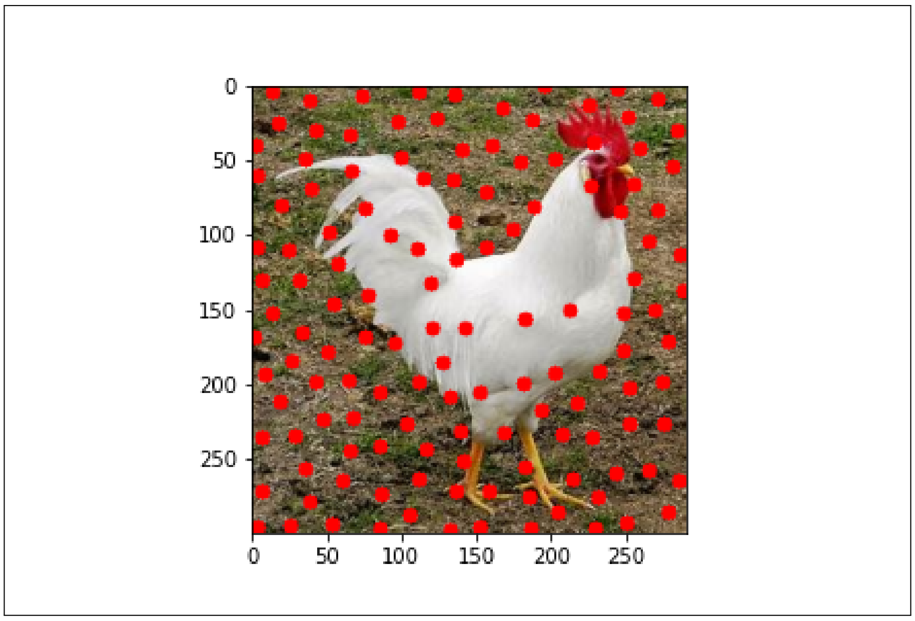

Figure 2: Example of points of interest found by Shi-Tomasi corner detector.

Finding interesting points is only half of the work. These interesting points now need to be represented by numerical values that have actual meaning. Remember from section 2.3 that the provided images differ a lot and thus a descriptor has to be used that minifies the impact of different lighting, scaling... The following descriptor are used and their output are provided: DAISY, ORB, FREAK, LUCID, VGG, BoostDesc, SIFT. Whilst SURF is another great descriptor, it's not provided nor is the license available for this project. SIFT is often referred to as the most famous and successful of these descriptor, but all of them should be explored.

2.5 The numerical representation

As discussed in section 2.4, the images are stored as numerical representations of different features using descriptors. To save time, these numerical representations for all the descriptors are stored in a separate *pickle* file. Since these representation form the input of a model, it's important to get a grip on how these look. The provided *createCodebook* function allows for easily loading in this data. It also allows to specify how many features per image are wanted with the *codebook_size* parameter. As discussed in section 2.4, at most 500 of these are available per image by default. The function returns 2 lists, one containing the labels for the image represented at a certain index, the other containing the requested amount of features per image.

An overview of the data from such features given by the SIFT descriptor is given in figure 5, available in the figures list at the end of this report. From this, it is visible that the values seem to be normalized by the SIFT descriptor. This would have to be checked for all descriptors used and perhaps some outliers would need to be removed.

In figure 3 the correlation matrix is shown for the first 30 features of the SIFT descriptor. The correlation between these values doesn't seem to dramatic, which is mostly positive for our model building.

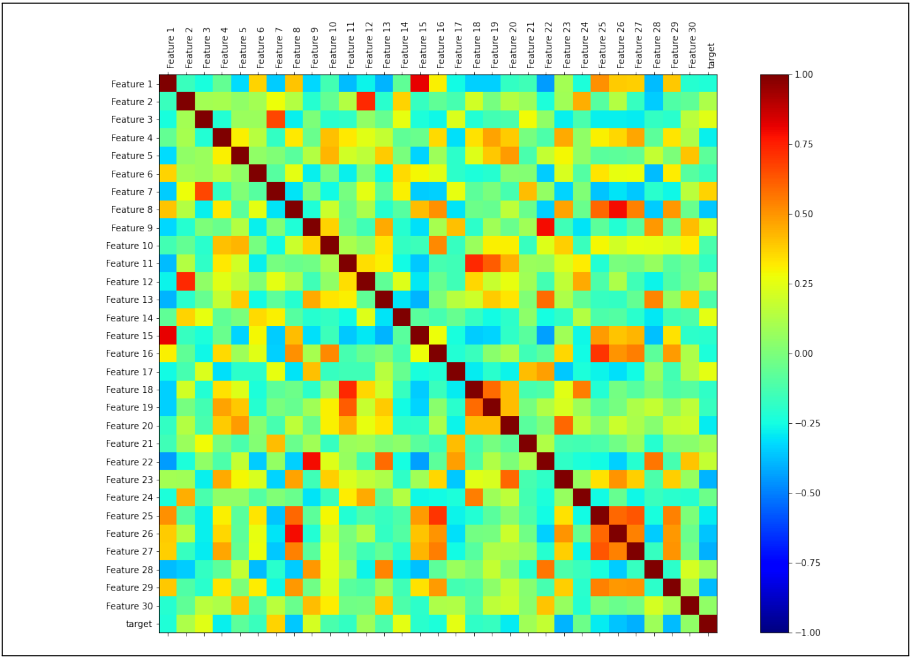

Figure 3: Example of points of interest found by Shi-Tomasi corner detector.

Part III

Linear baseline model

3.1 About this part

TODO XXX

Part IV

What's next

4.1 Further development

Due to limited time available this intermediate report and the current state of the project isn't overwhelming in any stretch of the imagination. Because of this, a special thanks is given to the teacher and teaching assistants who've softened the requirements for the intermediate report.

In the time available until the final deadline many new possibilities for creating a better model will be explored, this might include but is not limited to:

- Preprocessing can be explored.
- The different descriptors can be compared with each other.
- More available models need to be tested and compared to the linear baseline model.
- Nesting multiple good performing models might be an option, using weighted probabilities.
- TODO XXX

Some aspects of the created models and pipeline might also be modified further once the open issues discussed in section 4.2 are resolved.

4.2 Open issues

While developing the first models many questions arose. Most are already answered by googling or discussion with colleagues. Open questions that are not yet answered are listed below, for which guidance from the teaching assistants is asked.

- The data is not available as a pickle file for the SURF descriptor.
- TODO XXX

Figures

The most notable figures are included in this list. They're high resolution thus zooming in the PDF should be viable to get a clearer view.

Data distribution of training set

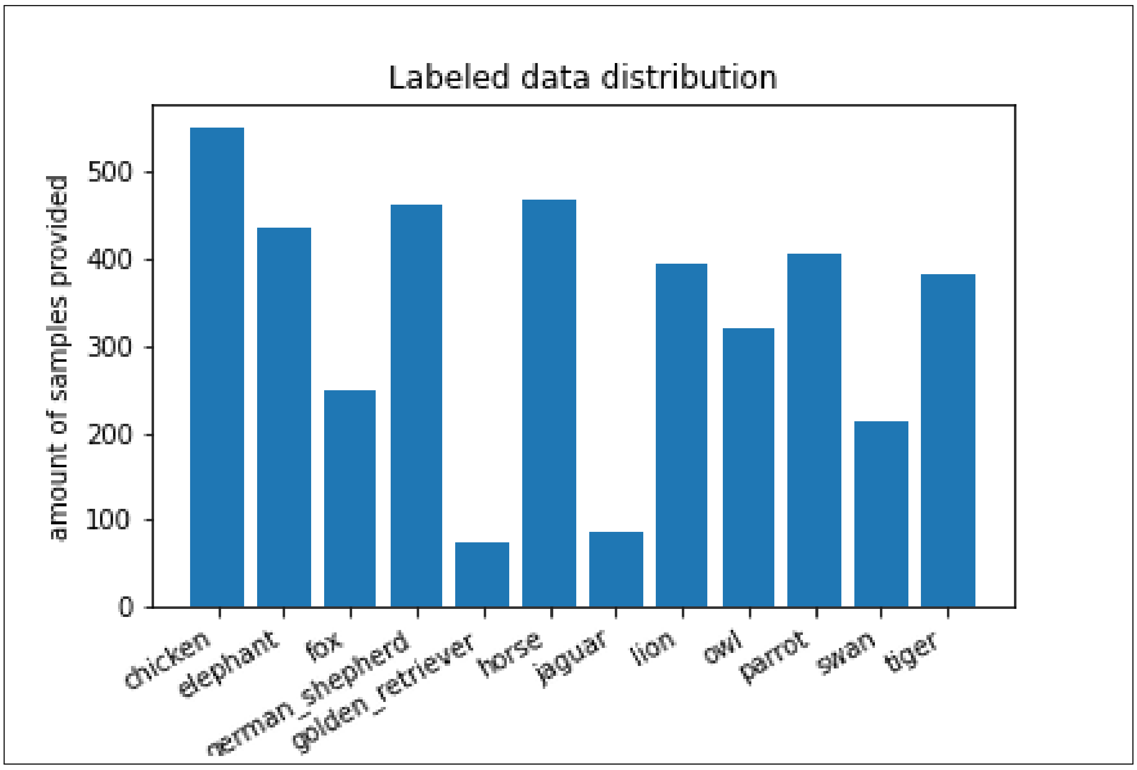

Figure 4: The data distribution of the supplied training set.

Overview of training set

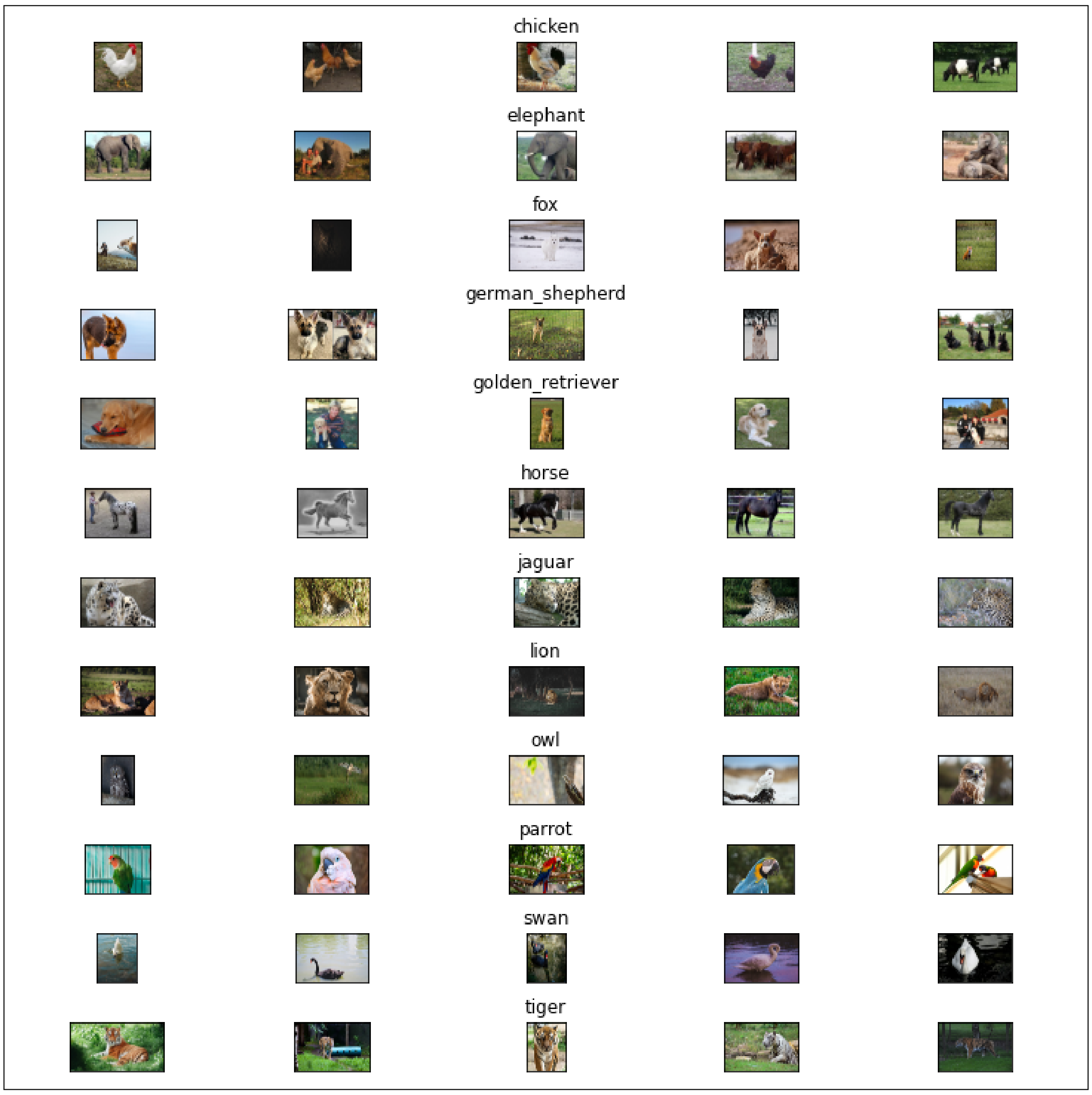

Figure 5: An overview of the supplied data per class.

Shi-Tomasi corner detector example

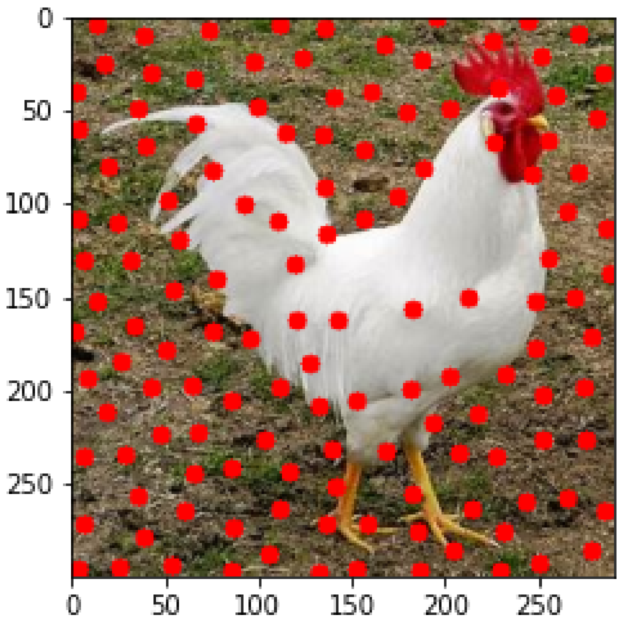

Figure 6: Example of points of interest found by Shi-Tomasi corner detector.

Overview of features data

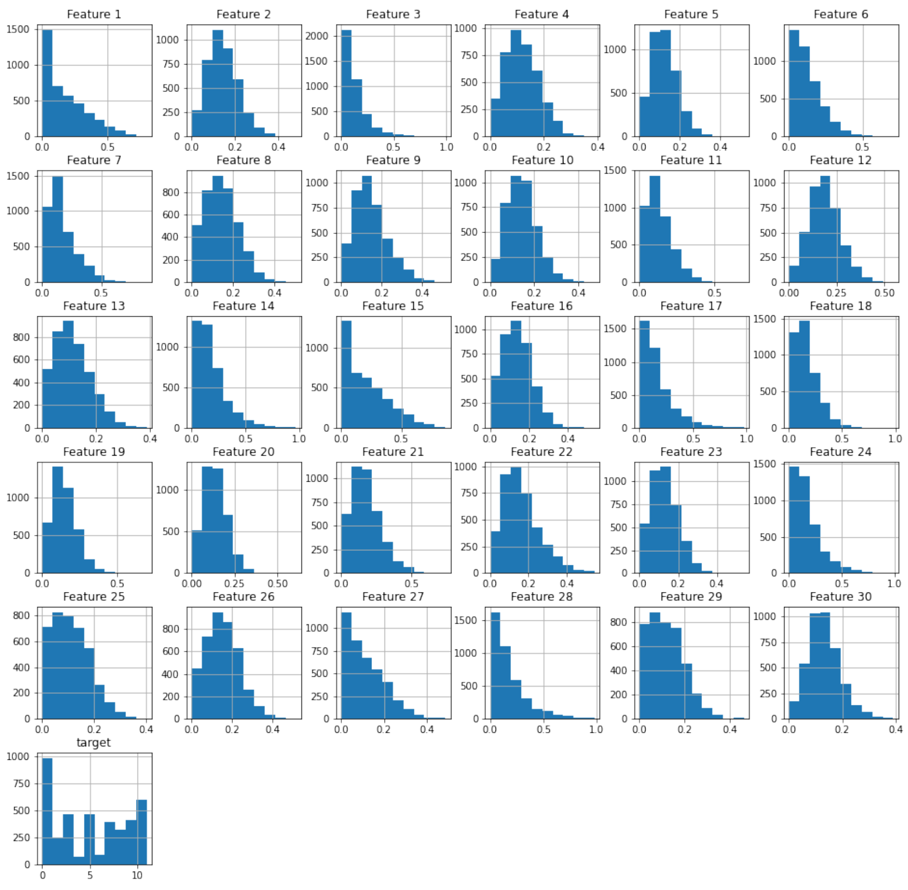

Figure 7: An overview of the first 30 features data from a SIFT descriptor.

References

Bontinck, L. (2020). *Machine learning project* [GitHub commit: Todo]. Retrieved December 11, 2020, from https://github.com/VUB-CGT/ml-project-2020-pikawika

De Smet, R. (2020). *Vub latex huisstijl* [GitHub commit: d91f55799abd390a7dac92492f894b9b5fea2f47]. Retrieved November 2, 2020, from https://gitlab.com/rubdos/texlive-vub

Rosseau, A. (2020). *Vub: Animal classification*. Retrieved November 15, 2020, from https://www.kaggle.com/c/vub-animal-classification-20/