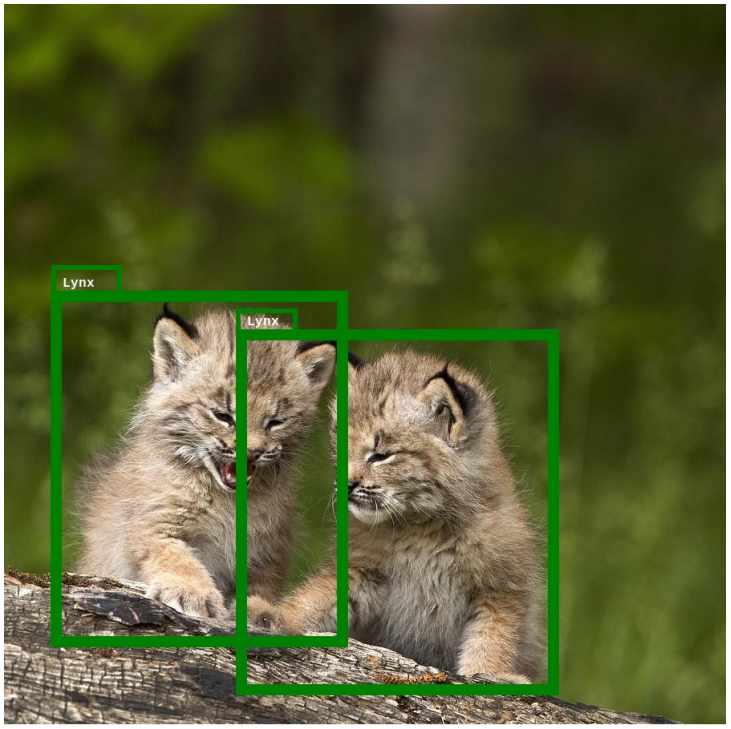


MASTER - DATA SCIENCE

LOUIS BEAUMONT - 2019

CLOUD COMPUTER VISION FOR WILDLIFE MONITORING

---

ynov CAMPUS

When the first humans reached Australia about 45,000 years ago, they quickly drove to extinction 90% of its large animals. This was the first significant impact that Homo sapiens had on the planet's ecosystem. It was not the last.

— Yuval Noah Harari

Acknowledgements

The past six months have been an exciting journey, dedicated to the last step in finishing my master Data Science at Ynov Aix-en-Provence: writing this thesis. I would like to express my sincere gratitude to those who helped me during this period.

First of all, I would like to thank Pascal Alanche for giving me the opportunity to work at Arpège SAS a Rohde & Schwarz company during my 3 years of apprenticeship. It was a great experience to be part of the R&D team and participate in the many interesting projects of the company. I have met a lot of remarkable people who taught me valuable knowledge about geostationary satellites and high-frequency radio communication. I would like to thank my supervisor from Ynov, Renaud Angles, for his guidance throughout the research project. His feedback, whether it was enthusiastic or critical, helped me to structure and improve this thesis and to continue the project at tough points.

Likewise, I would like to thank Olivier Gimenez, Oksana Grente, Anna Chaine, Lucile Marescot from the CNRS and ONCFS for their invaluable information about the context of camera-trap and wildlife monitoring, as well as the introduction to ecology research.

My gratitude also goes to the people of the open-source world from the people sharing publicly datasets, to the people on the algorithm side, developing amazing framework and making fantastic tutorials, thanks Github, Microsoft, Google, especially in my case at least.

Finally, I would like to express my gratitude to my parents and family for their support and wisdom throughout my thesis. Especially my brother Romain who guided me through the difficulties encountered on the technical parts.

I hope that you will enjoy reading the current research. If you have any questions or comments, please do not hesitate to contact me.

Résumé

Dans ce travail, je présente une solution hébergée dans le cloud facile à déployer, à maintenir, à mettre à l'échelle et à tarifer pour la surveillance de la faune à l'aide de l'intelligence artificielle, qui permet d'effectuer une vaste collecte et analyse de données et de déployer et suivre rapidement de nouvelles données de surveillance pour tout groupe particulier d'espèces et dans tout habitat. En particulier, je montre comment ce logiciel peut évoluer de quelques images initiales à des dizaines de milliers de vidéos par mois.

La surveillance de l'état des espèces de notre planète à l'aide d'une grande variété de technologies et de plateformes est cruciale pour améliorer la conservation de la biodiversité et pour développer de nouvelles méthodes de gestion des espèces. Je montre comment mettre en œuvre une application de surveillance de la faune directement dans le cloud à l'aide de l'intelligence artificielle (IA). Cette solution offre également des moyens d'analyser les données pour la surveillance des espèces et la capacité d'agréger les données afin d'identifier les modèles et les tendances pour informer les écologistes.

Plus important encore, pour l'identification des animaux, le système permet d'économiser un temps considérable de travail manuel tout en offrant le même niveau de précision que les bénévoles humains.

Abstract

In this work I present an easy to deploy, maintain, scale and price cloud solution for wildlife monitoring powered by artificial intelligence, which makes it possible to conduct an extensive data collection and analysis and to quickly deploy and track new monitoring data for any particular group of species and in any habitat. In particular, I show how this software can scale from an initial few images to up to tens of thousands of videos per month.

I believe that monitoring of the status of our planet's animals using a wide variety of technology and platforms is crucial to improve biodiversity conservation and to develop novel methods of managing species. I demonstrate how to implement a cloud application for cloud based monitoring of wildlife using artificial intelligence (AI). This solution also offer ways to analyze the data for the monitoring of species and the ability to aggregate data to identify patterns and trends to inform ecologists.

Most importantly, for animal identification, our system can save a tremendous amount of time of the manual labor while performing at the same accuracy level than human volunteers.

TABLE OF CONTENTS

List of Figures

Introduction

The present work is intended, as far as possible, to give an insight into the process of automating cognitive task in the cloud to those readers who, from a general scientific and philosophical point of view, are interested in the field of artificial intelligence and ecology.

As the planet changes due to urbanization and climate change, biodiversity in the world is declining. We are currently witnessing an estimate of the increasing rate of species loss 200 times more than historical rates[1].

The Earth's biodiversity is rapidly changing in what may be considered as one of the largest environmental challenges of our time.

For example, there is no doubt that in 2040 approximately one-third of all living species will be in the range of mammals living in the world today. How much of these new species will become endangered is unclear, and it might be impossible to assess exactly because there are still so many unknowns about the changes in our planet at large. There is no good data to guide us to any prediction for how our species will recover from the loss of species and how many we will need to lose to sustain human population growth of current levels.

At least for now, ecologists and volunteers are trying to collect such data but they spend a huge amount of time manually identifying species in images, in this era of digitization, something can be done.

Indeed, recent pattern recognition techniques allow computers to understand the semantics of images, allowing the automation of this cognitive task.

This subject is important, first, it has high practical relevance. The amount of time gained by automatizing manual labours is tremendous. Second, this study can be beneficial to the theory building in the emerging research field of deep learning and especially in the area of computer vision.

Furthermore, this work can be valuable for those interested in the field of big data and cloud computing, which grant an easy access to deploying, scaling and maintaining automated tasks.

This work includes several dimensions:

- Ecology: the balance of living being survival
- Artificial intelligence
- Big data & cloud computing

We propose an architecture that is easy to deploy, use, maintain and price entirely in the cloud, allowing wildlife monitoring through concrete data visualisations.
Through this thesis, the answer to this question will be given:
How to automatize and scale wildlife monitoring using cloud computer vision ?
In this thesis, we first introduce the collaborators of this projects such as the ONCFS and CNRS.
Secondly, a short explanation of computer vision is stated.
We then mention the related work on computer vision and wildlife monitoring, especially the datasets available and existing algorithms.
Finally, the implementation done and the future possible work will be presented.
The answer to the problematic will then be discussed, followed by a conclusion.

4.1 ONCFS

the French National Office for Hunting and Wildlife aims to safeguard and sustainably manage wildlife and its habitats. A public institution under the dual supervision of the Ministries of Ecology and Agriculture, the National Hunting and Wildlife Office fulfils five main missions responding to the main lines of the last Environmental Conference, following the Grenelle de l'Environnement:
territorial surveillance and the environment and hunting police, studies and research on wildlife and its habitats, technical support and advice to administrations, local authorities, managers and spatial planners, the evolution of hunting practices in accordance with the principles of sustainable development and the development of environmentally friendly rural land management practices, the organization of the examination and the issuance of the hunting permit. The ONCFS in a few figures Created in 1972, the Office has a budget of 120 million euros to carry out its missions throughout the country (metropolitan France and the French overseas departments).
1,700 people working for biodiversity:

- 1,000 Environmental Technical Agents, commissioned by the Ministry in charge of sustainable development, divided into Departmental Services and Mobile Intervention Brigades
- 350 Environmental Technicians, also commissioned, assigned to Departmental Services (supervision), Inter-Regional Delegations and the various departments
- 70 engineers and technicians, grouped in five C.N.E.R.A. specialized in a group of species: migratory birds, deer and wild boars, mountain fauna, small sedentary plain fauna, predators and predatory animals.
- 80 technical managers
- 156 administrative staff
- 30 workers involved in the management of the domains and reserves managed or co-managed by the Office.
- 25 hunting licence inspectors
- 6 departments, in support of the Director General, implement the institution's action in their areas of competence

- 10 Inter-Regional Delegations - 90 Departmental Services
- 1 Board of Directors
- 1 Scientific Council
- 27 wildlife reserves, totalling nearly 60,000 hectares of protected areas that allow the ONCFS to carry out studies and experiments.

ONCFS brought useful information about camera-trap context, the animals monitored and provided a set of videos and images of camera-trap taken lynxes and wolves.
ONCFS could gain a lot of time from the solution presented in this thesis, without losing any accuracy compared to human-level animal detection.

4.2 CNRS

The Centre national de la recherche scientifique, better known by the acronym CNRS, is the largest French public scientific research organisation. Legally, it is a public scientific and technological institution (EPST) under the administrative supervision of the Ministry of Higher Education, Research and Innovation.

Founded by the Decree-Law of 19 October 19391, in order to ”coordinate the activity of laboratories in order to obtain a higher return from scientific research”, the CNRS was reorganized after the Second World War and then moved clearly towards fundamental research.

The CNRS operates in all fields of knowledge through a thousand accredited research and service units, most of which are managed with other structures (universities, other EPSTs, grandes écoles, industries, etc.) for five years in the administrative form of ”mixed research units”.

The CNRS was created on 19 October 1939, following the merger between an agency of resources, the Caisse nationale de la recherche scientifique and a major institution of laboratories and researchers, the Centre national de la recherche scientifique appliquée.

This merger was prepared by Jean Zay with the help of the Under-Secretaries of State for Research Irene Joliot-Curie and Jean Perrin. The decree organizing the CNRS is signed by the current President of the Republic, Albert Lebrun, the President of the Council, Édouard Daladier, the Minister of National Education, Yvon Delbos, succeeding Jean Zay, and the Minister of Finance, Paul Reynaud. The creation of the CNRS was intended to ”coordinate the activity of laboratories in order to obtain a higher return from scientific research” and, in the words of Jean-François Picard, to ”merge it into a single body, in a way the logical outcome of scientific and centralizing Jacobinism”.

Fusion was encouraged by the Second World War: the French authorities, not wishing to reproduce the mistakes made during the First World War (all the scientists had been mobilized, often as executives in the infantry or artillery, which led to the disappearance of a high proportion of young scientists), assigned researchers to the CNRS. This merger therefore did not attract any press coverage. At the beginning, part of the research was conducted for the needs of the French army. Threatened by the Vichy Regime, which finally maintained it and confirmed geologist Charles Jacob6 at its head, the CNRS was reorganized at the Liberation. Frédéric Joliot-Curie was appointed director and provided him with new research grants. De Gaulle’s arrival in power in 1958 opened a period described as the ”golden age of scientific research”

and the CNRS: the CNRS' budget doubled between 1959 and 19628.

In 1966, associated units were created, ancestors of the UMRs. These are university laboratories, supported by the CNRS, thanks to its human and financial resources. In 1967, the National Institute of Astronomy and Geophysics was founded, which in 1985 became the National Institute of Universe Sciences (INSU). The National Institute for Nuclear and Particle Physics (IN2P3) was established in 1971.

In the 1970s, there was a change from science to society: the CNRS wondered about its ambition and its modes of action. The first interdisciplinary programmes are launched and global contracts with industry are signed (the first with Rhône-Poulenc in 1975).

In 1982, the law of 15 July, known as the Chevènement de programmation des moyens de la recherche publique law, decreed that research personnel, technical and administrative engineers and administrative staff are to be transferred to the civil service: they become civil servants, with, for researchers, a status similar to that of lecturers and university professors.

According to a survey conducted in 2009 by Sofres for Sciences Po, the CNRS enjoyed a 90% level of trust among the French, well before the police (71%), the Government (31%), the President of the Republic (35%) or political parties (23%), and second only after the family (97%).

The CNRS of Montpellier has been of great help, both in terms of additional information for cameras, but also in terms of research in the field of ecology and the kind of data that could potentially interest them...

Computer vision

With the rise of the Internet and modern technologies, we can now confirm with certainty that we live in an image-based society, the video sharing platform Youtube is responsible for 37% of all mobile internet traffic alone (see Figure 5.1).

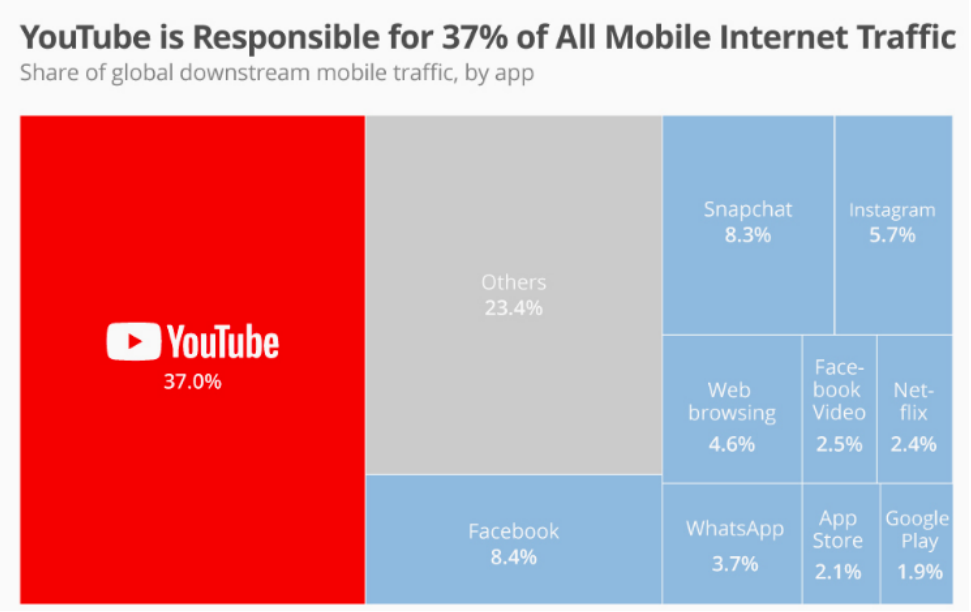


Figure 5.1: Youtube internet traffic share. Source

With about 3.5 billion smartphone users[2], a huge part using massively visual content sharing platforms, the amount of information transmitted by this data is enormous, so automating the abstract information extraction of these pixels has an unimaginable potential. Computer Vision is a technology that allows computers to extract abstract information from visual content.

5.1 What is it

The term ”computer vision” refers to the different techniques that allow computers to see and understand the content of images. This is a sub-category of artificial intelligence and machine learning.
The computer vision field brings together multiple techniques from various fields of engineering or computer science. In general, the different methods aim to reproduce the human vision. To understand the content of images, machines must be able to extract a description: an object, a description, a 3D model... Some computer vision systems may also require image processing, i.e. simplifying or augmenting the content of the image. Examples include normalizing the photo-metric properties of the image, cropping its contours or removing noise such as digital artifacts induced by low light.

5.2 Challenges

What could be simpler than vision? From birth, humans are naturally able to see. However, allowing a computer to do the same is not an easy task. To date, computer vision still fails to match human vision. One of the reasons is that we still do not really know how human vision works. We need to understand how perceptual organs such as the eyes work, but also how the brain interprets this perception. Although research in this area is progressing at a rapid pace, we are still a long way from unlocking all the mysteries of vision.

Another challenge is related to the complexity of the visual world. An object can be perceived from multiple angles, under various lighting conditions, partial hidden by other objects...

A true computer vision system must be able to perceive the content in any of these situations and extract information from it. In fact, computer vision is a real scientific challenge.

5.3 Deep learning

Several years ago, most computer vision algorithms consisted of simple heuristics to extract features from pixels (See an example of heuristic on Figure 5.2).

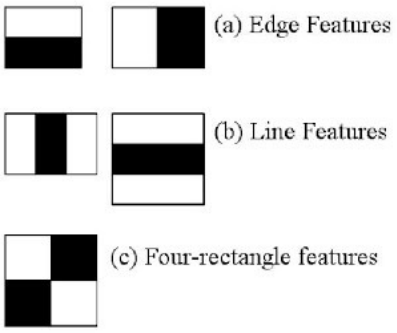


Figure 5.2: Haar features. Source

Over time, there has been more and more use of machine learning and such automatic pattern recognition algorithm, the true revolutionising technology that have made machine learning in general works so well is neural networks and especially deep neural networks (see Figure 5.3).

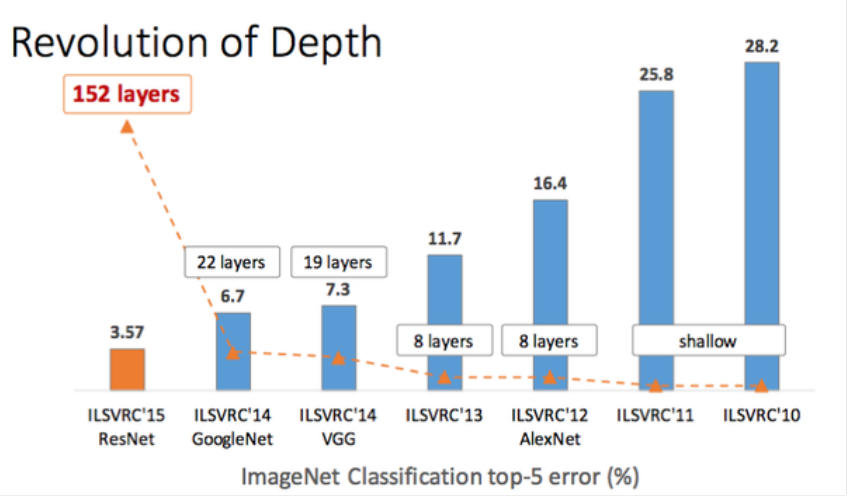


Figure 5.3: Revolution of depth, from a few network layers to hundred(s). Source

In addition to deep neural networks, we saw a birth of a game breaking algorithm in computer vision, it has been called ”convolutional” neural network (CNN)[3]. The difference with CNN and dense network is that CNN take advantage of the hierarchical pattern in images (see Figure 5.4).

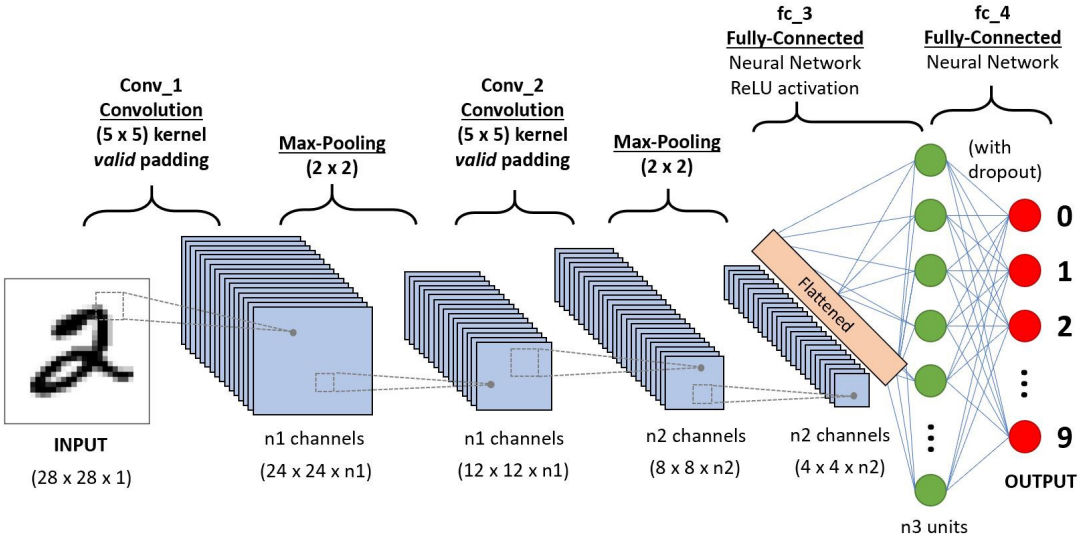


Figure 5.4: Convolutional neural network. Source

5.4 Applications

Despite the difficulties associated with the development of computer vision, the considerable advances made over the years have already enabled Computer Vision to perform many tasks.

This technology is proving to be effective for optical character recognition, also known as ”OCR”, everyone with an Android smartphone can use it through Google Lens (see Figure 5.5).

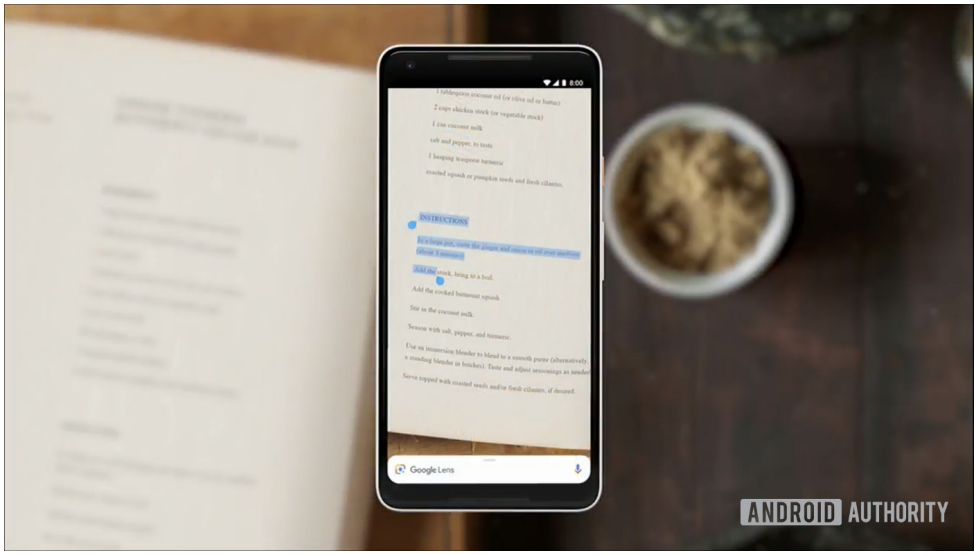


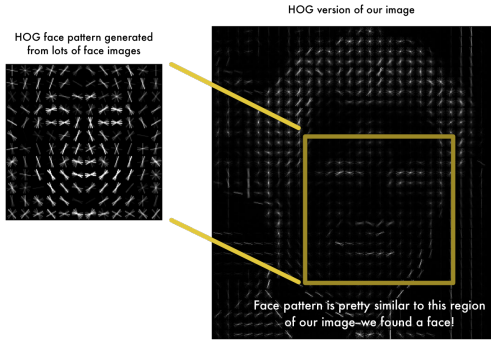
Figure 5.5: Optical character recognition directly on your smartphone with Google Lens. Source

In the field of automotive safety, computer vision is used for hazard detection. With the emergence of autonomous cars, computer vision will soon occupy a central place in the automotive industry as it will allow vehicles to see on the road (see an example of image segmentation in the field of smart vehicles on Figure 5.6).

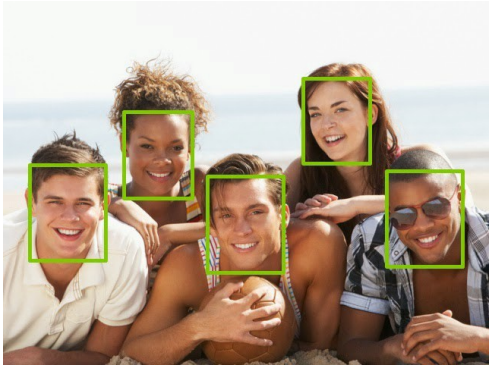
Figure 5.6: Image segmentation on the road. Source

It is also used in the film industry for ”match move”, i.e. to synchronize computer-generated images with the real actors. It is also used for motion capture.

The facial recognition technologies of the latest smartphones, such as the famous Face ID of Apple's latest iPhone, are also based on Computer Vision. The same technology is used by automatic surveillance cameras (see face-related applications on Figure 5.7).



(a) Histogram of oriented gradients (used in facial recognition) Source



(b) Face detection. Source

Figure 5.7: Examples of face-related applications

As you will have understood, computer vision is used in a wide range of fields. As this technology develops, the opportunities it offers will increase in the coming years. Soon, all machines will probably be able to see, in exactly the same way as human beings and better ...

Computer vision is not yet very practical to use on our current everyday interfaces: smartphones. In the future, we can imagine increasing human vision by coupling it with computer vision via mixed reality glasses, for example looking at a monument, having its history displayed on the glasses...

Related work

6.1 Object classification

An example of object classification is whether, for a given image, if there is a cat or not in the image.
The most popular example of object classification is MNIST[4], which was a task to find the number in the image (between 0 and 9).
Object classification is at the core of computer vision and it is probably the easiest task to solve. This task could be used as an alternative to object detection though, because there isn't usually multiple animals anyway on the same picture and object classification is also faster since it doesn't have to infer location.

6.1.1 Datasets

6.1.1.1 ImageNet

ImageNet[5] is a database of annotated images produced by the organization of the same name for computer vision research. In 2016, more than ten million URLs were manually annotated to indicate which objects are represented in the image; more than one million images benefit in addition to bounding boxes around the objects. ImageNet is based on WordNet (see Figure 6.1) which is a semantic lexicon for the English language that is used extensively by computational linguists and cognitive scientists. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonym rings (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

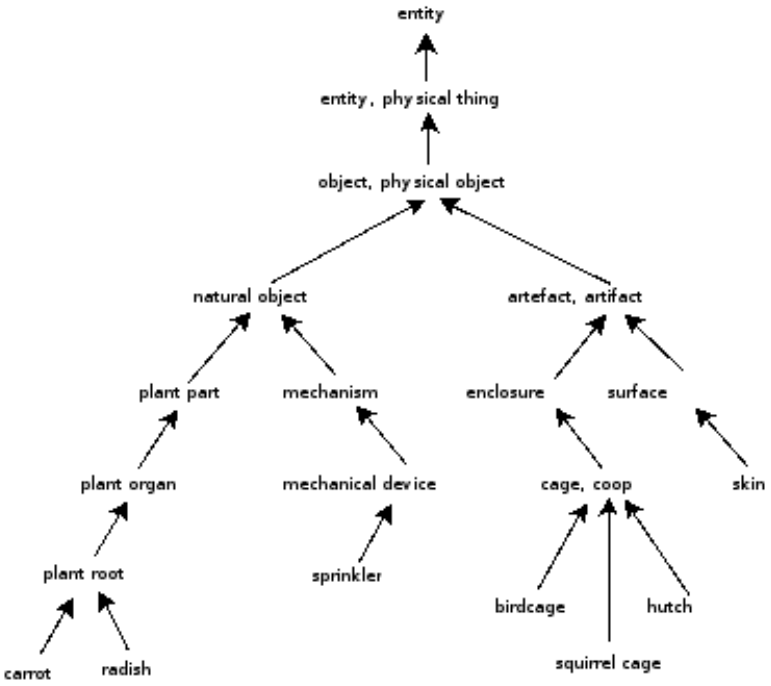


Figure 6.1: WordNet directed acyclic graph. Source

The database of annotations on URLs of third-party images is freely available, but ImageNet does not own the images themselves. Since 2010, the ImageNet project has been organizing an annual competition: ImageNet Large Scale Visual Recognition Challenge (ILSVRC), or "ImageNet Large Scale Visual Recognition Competition". It consists of a software competition whose goal is to accurately detect and classify objects and scenes in natural images.

Imagenet is therefore a general classification dataset that aims to develop the vision in general, it is widely cited and widely used in many models

6.1.2 Models

6.1.2.1 FixRes

The top performance algorithm to date is called FixRes[6], it propose a simple but effective strategy to optimize classifier performance when train and test resolutions differ. It is only a fine adjustment of the network to the resolution of the test, inexpensive by calculation. This allows powerful classifiers to be trained using small training images. For example, FixRes achieve a top-1 accuracy of 77.1% on ImageNet with a ResNet-50 trained on 128x128 images and 79.8% with an image trained on 224x224 images. Conversely, a pre-trained 32x48d ResNeXt-101 with low supervision on 940 million public images at 224x224 resolution and further optimizing the 320x320 test resolution, obtain a maximum accuracy of 86.4% (top-5: 98.0)%).
FixRes is both a state of the art classification model on ImageNet[7], but also on iNaturalist[8]

6.2 Object detection

Object detection is locating instances of objects in images, for example, while you are looking at this text, you see multiple words next to each other, it is object detection.
Object detection models are more appropriate for identifying multiple relevant objects in a single image. The second important advantage of object detection models over image classification models is that the location of objects is provided.

6.2.1 Datasets

6.2.1.1 COCO

COCO[9] dataset is a collection of images of scenes of daily life containing common objects in their natural context (see Figure 6.2). Objects are labeled using instance-based segmentations to precise location of objects. The data set contains photos of 91 types of objects that are *"easily recognizable by a 4-year-old child"*. With a 2.5 million instances tagged in 328,000 images.
COCO is widely used for object detection and image segmentation due to its recognition simplicity.

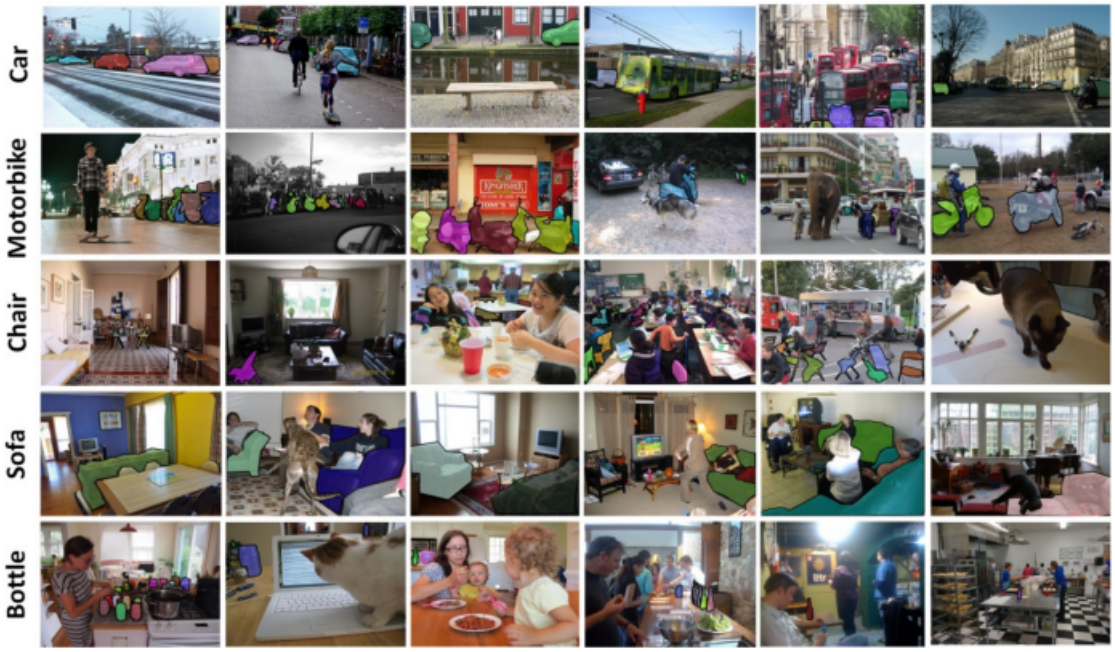


Figure 6.2: Examples of segmented objects from the 2015 COCO dataset. Source

6.2.1.2 Open Image

Open Image V4[10] is a 9.2 million pixel image data set with unified annotations for image classification, object detection and visual relationship detection. The images have a Creative Commons Attribution license that allows you to share and adapt the material. They were collected from Flickr without a predefined list of class names or tags, thus generating natural class statistics and avoiding initial design bias. Open Images V4 offers a large scale in several dimensions: 30.1 million pixel image level labels for 19.8 KB concepts, 15.4 million pixel selection boxes for 600 object classes and 375,000 pixel visual relationship annotations for 57 classes. For object detection in particular, 15 times more enclosures is provided than the next largest datasets (15.4 million enclosures out of 1.9 million images). Images often show complex scenes with several objects (8 objects annotated per image on average). Visual relationships are annotated together, which support the detection of visual relationships, an emerging task that requires structured reasoning. Complete and detailed statistics on the dataset are provided, the quality of annotations are validated and the performance of many modern models evolution with increasing amounts of training data are delivered.

Open Image bring high quality images and is more complex for recognition than COCO, it also allow vision relationship detection.

6.2.2 Models

6.2.2.1 Performance metrics

6.2.2.1.1 Intersection on Union (IoU)

IoU is a measure based on Jaccard Index that evaluates the overlap between two bounding boxes. It requires a ground truth bounding box B_{gt} and a predicted bounding box B_p. By applying the IoU we can tell if a detection is valid (True Positive) or not (False Positive). IoU is given by the overlapping area between the predicted bounding box and the ground truth bounding box divided by the area of union between them:

$$\text{IoU} = \frac{\text{area}\,(B_p \cap B_{gt})}{\text{area}\,(B_p \cup B_{gt})}$$

IoU is a value between 0 and 1, which corresponds to the overlap zone between the predicted zone and the truth zone on the ground. The higher the IoU value, the better the intended location of the box for a given object. Usually, we keep all candidates in the bounding box with an IoU above a threshold.

6.2.2.1.2 mean Average Precision (mAP)

A common metric which is used for the Pascal VOC object recognition challenge is to measure the Average Precision (AP) for each class. The following description of Average Precision is taken from Everingham et. al.[1]

The mean Average Precision (mAP) is computed by taking the average over the APs of all classes. For a given task and class, the precision/recall curve is computed from a method's ranked output. Recall is defined as the proportion of all positive examples ranked above a given rank. Precision is the proportion of all examples above that rank which are from the positive class. The AP summarizes the shape of the precision/recall curve, and is defined as the mean precision at a set of eleven equally spaced recall levels [0,0.1, . . . ,1]:

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots,1\}} p_{interp}(r)$$

The precision at each recall level r is interpolated by taking the maximum precision measured for a method for which the corresponding recall exceeds r:

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$$

where p(˜r) is the measured precision at recall ˜r. The intention in interpolating the precision/recall curve in this way is to reduce the impact of the "wiggles" in the precision/recall curve, caused by small variations in the ranking of examples. It should be noted that to obtain a high score, a method must have precision at all levels of recall – this penalizes methods which retrieve only a subset of examples with high precision (e.g. side views of cars).

[1] Source

6.2.2.2 Convolutional network based on the region (R-CNN)

The region-based convolutional network is one of the main contemporary approach to object detection, In R-CNN[11], a selective search algorithm was used. Selective search is one of the generic object proposal generation methods.

1. Try to detect objects
2. Run a CNN on these objects
3. Run the output into a SVM to classify the object and a linear regressor to adjust the box

This approach can be expensive however because many crops are necessary, leading to significant duplicated computation from overlapping crops (see Figure 6.3).

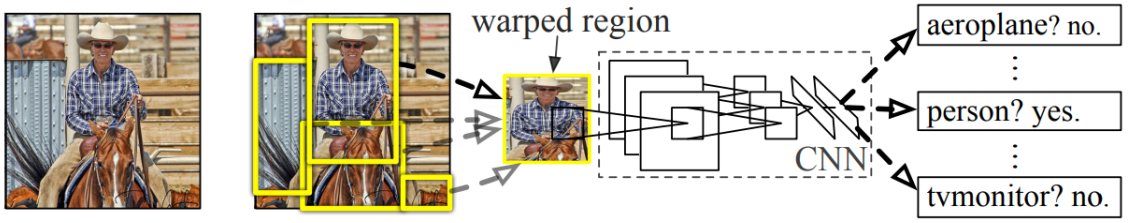


Figure 6.3: Region-based Convolution Network. (R-CNN). Source

6.2.2.3 Convolutional network based on a fast region (Fast R-CNN)

The convolutional network based on the fast region (Fast R-CNN)[12] aims to reduce the time consumption related to the large number of models required to analyze all region proposals (see Figure 6.4).

1. Performing feature extraction before proposing regions, therefore only running one CNN over the entire image instead of a CNN CNN per region

2. Replacing the SVM with a softmax layer, therefore extending the neural network for predictions instead of creating a new model

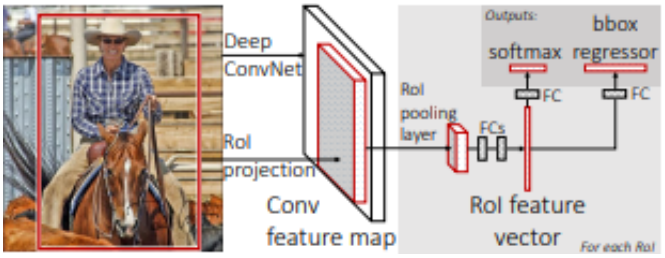


Figure 6.4: Region-based Convolution Network. (R-CNN). Source

6.2.2.4 Convolutional network based on faster regions (Faster R-CNN)

The region proposals detected with the selective search method were still needed in the previous model, which is expensive in terms of calculation.
Faster R-CNN[12] introduced the Regional Proposal Network (RPN) (see Figure 6.5) to directly generate regional proposals, predict selection frameworks and detect objects. The convolutional network based on a faster region (Faster R-CNN) is a combination of the RPN model and the Fast R-CNN model.

1. At the last layer of an initial CNN, a 3x3 sliding window moves across the feature map and maps it to a lower dimension (256-d)
2. For each sliding-window location, it generates multiple possible regions based on k fixed-ratio anchor boxes
3. Each region proposal consists of an "objectness" score for that region and 4 coordinates representing the bounding box of the region

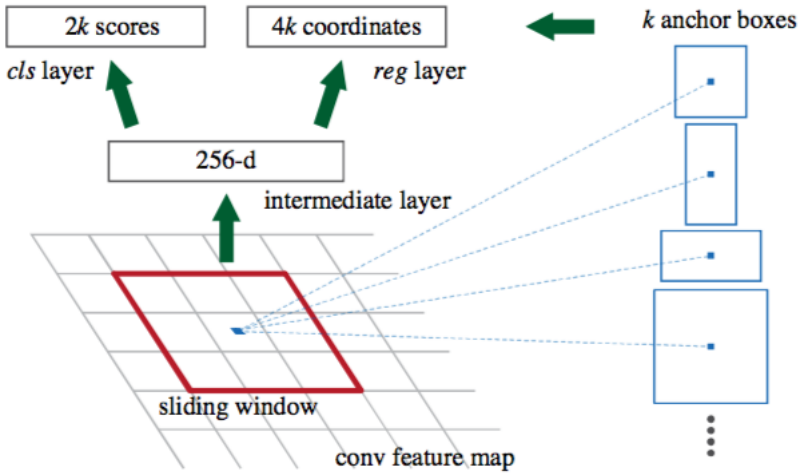


Figure 6.5: Region Proposal Network. Source

The faster R-CNN uses RPN to avoid the selective search method, it speeds up training and testing processes and improves performance.

6.2.2.5 Fully convolutional region-based network (R-FCN)

The fast and rapid methodologies of R-CNN consist in detecting region proposals and recognizing an object in each region. The fully convolutional region-based network (R-FCN) is a model with only convolutional layers allowing to share all the computation.
R-FCN merge the two basic steps into a single model to simultaneously take into account object detection (location invariant) and its position (location variant).

1. Run a CNN over the input image
2. Add a fully convolutional layer to generate a score bank of the aforementioned "position-sensitive score maps." There should be $k^2(C+1)$ score maps, with k^2 representing the number of relative positions to divide an object (e.g. 3^2 for a 3 by 3 grid) and C+1 representing the number of classes plus the background.
3. Run a fully convolutional region proposal network (RPN) to generate regions of interest (RoI's)
4. For each RoI, divide it into sub-regions as the score maps
5. For each sub-regions, compare with the score bank to see if it matches the corresponding position of some object.
6. Once each of the sub-regions has an object match value for each class, average the sub-regions to get a single score per class.
7. Classify the RoI with a softmax over the remaining C+1 dimensional vector

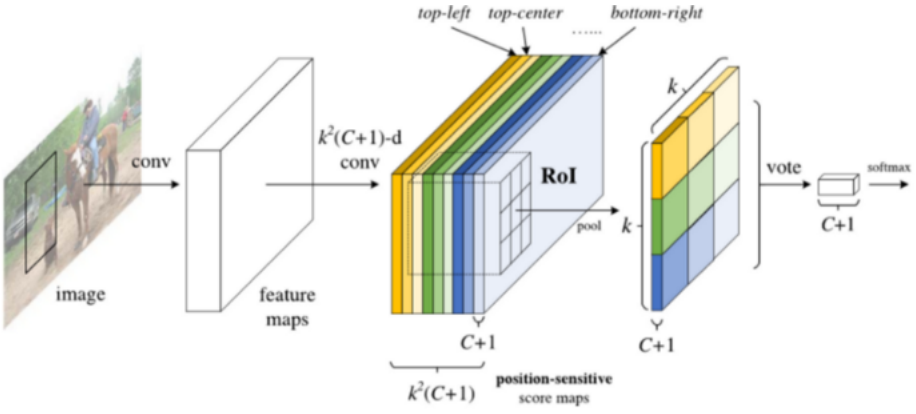


Figure 6.6: Fully convolutional region-based network. Source

In simple terms, the R-FCN examines each region proposal, dividing it into sub-regions. For each sub-region, he asks: "Does it look like the left end of a baby?", "Does it look like the upper center of a baby?" "Does it look like a baby's top right?" etc. For all possible classes. If enough sub-regions say "yes, I match up with a part of a baby", the RoI is classified as a baby after a softmax over all classes (see Figure 6.7).

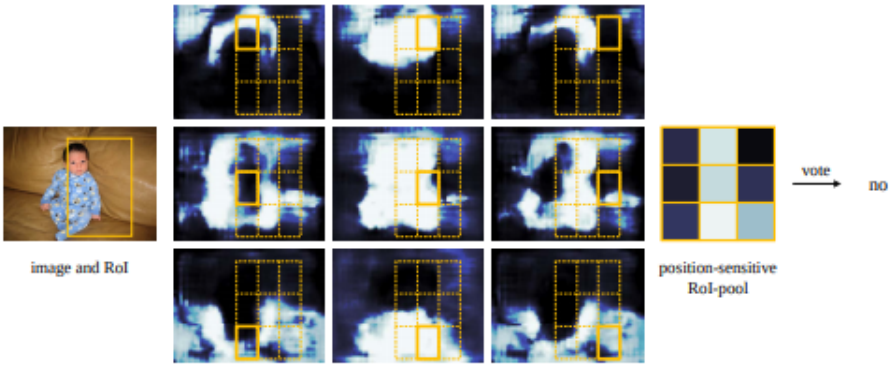


Figure 6.7: Visualization of R-FCN (k × k = 3 × 3) for the person category. Source

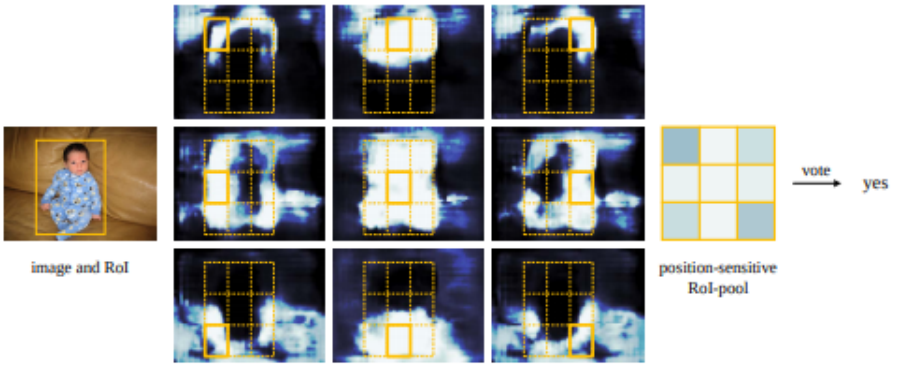


Figure 6.8: Visualization when an RoI does not correctly overlap the object. Source

6.2.2.6 You only look once (YOLO)

The YOLO[13] model directly predicts bounding boxes and class probabilities with a single network in a single assessment. The simplicity of the YOLO model allows real-time predictions. YOLO is doing an end-to-end learning paradigm: proposals, features, and the classifier becoming one neural network (see Figure 6.9).

One indispensable component is non-maximum suppression (NMS), a post-processing algorithm responsible for merging all detections that belong to the same object.

1. Split our image into cells. Each cell will be responsible for predicting bounding boxes score, this score is simply the probability to detect the object multiplied by the IoU between the predicted and the ground truth boxes.

2. Remove boxes with low object probability and bounding boxes with the highest shared area in the process called non-max suppression.

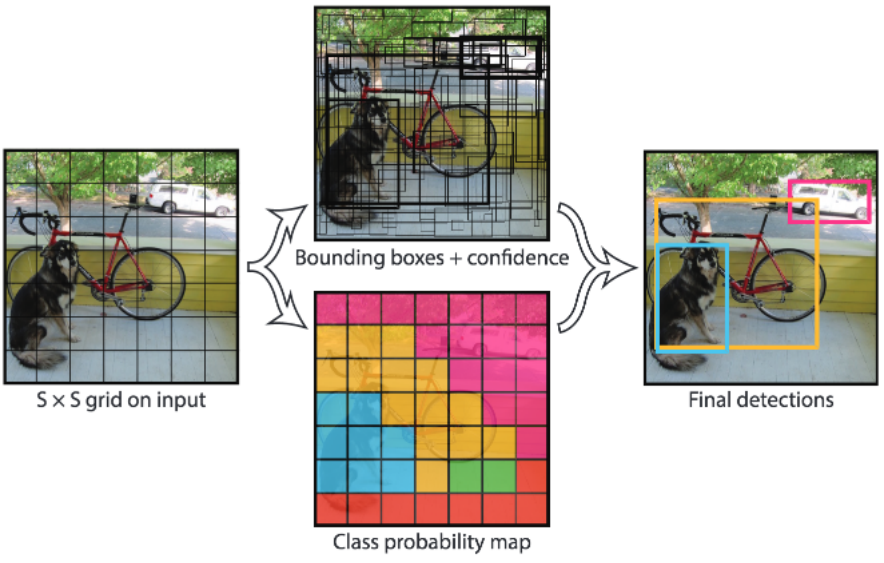


Figure 6.9: Imaged example of the steps of YOLO. Source

6.2.2.7 Single-shot detector (SSD)

Similar to the YOLO model, Single Shot Detector (SSD)[14] predict both bounding boxes and class probabilities with an end-to-end CNN architecture (see Figure 6.10).
The SSD model uses extra feature layers from different feature maps of the network in order to increase the number of relevant bounding boxes.

1. The model takes as input an image that passes through several convolution layers with different filter sizes (10x10, 5x5 and 3x3) that will generate feature maps.
2. Each location in these feature maps are processed by specific convolution layers with 3x3 filters, to produce a set of bounding boxes similar to the Fast R-CNN anchor boxes.
3. For each box, simultaneously predict the bounding box location and the class probabilities.
4. During training, match the ground truth box with these predicted boxes based on IoU. The best predicted box will be labeled a “positive,” along with all other boxes that have an IoU with the truth > 0.5.

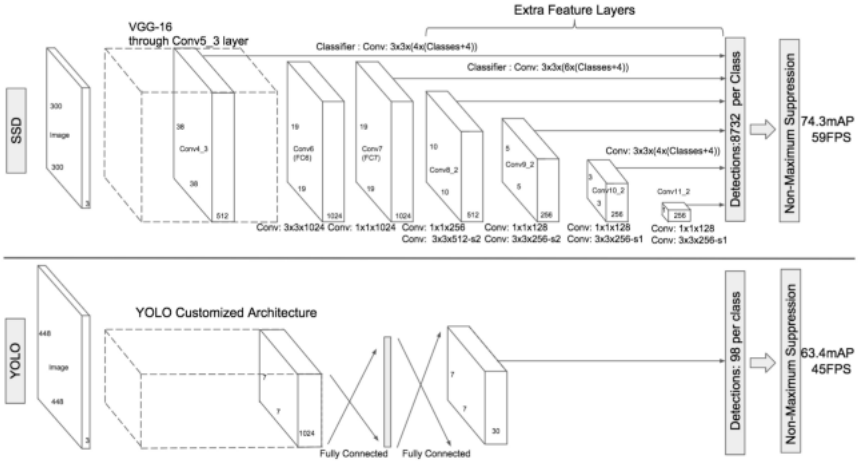


Figure 6.10: Comparison between the SSD and the YOLO architectures. Source

The varying-size feature maps brought by SSD help capture objects of different sizes. Here is an example of SSD inference (see Figure 6.11).

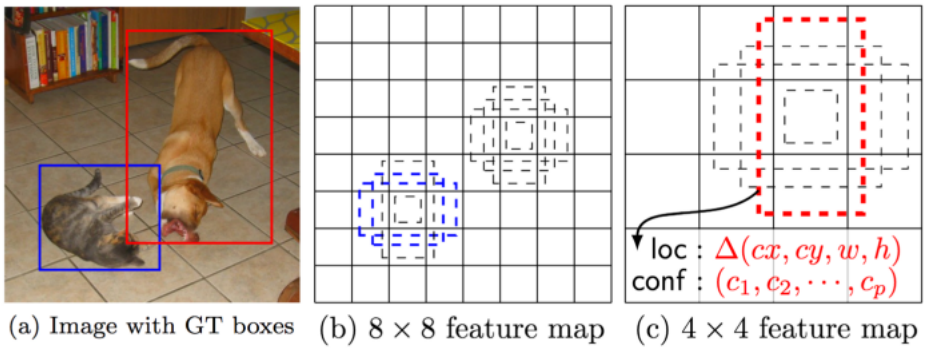


Figure 6.11: SSD Framework. Source

In the end SSD does two things:

- First, apply non-maximum suppression to group boxes that overlap strongly. In other words, if four boxes of similar shapes, sizes, etc. contain the same cat, NMS will keep the one with the most confidence and reject the rest.
- Second, the model uses a technique called hard negative mining to balance classes during training. In the case of hard negative mining, only a subset of the negative examples with the highest training loss (i.e. false positives) are used for each iteration of the training. SSD maintains a 3:1 ratio between negatives and positives.

6.2.2.8 Neural Architecture Research Network (NASNet)

NASNet[15] consists in learning the architecture of a model using reinforcement learning to optimize the number of layers while improving the accuracy of a given data set. NASNet achieved superior performance with a lighter model than previous work compared to the 2012 ImageNet classification challenge (see Figure 6.12). The NASNet network has an architecture learned from the CIFAR-10[16] dataset and is trained in the ImageNet 2012 dataset.

From the official paper:

"Our approach is inspired by the recently proposed Neural Architecture Search (NAS) framework, which uses a reinforcement learning search method to optimize architecture configurations. Applying NAS, or any other search methods, directly to a large dataset, such as the ImageNet dataset, is however computationally expensive. We therefore propose to search for a good architecture on a proxy dataset, for example the smaller CIFAR-10 dataset, and then transfer the learned architecture to ImageNet."

Figure 6.12: Example of object detection results differences between Faster-RCNN with Inception-ResNet-v2 featurization (top) and NASNet-A featurization (bottom). Source

6.2.2.9 Convolutional network based on a mask region (Mask R-CNN)

Mask R-CNN[17] is an extension over Faster R-CNN. Faster R-CNN predicts bounding boxes and Mask R-CNN essentially adds one more branch for predicting an object mask in parallel (see example on Figure 6.13).

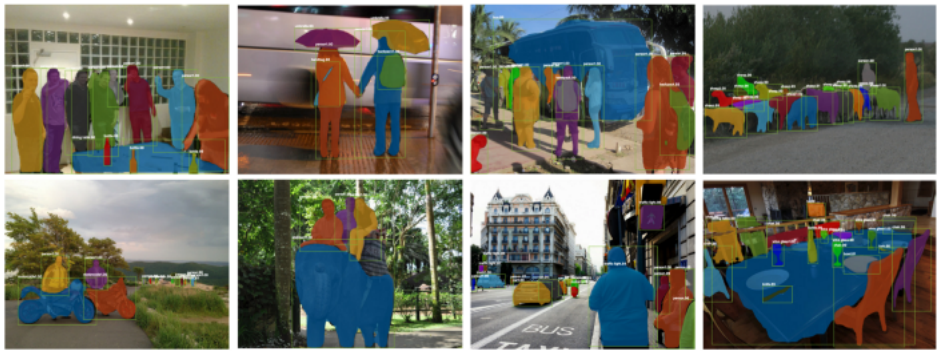


Figure 6.13: Examples of Mask R-CNN application on the COCO test dataset. Source

There are two stages of Mask R-CNN (see Figure 6.14).

1. Generates proposals about the regions where there might be an object based on the input image.
2. Predicts the class of the object, refines the bounding box and generates a mask in pixel level of the object based on the first stage proposal. Both stages are connected to the backbone structure.

"The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner."[17]

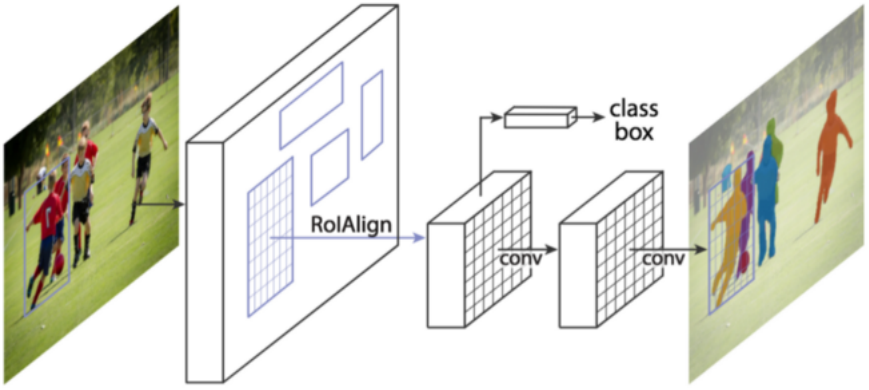


Figure 6.14: Mask R-CNN framework for instance segmentation. Source

6.2.3 Object detection conclusion

Over the years, object detection models tend to infer that location and classification have a fully differentiable network. Therefore, it can be trained from head to tail with neural networks in a end-to-end manner.

The most important question is not which detector is the best. It may not be possible to answer. The real question is which detector and which configurations offer us the best balance between speed and accuracy that your application needed. Below is a comparison of the accuracy vs speed tradeoff (time measured in milliseconds) (see Figure 6.15).

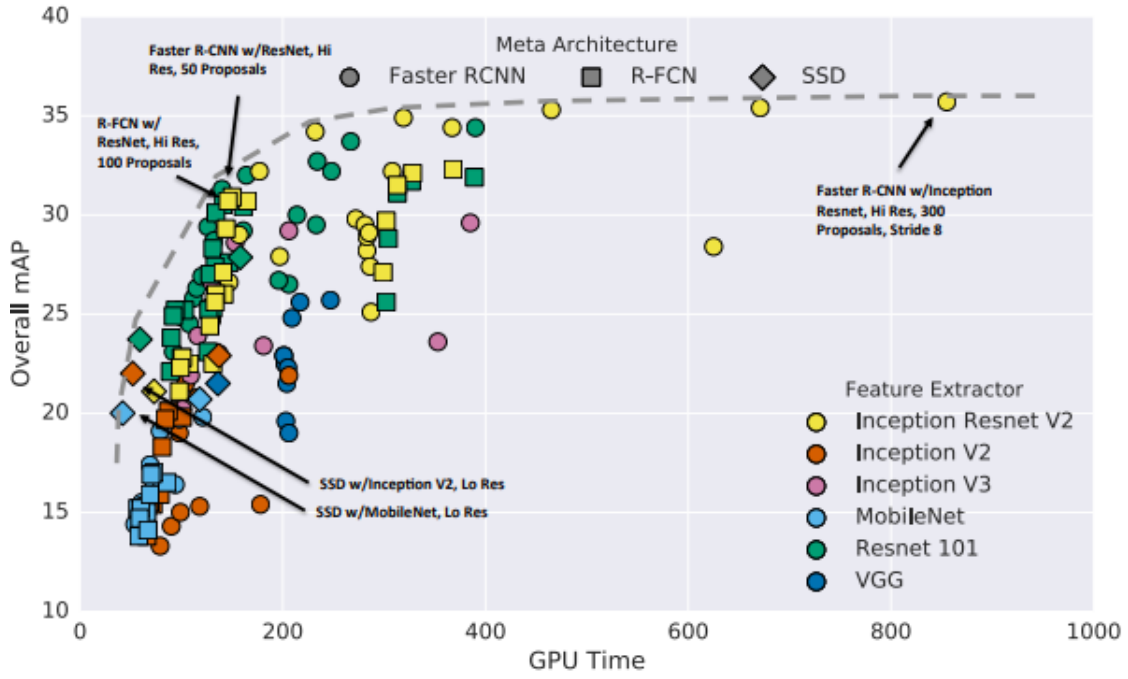


Figure 6.15: Object detection speed vs accuracy tradeoff. Source

6.3 Action classification

Learning to understand videos is a very difficult task, both in terms of algorithms and calculations, the networks have access to not only the appearance information present in single, static images, but also their complex temporal evolution. Such a task could be useful to analyse wildlife behaviours, a kind of information that is very rare.

6.3.1 Datasets

6.3.1.1 Charades

Charades-Ego[18] is a 68,536 instances of activity in 68.8 hours of first and third person video, making it one of the largest and most diverse egocentric data sets in the world. Charades-Ego also shares activity classes, scripts and methodology with the Charades data set, which consists of an additional 82.3 hours of third person video with 66,500 activity instances. Charades-Ego has time annotations and text descriptions, making it suitable for self-centered video classification, localization, subtitling and new tasks using the inter-modal nature of data.

6.3.1.2 Moments in Time

Moments in Time[19] dataset is a large-scale collection of one million short videos, annotated by man, corresponding to dynamic events taking place in less than three seconds. Modelling spatio-temporal dynamics, even for actions made in 3-second videos, poses many challenges: significant events include not only people, but also objects, animals and natural phenomena; visual and auditory events can be time-symmetric (”opening” is ”closing” in the opposite direction) and transient or prolonged. The annotation process (each video is labelled with an action or activity label among 339 different classes), analyze its scale and diversity compared to other large-scale video data sets for action recognition, and present the results of several basic models. treat separately and jointly three modalities: spatial, temporal and auditory. The Moments in Time dataset, designed to cover many visual and auditory events and a wide variety of events, can be a new challenge in developing models that adapt to the level of complexity and abstract reasoning that a human deals with on a daily basis.

6.3.2 Models

6.3.2.1 AssembleNet

Standard CNN video CNN architectures have been designed by directly extending architectures designed for image understanding to a third dimension (using a limited number of spatio-temporal modules such as 3D convolutions) or by introducing a handmade two-stream design to capture both appearance and motion in videos. AssembleNet[20] interpret a video CNN as a collection of multi-flow space-time convolution blocks connected to each other, and propose the approach of automatically searching for neural architectures with better connectivity for video comprehension. This is done by developing a population of overly connected architectures guided by learning the connection weight. Architectures combining representations that do not have different types of input (i.e. RGB and optical flow) at multiple temporal resolutions are sought, thus allowing different types or sources of information to interact. AssembleNet, surpasses previous approaches to public video data sets, in some cases with a large margin.

6.4 Visual object tracking

Visual object tracking is an interesting task that could also help to understand the species behaviour, for example if you can see on large amount of data that certain animal often goes into a certain direction ...

6.4.1 Datasets

6.4.1.1 VOT

The VOT2018[21] Challenge, followed by visual objects, is the sixth annual comparative analysis of monitoring organized by the VOT initiative. The results of more than 80 trackers are presented; many are leading trackers published at major computer vision conferences or in journals in recent years. The evaluation included the standard VOT method and other popular short-term follow-up analysis methods and a ”real time” experiment simulating a situation in which a tracker treats images as if they were provided by a continuously operating sensor. A long-term monitoring sub-challenge has been introduced for all standard VOT sub-challenges. The new sub-challenge addresses the long-term monitoring properties, namely the management of target disappearance and reappearance. A new data set has been compiled and a performance evaluation methodology focusing on long-term monitoring capacities has been adopted. The VOT toolbox has been updated to support standard and long-term short-term monitoring sub-challenges. The performance of the follow-ups tested generally far exceeds the standard reference levels.

6.4.2 Models

6.4.2.1 THOR

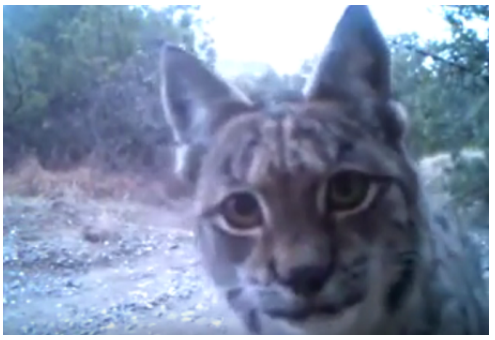
Currently, the state-of-the-art algorithm to date on the VOT challenge is called thor[22]:
This algorithm focus on the construction of holistic object representations for tracking. The structure exploits the idea of obtaining additional object models during the tracking process. The resulting representation contains information beyond the location of the truth object on the ground provided to the system. It is then useful for orientation, but also for other tasks requiring a visual understanding of objects. Strong empirical results on monitoring benchmarks indicate that the method can improve the performance and robustness of underlying monitoring while reducing its speed. In addition, the method is able to reproduce current results, while using a simpler and older network architecture and operating three times faster.

6.5 Wild life

During the past decades, engineers and wildlife researchers have developed various hardware technologies for professionals to monitor individual mammals, including very high frequency (VHF) radio tracking[23][24][25], and Global Positioning System (GPS) tracking [26][27][28], wireless sensor networks[29][30], animal-mounted video monitoring systems [31], drones to captures images from above[32] and ultimately motion-triggered cameras which are useful to capture images only when an animal is present and in his natural environment including at night thanks to infrared.

A motion-triggered camera is a camera with an automatic gesture detection system, which activates the shot as soon as the sensor detects a movement (see example on Figure 6.16). Its continuous image analysis operation is similar to the proximity sensor with the additional capabilities to be transported, adapt to a medium, record and process images.

(a) Camera trap. Source



(b) Lynx taken by camera trap. Source

Figure 6.16: Camera trap & Picture taken by it

6.5.1 Datasets

6.5.1.1 iNaturalist

iNaturalist is a citizen science project and an online social network of naturalists, citizen scientists and biologists based on the concept of mapping and sharing biodiversity observations around the world. iNaturalist is accessible through its website or mobile applications. Observations recorded with iNaturalist provide valuable open data for scientific research projects, conservation agencies, other organizations and the public. The project has been described as "flagship for mobile natural history applications. The dataset consists of 859,000 images from over 5,000 different species of plants and animals. iNaturalist is different from camera trap datasets, it is probably between camera trap and ImageNet-like dataset since it is taken by amateurs.

6.5.1.2 iWildCam 2018 Challenge Dataset

iWildCam[33] consists of 292,732 images on 143 locations in American SouthWest, each labelled either as containing an animal or as empty.
iWildCam is also a challenge in which training data and test data come from different regions. The species observed in each region overlap, but are not identical.
iWildCam 2018 is the biggest public camera-trap dataset for southwest wildlife.

6.5.1.3 North American Camera Trap Images

This data set[34] contains 3.7 million photographic trap images from five different locations in the United States, with tags for 28 animal categories, mainly at the species level (for example, the most common tags are cattle, boars and deer). About 12% of the images are labelled as empty. North American Camera Trap Images is the biggest public camera-trap dataset for north American wildlife

6.5.1.4 Snapshot Seregenti

Hundreds of camera traps in Tanzania's Serengeti National Park open a powerful new window into the dynamics of Africa's most elusive wildlife species. The camera traps have been in continuous operation since 2010 and has produced millions of images, classified by volunteers.

This data set contains approximately 2.5 million camera trap image sequences, for a total of 6.7 million images, from seasons one to ten of the Snapshot Serengeti project. Tags are provided for 55 animal categories, mainly at the species level (for example, the most common tags are wildebeest, zebra and Thomson's gazelle). About 75% of the images are labelled as empty. There is approximately 150,000 selection box annotations to approximately 78,000 of these images.

Snapshot Seregenti is surely the biggest public camera-trap dataset for African wildlife.

6.5.1.5 Synthetic data

Synthetic data and simulated environment are really promising, the advances in the field of computer graphics enable the modeling of 3D world looking closely to reality and can be generated infinitely. Airsim[35] is an open source system designed to form autonomous systems. AirSim provides realistic environments, vehicle dynamics and multimodal detection to researchers who build autonomous vehicles using AI to improve their safe operation in an open world.
Engineers who build autonomous systems can create accurate and detailed models of systems and environments, making them intelligent using methods such as in-depth learning, imitation learning and reinforcement learning. Tools such as Bonsai can be used to train models for a variety of environmental conditions and vehicle scenarios in the cloud on Microsoft Azure - much faster and safer than what is feasible in the real world. Once the training is complete, designers can deploy these trained models on real hardware. Airsim has evolved towards a general purpose simulated environment and now contains also wildlife and camera-traps (see Figure 6.17).

(a) Airsim elephant. Source

(b) Real elephant. Source

Figure 6.17: Example showing differences between synthetic and real animal

Synthetic data are still far from real data but it can still be used to learn the general pattern of specific data, the advantage is that synthetic data are unlimited.

6.5.2 Models

In the case of camera-trap images, we can't simply directly use the same models that has been trained on animal images taken by professionals. Motion triggered camera images have lower quality and are highly cluttered with low contrast, and with dramatic background motion. Pattern detection also needs to handle a wide variety of animal body sizes, appearance, and poses (see differences example in Figure 6.18).

(a) Camera-trap sample

(b) ImageNet sample

Figure 6.18: Example showing the differences between camera-trap image and professional photograph quality image

Often, the images taken by camera trap are not free of right, causing a huge difference in quantity of publicly available datasets compared to general images datasets.
Transfer learning is an interesting solution when there is a lack of the specific data we want to train on. Used by many[36][37][38][39][40], usually pre-training a model with ImageNet or similar general dataset then fine tuning with camera trap datasets like iWildcam and such.

6.6 Conclusion

Classification and detection are useful for wildlife monitoring, but action classification and other ”exotic” tasks may be interesting in the future. The advantage of object detection over classification is to know the position of the objects as well as to have several results per image, that’s why I chose to focus on detection in the rest of this work

Implementation

7.1 Methodology

As a reminder of the objective, there are camera traps that produce images and videos, ecologists want to have statistics on species to prevent their extinction. At the moment, they recognize and count species manually, spending hours watching videos and photos, it takes a lot of time and is therefore very expensive.

The solution is to automatize this process with computer vision, this must meet the ”four easy constraints”:

- Easy to deploy because ecologist usually don’t have much computer science knowledge.
- Easy to maintain because we don’t want something that need 10 hours/day monitoring if anything went wrong.
- Easy to scale, being able to handle either 1000 images / month or 1 000 000 images / month.
- Easy to price, having a fully transparent estimation of the different costs.

7.1.1 Existing solutions

7.1.1.1 Microsoft Camera trap

Microsoft AI for Earth propose a set of tools and algorithms especially designed for camera traps monitoring[41]. It goes from data parsing scripts, pre-trained models on camera-trap images or web interfaces to use their models ...

An example of the model has been developed (see Figure 7.1).

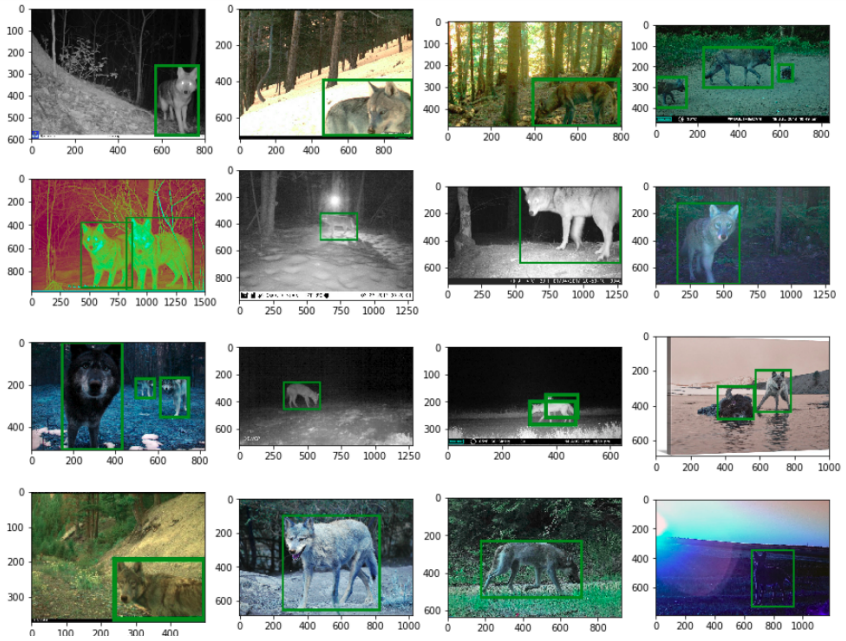


Figure 7.1: Example usage of Microsoft's camera trap model. Source

This solution is not usable in production, it is a bit messy and no clear direction is given.

7.1.1.2 Google DeepMind camera trap

DeepMind, one of the lead actor in artificial intelligence research, the author of the major breakthroughs in AI, such as AlphaGo[42], AlphaStar[43] and much more are begining to take an interest in accelerate ecological research using machine learning[44]. Their recent announcement is nevertheless rather vague, just an article and no code is available, they seems to work on training algorithm on the Seregenti Snapshot datasets.

Their solution will probably be interesting but not available in the short term

7.1.2 Considered solutions

Running a dedicated server is hard to deploy, it requires the implementation of containers, container-orchestration pipelines and other infrastructure tools. To maintain it properly you also need to develop additional monitoring tools in order to keep track of its state. It is also difficult to scale due to the work required on hardware side and finally the price can still be high.

With cloud providers (Amazon, Google, Microsoft, DigitalOcean ...) no need to worry about infrastructure, hardware, easy to deploy, maintain, scale and relatively price transparent.

We use Google Cloud Platform as a choice of experience, long-term potential and machine learning and big data orientation.

7.2 Technologies

7.2.1 Machine learning

Python has been chosen because it is the most popular programming language for machine learning[45] due to numerous advantages:

- Machine learning, vectors, math, visualisation libraries
- Massive community
- Machine learning frameworks
- Fast results

Next, usually the choice of framework have to been done between the most popular: Google's Tensorflow[46] or Facebook's Pytorch[47].
Both are open-source, the most important difference between the two is the way these frameworks define the computational graphs. While Tensorflow creates a static graph, PyTorch believes in a dynamic graph. It means that Tensorflow is more difficult to learn at the moment, but Tensorflow will soon be released as 2.0 and it will be possible to create dynamic graph like Pytorch. Tensorflow has way bigger community, more tools (Tensorboard, Tensorflow-serving ...). Also it is obviously easier to use Google services with Tensorflow than Pytorch, for all these reasons Tensorflow has been chosen.

7.2.2 Google Cloud Platform

Google Cloud Platform is the platform that brings together Google's various cloud services. The Google Cloud Platform is a suite of cloud services offered by Google. The platform includes various cloud services for computing, storage, networking, Big Data, machine learning, Internet of things, security, cloud management and application development that are directly launched on Google's servers.

7.2.2.1 Advantages

Like all cloud platforms, Google's platform has the advantage of sparing companies the management of an infrastructure, server provisioning and network configuration. In addition, Google also highlights the evolutionary aspect of its infrastructure. Constantly updated and optimized, the platform benefits from Google's expertise and is efficient, economical and secure.
With the fully managed, server-less computing system, users can move from prototype to global production without worrying about capacity, reliability or performance. Other strengths of the Google Cloud Platform include a data center backbone network composed of thousands of kilometers of fiber optic cables combined with an advanced networking solution and edge caching services to deliver extreme performance.

7.2.2.2 Services

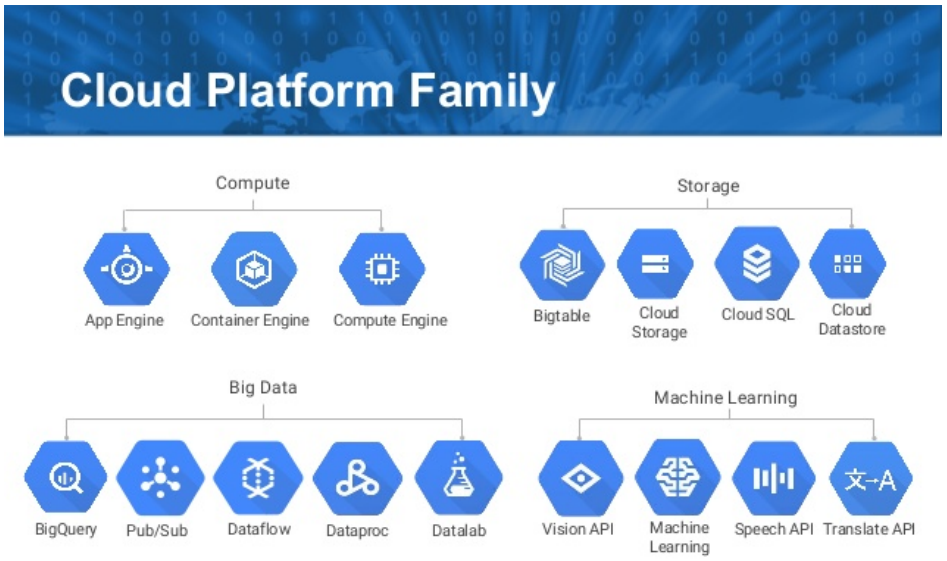


Figure 7.2: Google Cloud Platform services

The Google Compute Engine is an Infrastructure as a Service (IaaS) that allows users to launch instances of virtual machines. This allows them to run their workloads on the cloud (see Figure 7.2).

- The Google App Engine is a Platform as a Service (PaaS). It allows software developers to access a scalable hosting offer. Developers can use an SDK to develop software that is compatible with the App Engine.
- Google Cloud Storage is a cloud storage platform designed to store large unstructured data sets. Google also offers database storage options, such as the Datastore for NoSQL storage, or CloudSQL for MySQL. We also find the native database Bigtable.
- The Google Container Engine is a management and orchestration system for Docker containers running on Google's public cloud. This system is based on the Google Kubernetes container orchestration engine. The Google Cloud Platform also offers application development and integration services. For example, Cloud Pub/Sub is a real-time messaging service that allows messages to be exchanged between applications. Similarly, Endpoints allows developers to create services based on RESTful APIs and make these services very accessible.

7.2.2.3 Other services

Google also offers higher-level services on its Cloud platform, such as those dedicated to Big Data and Machine Learning. Google's Big Data services are used to process and analyze data. Google BigQuery allows you to search for data sets of several terabytes, for example. Dataflow is a data processing service designed for data analysis, extraction, transformation and loading. Dataproc offers Apache Spark and Hadoop services for Big Data processing. It also integrates the databases of Cassandra and MongoDB.
In terms of artificial intelligence, Google offers its Cloud Machine Learning Engine(Google Cloud AI Platform), a managed service that allows users to develop and train Machine Learning models. Different APIs are also available for the translation and analysis of speech, text, images or videos. AI Platform allow to deploy any model in the Tensorflow SavedModel format, while the different API such as Video Intelligence is simple to use but is, at the moment, only for general use cases, for example it won't we suitable for camera-trap images.

7.2.2.4 Prices

Like most cloud service providers, Google Cloud Platform pricing is based on a pay-as-you-go (pay-per-use) business model, which means that users pay according to the resources they consume. Resource consumption is accurately calculated to the minute. According to Google, the platform's rates are on average 60% lower than those of other providers[48].
The other advantage of this model is that users do not have to pay upfront fees. Similarly, billing ends as soon as the user ceases to use the services without having to pay termination fees. For Google Compute Engine and Cloud SQL services, Google also offers an automatic discount system of up to 30% on the most used workloads. Rates vary between the different services and should therefore be consulted on a case-by-case basis on the official website.

7.2.3 Front end

We have chosen to use Javascript, which is the most popular language for the front end. Front end web development is a field that is evolving very quickly and solutions are quickly becoming obsolete.
Especially, we use Lit-Element which is created to support a component-oriented approach to front-end Web development.
Lit-Element is an open-source JavaScript library used to develop Web applications using Web components. Created by Google developers and contributors on GitHub.
The advantage of Lit-Element is that you don't need to know in detail how it works unlike popular front end heavy frameworks such as Angular, React, Vue and others.

7.3 Online versus Batch predictions

AI Platform offers two methods for obtaining predictions from trained models: online prediction and batch prediction. In both cases, you transmit input data to a machine learning model hosted in the cloud and obtain inferences for each instance of data (see the differences in Figure 7.3).

Online prediction	Batch prediction
Optimized to minimize the latency of serving predictions.	Optimized to handle a high volume of instances in a job and to run more complex models.
Can process one or more instances per request.	Can process one or more instances per request.
Predictions returned in the response message.	Predictions written to output files in a Cloud Storage location that you specify.
Input data passed directly as a JSON string.	Input data passed indirectly as one or more URIs of files in Cloud Storage locations.
Returns as soon as possible.	Asynchronous request.
Accounts with the following IAM roles can request online predictions: • Legacy Editor or Viewer • AI Platform Admin or Developer	Accounts with the following IAM roles can request batch predictions: • Legacy Editor • AI Platform Admin or Developer
Runs on the runtime version and in the region selected when you deploy the model.	Can run in any available region, using any available runtime version. Though you should run with the defaults for deployed model versions.
Runs models deployed to AI Platform.	Runs models deployed to AI Platform or models stored in accessible Google Cloud Storage locations.
Can serve predictions from a TensorFlow SavedModel or a custom prediction routine (beta).	Can serve predictions from a TensorFlow SavedModel.
$0.0401 to $0.1349 per node hour (Americas). Price depends on machine type selection.	$0.0791 per node hour (Americas).

Figure 7.3: Online versus batch predictions. Source

It is generally preferable to use online prediction to make requests in response to an application entry or in other situations where rapid inference is required.

Batch prediction is ideal when processing accumulated data, when you do not need immediate results. For example, a periodic task that makes predictions from all the data collected since the last task.

You must also make your decision taking into account potential variations in the cost of prediction.

Here we obviously chose batch prediction because our goal is to have statistics of species over long times, we don't need to have instant result, the user will drop big amount of videos / photos that will be processed.

7.3.0.1 Latency of batch prediction

If you use a simple model and a small set of input instances, you will find that, to complete identical prediction requests, there is a considerable difference in the time required if online prediction is used compared to batch prediction. A batch task can take several minutes to complete predictions that would be returned almost instantly by an online request. This is a side effect of the difference between the infrastructures used by the two prediction methods. AI Platform allocates and initializes resources for a batch prediction task when you send the request. The online prediction is usually ready to be processed at the time of the request.

7.4 Architecture

The ultimate high-level objective of this is to allow users to easily upload visual data in a secure and privacy preserving manner and get useful insights about wildlife populations through visualisation like graphics, heat maps ...

The core of this architecture is powered by Google Cloud AI Platform, a service that allow quickly and cost-effectively deployment of machine learning models to production. Google Cloud AI Platform has nevertheless a few flaws, such as the unavailability of GPUs for inference which are very powerful for computer vision[49], the lack of documentation or the limit of 250mb models.

The second most important part is Google Cloud App Engine, which allow to create and deploy an application on a fully managed server-less platform. It allow easy scaling without having to worry about managing the underlying infrastructure.

Then come the storage component, Google Cloud Storage is a web-based RESTful online file storage service for storing and accessing data on the Google Cloud platform infrastructure. This service combines the performance and scalability of Google's cloud with advanced security and sharing features.

To persist data, a NoSQL solutions of Google is used: Cloud Datastore which offers great scalability for the applications.

This tool automatically manages the partition and replication of data allowing a durable and highly available database that can dynamically evolve to absorb the load of the applications.

Cloud Datastore offers a multitude of features, such as ACID transactions, SQL queries, indexes and more.

Lastly, multiple small event-triggered codes have been deployed through Google Cloud Function, GCF is a server-less runtime environment for creating and connecting cloud services.

Cloud Functions are usually used to write simple and single-use functions that are attached to events issued by the cloud infrastructure and services.

The Cloud function is triggered when an observed event is triggered. The code runs in a fully managed environment. There is no need to provision the infrastructure or worry about server management.

Multiple cloud pipelines have been considered, their advantages and disadvantages have been studied carefully, where some infrastructure fails, where it succeed.
To conduct such a process, a list of use cases needed to be put down:

1. I want to send an image and have the prediction right away
2. I want to send 1000 images and have the prediction as soon as available
3. I changed my model, I want to make all the predictions again
4. I want to send a video, and have the predictions right away
5. I want to send 100M images and have the predictions as they happen, knowing how long it will be before I have the predictions and how much it will cost me
6. I want to send 1000 videos and know time and cost
7. I want to automatically send videos from camera trap device
8. I have 1000 cameras trap that will send videos, how much it will cost me

Of course, there are probably more cases, but it is at least a few simple cases where an architecture can have bottlenecks.

7.4.1 Pipeline version 1

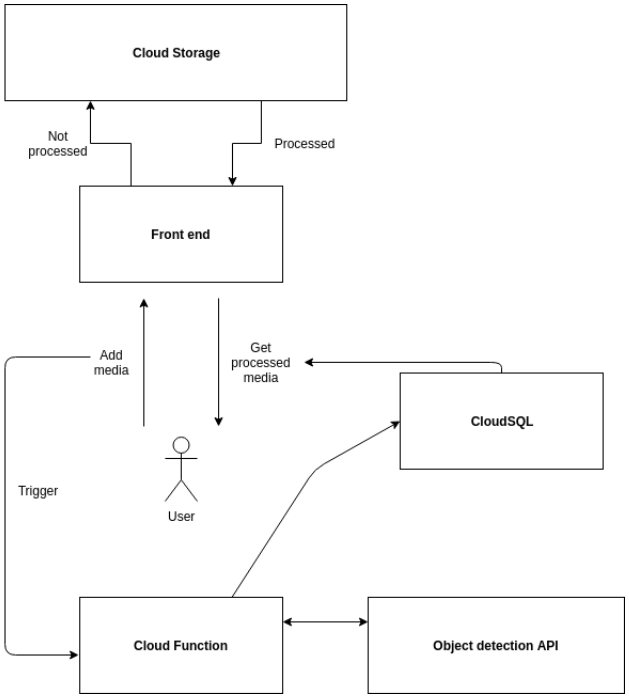


Figure 7.4: GCP pipeline version 1

In this version, data would be stored in Google Cloud Storage, the front end would be hosted on a Google Cloud App Engine using NodeJs programming language.
A Golang-written (for performance) cloud function will be triggered when data have been added, calling the AI Platform's hosted model to process it (see 7.4).

It is a first nice simple solution in theory, but in practice it encounters few problems

1. No queue (if you add 100 images, the cloud function will ask a lot to AI Platform, it should wait the end of each processing through a queue)
2. On the cloud function side again, Python has way more tools to process images, machine learning ... than Golang which is really focused on server side communication

Regarding the use cases here is how it goes for this version:

1. Works and on average shouldn't take more than a few seconds maximum: success
2. The first images will be processed effectively, then a lot of errors and crashes will occurs, overloading the model: fail
3. All images will have to be uploaded again: fail
4. This version doesn't handle videos: fail
5. Everything will crash, no estimations: fail
6. This version doesn't handle videos: fail
7. This version doesn't handle videos: fail
8. This version doesn't handle videos: fail

7.4.2 Pipeline version 2

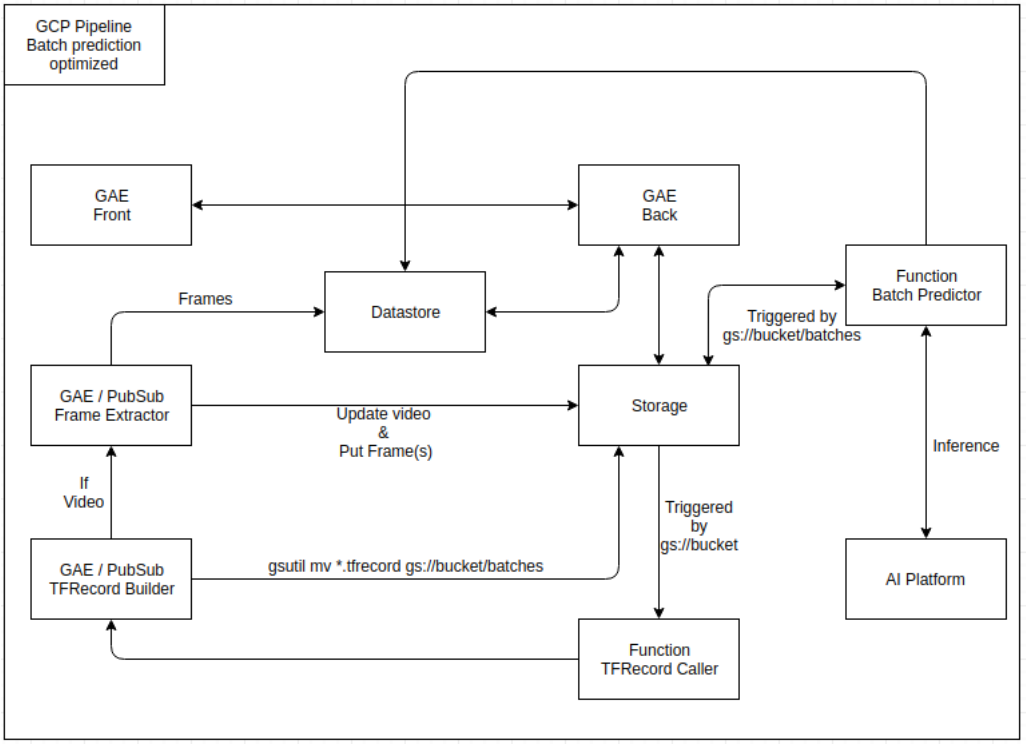


Figure 7.5: GCP pipeline version 2

The major change here is the use of batch prediction which takes a different kind of input. Online prediction usually either require an input four dimensions vector of the shape (-1, -1, -1, 3) (first dimension is the number of images, so in online it is always 1, second dimension is the width, third the height and lastly the channel).

Example of 4-d vector input in Python, converting a Numpy array into a serializable list

```
{"inputs": img.tolist()}
```

Either it can accept a byte string of the shape (-1), encoded for example in Python:

```
{"inputs": base64.b64encode(jpeg_data)}
```

Batch prediction require the sames types of inputs but stored in a file put in Google Cloud Storage, the file has either a JSON format, a TFRecord format which is the archive format of Tensorflow or a TFRecord GZipped.
TFRecord is way lighter than JSON so an implementation of a TFRecord builder has been developed and deployed in an App Engine.
The second advantage of this version over the first is the introduction of queues through Google Cloud PubSub which handle the communications between Google Cloud services, making concurrent-running components less likely to crash. This version also includes some more little cloud functions and is more modular (see Figure 7.5).
The problem in this version is the disappearing of online predictions which can be useful for fast predictions, also too many App Engine are used while simple cloud functions would have been enough.
Regarding the use cases here is how it goes for this version:

1. This version doesn't handle online prediction: fail
2. Works and on average shouldn't take more than a few minutes maximum: success
3. All images will have to be uploaded again: fail
4. This version doesn't handle online prediction: fail
5. Possible but no estimations: success
6. Possible but no estimations: success
7. Possible, depending in the amount of scaling required: success
8. Possible, depending in the amount of scaling required but no estimations: success

7.4.3 Pipeline version 3

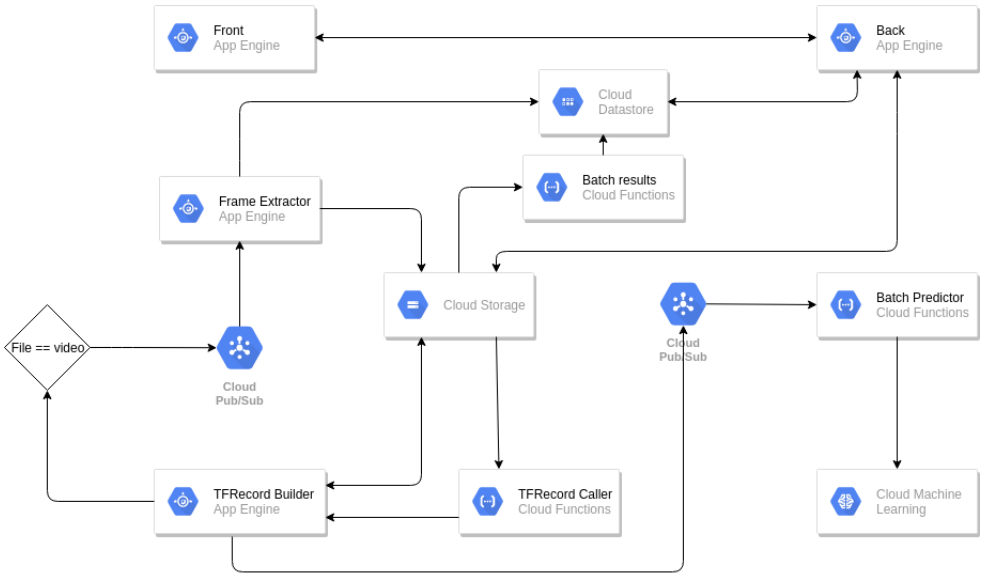


Figure 7.6: GCP pipeline version 3

The third version is quite similar to the previous one, some simplifications have been done (see Figure 7.6).

It still lacks online predictions and overuse Google App Engine which are quite expensive.

Regarding the use cases here is how it goes for this version:

1. This version doesn't handle online prediction: fail
2. Works and on average shouldn't take more than a few minutes maximum: success
3. All images will have to be uploaded again: fail
4. This version doesn't handle online prediction: fail
5. Possible but no estimations: success
6. Possible but no estimations: success
7. Possible, depending in the amount of scaling required: success
8. Possible, depending in the amount of scaling required but no estimations: success

7.4.4 Pipeline version 4

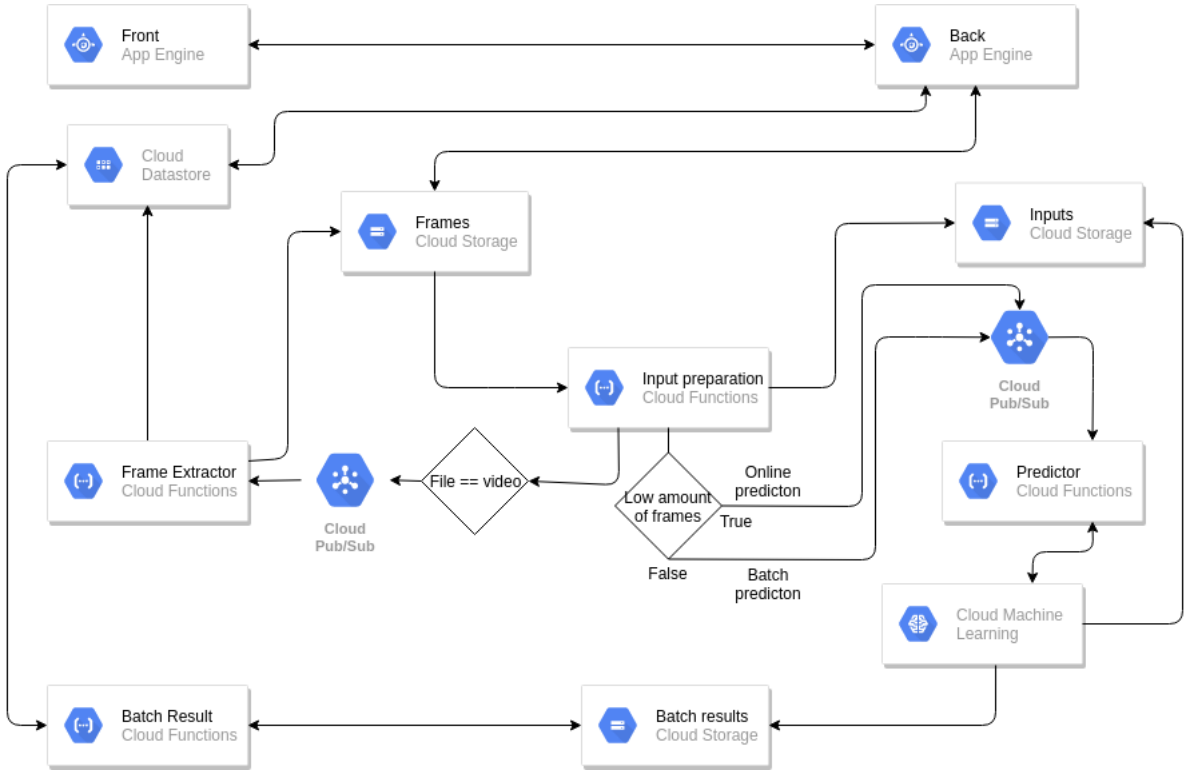


Figure 7.7: GCP pipeline v4

The fourth version have evolved through his predecessor, fixing bottlenecks, improving the overall pipeline, handling both batch and online predictions (see Figure 7.7).

The choice between online and batch prediction is customizable by the user, by default batch prediction are preferred over a certain threshold of data to process, under this threshold online predictions are called for fast predictions.

Regarding the use cases here is how it goes for this version:

1. Works and on average shouldn't take more than a few seconds maximum: success
2. Works and on average shouldn't take more than a few minutes maximum: success
3. All images will have to be uploaded again: fail
4. Works and on average shouldn't take more than a few minutes maximum: success
5. Possible but no estimations: success
6. Possible but no estimations: success
7. Possible, depending in the amount of scaling required: success
8. Possible, depending in the amount of scaling required but no estimations: success

In the end, the only lacking features are price and time estimations before and after upload and the change of model, both are easy to implement and shouldn't take more than a few hours, the biggest part of the architecture was the front end, back end and make everything communicate well together, in addition of the fact AI Platform batch jobs require 6 minutes to boot at cold start, so it took a while to test everything.

7.4.5 Components

Now all components of this final pipeline will be described in detail.

Front component is in charge of directly interfacing with user, it allows to upload images and videos, get prediction results, see statistical graphics about the occurrences of animals over time and space. It communicates with **Back** to send images, retrieve predictions and statistics. **Front** is implemented with Lit-Element as a single page application.

Back is a REST API implemented in JavaScript with Express package, it saves information about frames and predictions in **Cloud Datastore**. It stores images and videos in **Cloud Storage**. It runs on App Engine so it can scales to any number of instances to handle higher traffic.

Cloud Datastore is a NoSQL database which stores frames, videos, predictions and detected objects, it is convenient to be able to iterate fast without having to define the schema of the database first.

Cloud Storage is a scalable filesystem to store large amount of files, it is used to store images, videos, batch predictions inputs and outputs.

Frame extractor is a cloud function that split videos into frames and store them in **Cloud Storage** and their metadata in **Cloud Datastore**.

Input preparation is responsible for building the input of both online and batch predictions from frames and storing it on **Cloud Storage**. It retrieves the frames from **Cloud Storage** and publish on **Cloud PubSub** queue. It chooses between online or batch prediction based on different parameters to fulfill user needs, for example if the user has only few images to tag it makes sense to use online prediction, if he has ten thousands of videos it makes more sense to use batch prediction.

Cloud PubSub is used to limit the calling rate of batch and online prediction. It is useful because Google API are rate-limited to avoid waste of resource.

Batch prediction AI Platform receive batch of frames and start a job which automatically scales in many workers and output the predictions efficiently for large number of frames, launching a job takes several minutes so it is not used for few frames. Results are stored in **Cloud Storage**.

Online prediction AI Platform produces prediction for one frame with low latency, the result is available in the HTTP answer.

Batch result is a cloud function that update the predictions once they are done into **Cloud Datastore**.

7.5 Data flows, number of operations : price & time estimations

In the last few years, people are still hesitating to dive in cloud providers like Amazon, Google, Microsoft with the fear of multiple concerns: losing control, security, data protection, performance and price. The cloud services are usually difficult to estimate in price, here an example use case showing the consumption of storage and cloud operations using this solution in production is described. The cost in term of storage, Datastore, ... is detailed in next parts

7.5.1 Data & Ops

A typical use case would be:

1. Reach main page
2. Upload a picture
3. Back on main page to see the results (after a while of course)

7.5.1.1 Storage

- Two megabytes image
- One operation to store the image

7.5.1.2 Datastore

- Storage of the annotation mapping approximately <= 1000 rows
- Two rows Frame entity (store, then update when prediction are ready)
- Predictions into Prediction entity 1 row
- Objects detected into Object entity < 50 rows (being large)
- One operation put class mapping
- One operation put frame
- One operation prediction
- One operation put objects

7.5.1.3 Function

- One operation input preparation
- One operation prediction

7.5.1.4 PubSub

- One message being stored and published

7.5.1.5 App Engine

- Two operations frontend rendering main page
- One operation front render upload page
- One operation back call frames list
- One operation back call frames upload

7.5.1.6 AI Platform

- One operation online prediction

7.5.2 Price & Time

Cloud Storage	Storage	200 GB	USD 4.00
Cloud Storage	Class A operations (millions)	10 1000	USD 0.50
Machine Learning		0.5661	USD 9.26
Pub/Sub	Message Volume	20 GB	USD 0.40
Datastore Storage	Cloud Datastore	2 GB	USD 0.00
Datastore	Cloud Datastore	10000000	USD 5.09
Datastore	Cloud Datastore	1000000	USD 0.71
App Engine standard environment	Instances	1460 GB	USD 30.44
Cloud Functions	prediction	250000	USD 59.49
Cloud Functions	input	250000	USD 59.49
Total Estimated Monthly Cost			**USD 169.38**

Figure 7.8: Example price estimation. Source

You can see on Figure 7.8 the largely estimated price for 100 000 images per month.

This price can probably be highly lowered since no optimization has been done and the ”default” and even non needed higher configurations of services have been chosen (2 permanently running App Engine instances, high memory capacity for cloud functions ...).

In addition, creating this estimation helped to recognize where the solution can be optimized to reduce the overall cost.

7.6 Statistics

Statistics based on graphs and other visualizations are available to monitor the evolution of species and their behaviour, some interesting information are:

- Occurrence of species over time and zone (plots, charts, histograms, heatmap ...)
- Co-occurrence of species, showing the link between them

Once a certain amount of data has been generated, one can even imagine using predictive models, from simple linear regression, random forests, boosted trees to neural networks for prevention.
Algorithms to detecting anomalies such as principal component analysis could also help analysing the changes and alert ecologists, for example if a species is suddenly in a sharp decline, this could potentially be the cause of poaching.
Another example would be the strong growth of an invasive mosquito species that would be automatically detected by such algorithms above a certain threshold and would prevent the competent authorities

7.7 Used models

Here a choice needed to be made to take the best model in term of speed / accuracy (mAP) tradeoff AND less than 250mb model to respect AI Platform constraint.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes

Figure 7.9: COCO trained models, speed / accuracy tradeoff. Source

Another constraint was the fact that changing the graph makes the file size grow.
Faster RCNN NAS is the most accurate model obviously because of its neural architecture search algorithm, but 1833 is very high compared to others (see Figure 7.9). An interesting one was SSD ResNet 50 FPN, ending around 180mb after the graph modification.

7.8 Performances

With AI Platform configured with 72 max workers, 1 CPU core and a SSD ResNet50 FPN model, it takes about 20 minutes to process a batch of 10 000 images with a resolution of 100x100 (see Figure 7.10)

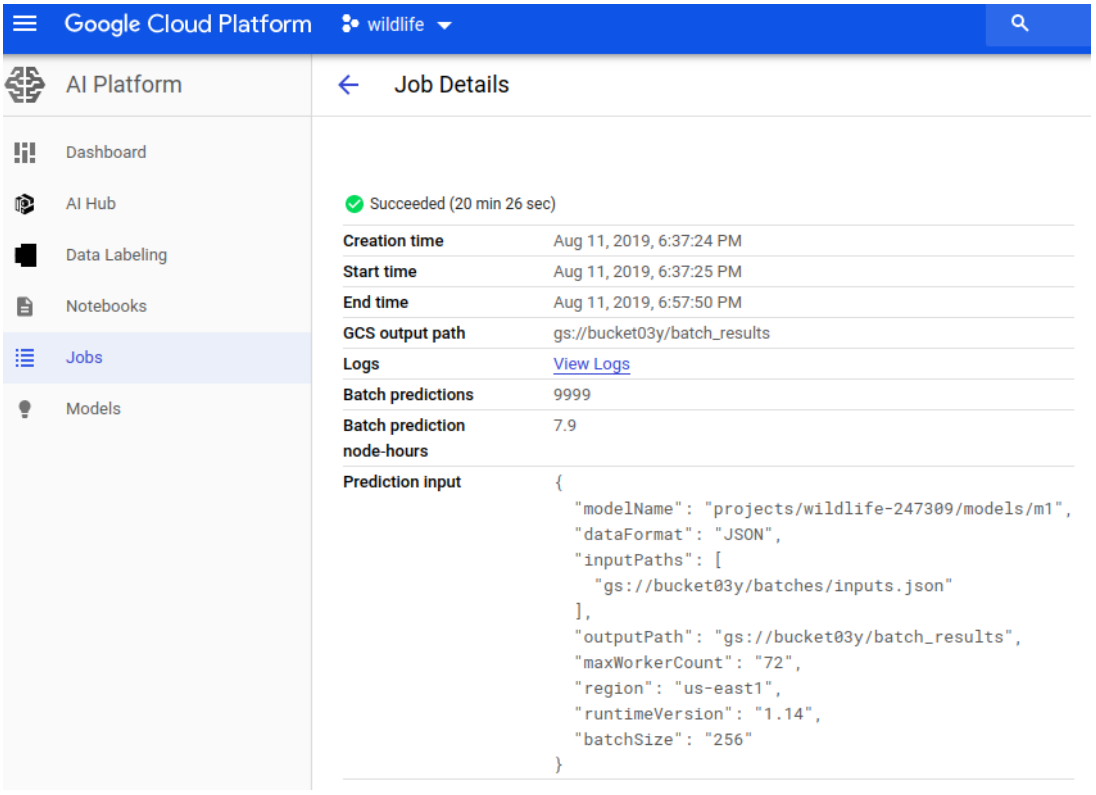


Figure 7.10: Batch prediction job of 10 000 images

On 7.10 you can see that in this context it has cost 7.9 node hours at the price of $0.0791[1] which makes a total price of about $0.6.

[1] see https://cloud.google.com/ml-engine/docs/pricing

7.9 Optimization

In the world of machine learning, the optimization of training is receiving a lot of attention. There is much less information available on inference optimization.
Yet, isn't serving prediction models the point of ML ?
Serving performance can have a significant impact on the ML value for your use case. Indeed, the cost of inference can be a major factor in the total return on investment of an ML application.

7.9.1 Latency (and size) count

When it comes to optimizing production models, we are mainly concerned with three things:

- Model size
- Prediction speed
- Prediction rate

When serving ML, the size of the model is important. Of course, smaller models use less memory, less storage and network bandwidth, and they load faster. In some cases, hardware memory constraints or service limitations may impose a limit on the size of the model. For example, the Machine Learning Engine service on Google Cloud (GCP AI Platform) sets a default size limit of 250 MB for models.
When we use hardware acceleration for prediction purposes, we must ensure that our model fits into the memory of the acceleration device. The size of the model has a particular impact in situations where we serve it on a peripheral or mobile device with limited capacity. We want the model to be downloaded as quickly as possible, using as little network bandwidth as possible and using as little memory and storage capacity as possible.
Prediction speed is another metric that is important. When we make our online inference, we generally want the results to be returned as quickly as possible. In many online applications, latency is critical to the user experience and application requirements. But we care about the speed of inference, even when we process our data in batches. The inference speed is directly related to the cost of the service, as it is directly related to the amount of computing resources required to make a prediction. The time required to make a prediction will always be a critical variable in any formula that measures the rate of prediction.

Faster forecasts mean more prediction throughput on the same hardware, resulting in lower costs. The prediction rate is a measure of the number of predictions that our system can make in a given time period. In addition to the prediction speed mentioned above, other system attributes are involved in determining throughput, including batch processing of forecasts, hardware acceleration, load balancing and horizontal scaling of service instances.

7.9.2 Exported model formats in TensorFlow

TensorFlow has several model serialization formats, but the most important ones to know are the GraphDef and SavedModel formats.

- The GraphDef format is a version of the ProtoBuf[50] serialization protocol, in text or binary form, that codes the definition of a TensorFlow graph. A GraphDef can also include the weights of a trained model, but it does not have to - the weights can be stored in separate control point files.
- The SavedModel format combines a GraphDef with control point files that store weights, all gathered in a folder.

7.9.3 Tools and techniques

TensorFlow offers several techniques to reduce the size of a model and improve prediction latency. Here are some of them:

- Freeze: Convert variables stored in a SavedModel control point file to constants stored directly in the model graph. This reduces the overall size of the model.
- Pruning: Remove unused nodes in the prediction path and graph outputs, merging duplicate nodes and cleaning other node operations such as summary, identity, etc.
- Constant folding: Substitute the values of known constants in expressions at compile time in the sub-graphs of the model
- Batch standard folding: Fold the multiplications introduced in the batch normalization into the weight multiplications of the previous layer.
- Quantization: Convert floating point weights to lower accuracy, such as 16 or 8 bits.

For mobile deployment, there is also TFLite, which performs 8-bit quantification on mobile models.
TensorFlow offer a graph transformation tool to perform most optimizations, which is a C++ command line tool.
The Graph Transform tool is designed to work on models saved as GraphDef files in protobuf format. However, the SavedModel format is the most modern and most supported by other tools and services. This is the only format supported by Google Cloud AI Platform for prediction. The optimization steps, as well as the model format transitions, are as follows:

1. Freeze the exported model: SavedModel => GraphDef
2. Optimize the fixed model: GraphDef => GraphDef
3. Convert the optimized frozen model back: GraphDef => SavedModel

7.10 Other tools built

All the tools are publicly available on the main Github repository.

7.10.1 Inference graph modification

Batch prediction on AI Platform cause a problem, when the prediction are done and written to a file, nothing give any information about which prediction is linked to which input.

To solve this problem, a tool to change the graph of the model needed to be developed.

Tensorflow offer a repository specific to models[51], many tools are available there to work with object detection graphs. Tensorflow also make the possibility to display a graph through a command-line script.

```
1 os.chdir('/root')
2 !python tensorflow/tensorflow/python/tools/saved_model_cli.py show --dir model/saved_model --all
```

```
MetaGraphDef with tag-set: 'serve' contains the following SignatureDefs:

signature_def['serving_default']:
  The given SavedModel SignatureDef contains the following input(s):
    inputs['inputs'] tensor_info:
        dtype: DT_UINT8
        shape: (-1, -1, -1, 3)
        name: image_tensor:0
  The given SavedModel SignatureDef contains the following output(s):
    outputs['detection_boxes'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 100, 4)
        name: detection_boxes:0
    outputs['detection_classes'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 100)
        name: detection_classes:0
    outputs['detection_scores'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 100)
        name: detection_scores:0
    outputs['num_detections'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1)
        name: num_detections:0
  Method name is: tensorflow/serving/predict
```

Figure 7.11: Initial object detection graph

A Tensorflow object detection model's graph input and output is shown on Figure 7.11.

```
1 os.chdir('/root')
2 !python tensorflow/tensorflow/python/tools/saved_model_cli.py show --dir exported_model/saved_model --all
```

```
MetaGraphDef with tag-set: 'serve' contains the following SignatureDefs:

signature_def['serving_default']:
  The given SavedModel SignatureDef contains the following input(s):
    inputs['input_keys'] tensor_info:
        dtype: DT_STRING
        shape: (-1)
        name: input_keys:0
    inputs['inputs'] tensor_info:
        dtype: DT_UINT8
        shape: (-1, -1, -1, 3)
        name: image_tensor:0
  The given SavedModel SignatureDef contains the following output(s):
    outputs['detection_boxes'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 100, 4)
        name: detection_boxes:0
    outputs['detection_classes'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 100)
        name: detection_classes:0
    outputs['detection_scores'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1, 100)
        name: detection_scores:0
    outputs['num_detections'] tensor_info:
        dtype: DT_FLOAT
        shape: (-1)
        name: num_detections:0
    outputs['output_keys'] tensor_info:
        dtype: DT_STRING
        shape: (-1)
        name: output_keys:0
  Method name is: tensorflow/serving/predict
```

Figure 7.12: Changed graph to handle keys

Then you can see on Figure 7.12 the modified graph with the input and output keys used to map an image to its prediction (E.g. I detected a cat at top-left ...)

A fork and modification of Tensorflow Models repository has been done[2] to allow such graph modification.

7.10.2 Gitpod configurations

The growth of cloud services made also available cloud development possible, an example of this is Google Colaboratory[52] which have been of great use but is only for Python, Gitpod[53] allows to develop in most languages.

A customised Docker configuration has been developed to allow ready-to-code environment in a click[3].

[2] See https://github.com/louis030195/models

[3] see https://github.com/louis030195/vision-client

Future Work

8.1 Individual species identification

It could be as much interesting as useful to be able to differentiate a species from another, it could be used to track and understand some behaviours. In order to implement such a thing a similar approach than face recognition algorithm could be used, previous work has been done on bear[54], indeed facenet[55] algorithm is adapted to this kind of task.

8.2 New tasks

In the solution we propose, only object detection is handled, in the future some more tasks could be setup, putting multiple models at disposal to the user.

8.2.1 Video classification

Video classification(action classification) in the field of wild life could be used to understand the behaviours of different species over time and space, while it's probably inaccessible now due to the lacks of data and being a less popular task, it can still be solved later by means of creating datasets specially designed and annotated for this.

8.2.2 Pose estimation

Pose estimation is the task to track an animal's every part of the body small movement in real time. Of similar utility than video classification, it could help to understand species behaviour and interactions. Again, this task imply the creation of datasets designed and annotated for this task.

Conclusion

In this work I presented an easy to deploy, maintain, scale and price cloud solution for wildlife monitoring powered by artificial intelligence.

First, an introduction to the context and who is concerned has been detailed.

Secondly, a quick explanation of the domain of computer vision, what makes it work and its use has been described.

The previous research in this field on the algorithmic and data side, in addition to the different tasks and its potential on wildlife monitoring has been reported.

Then, we tested and showed the results of the publicly accessible solutions, their disadvantages. We developed and tested the considered solutions, proposed incremental concrete cloud pipelines that presented functional results, their price and time consumption and their ability to scale.

A link with ecological research has been established, be it about the possibility to predict species decline or growth and automatically detect anomaly in these populations.

Finally, the future of this solution, with the complex but long-term possible new tasks of computer vision has been expressed.

I hope that this document and the concrete cloud computer vision code will be used to save what is left of biodiversity on Earth, help those that are timid with cloud providers and contribute to the immensely important field of artificial intelligence.

Bibliography

[1] S. CBD. Global biodiversity outlook 1. in convention on biological diversity, 2017.

[2] www.statista.com. Smartphone usage worldwide. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[4] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. *http://yann.lecun.com*, 2010.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] Touvron Hugo, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. *arXiv e-prints*, June 2019.

[7] paperswithcode.com. Benchmark image classification on imagenet. https://paperswithcode.com/sota/image-classification-on-imagenet.

[8] paperswithcode.com. Benchmark image classification on inaturalist. https://paperswithcode.com/sota/image-classification-on-inaturalist.

[9] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[10] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *CoRR*, abs/1811.00982, 2018.

[11] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[12] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[13] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[15] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017.

[16] Vinod Nair Alex Krizhevsky and Geoffrey Hinton. The cifar-10 dataset, 2009.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[18] Gunnar A. Sigurdsson, Abhinav Gupta, Cordelia Schmid, Ali Farhadi, and Karteek Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *CoRR*, abs/1804.09626, 2018.

[19] Mathew Monfort, Bolei Zhou, Sarah Adel Bargal, Alex Andonian, Tom Yan, Kandan Ramakrishnan, Lisa M. Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, and Aude Oliva. Moments in time dataset: one million videos for event understanding. *CoRR*, abs/1801.03150, 2018.

[20] Michael S. Ryoo, A. J. Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. *CoRR*, abs/1905.13209, 2019.

[21] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka Čehovin, Tomas Vojir, Goutam Bhat, Alan Lukežič, Abdelrahman Eldesokey, Gustavo Fernandez Dominguez, Alvaro Garcia-Martin, Álvaro Iglesias-Arias, A Alatan, Abel Gonzalez-Garcia, Alfredo Petrosino, Alireza

Memarmoghadam, Andrea Vedaldi, and Andrej Muhič. The sixth visual object tracking vot2018 challenge results, 01 2019.

[22] Axel Sauer, Elie Aljalbout, and Sami Haddadin. Tracking holistic object representations. In *British Machine Vision Conference (BMVC)*, 2019.

[23] L. David Mech. *Handbook of Animal Radio-Tracking*. University of Minnesota Press, ned - new edition edition, 1983.

[24] Rohde & Schwarz. The internet of animals. https://www.rohde-schwarz.com/lv/about/news-press/details/news-library/news-detailpages/the-internet-of-animals-news_detailpage_229349-561345.html, 2018.

[25] Evan Buechley, Michael Mcgrady, Emrah Coban, and Cagan Sekercioglu. Satellite tracking a wide-ranging endangered vulture species to target conservation actions in the middle east and east africa. *Biodiversity and Conservation*, 03 2018.

[26] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *SIGARCH Comput. Archit. News*, 30(5):96–107, October 2002.

[27] Ian A.R. Hulbert and John French. The accuracy of gps for wildlife telemetry and habitat mapping. *Journal of Applied Ecology*, 38:869 – 878, 12 2001.

[28] Schalk Willem Krüger. An inexpensive hyperbolic positioning system for tracking wildlife using off-the-shelf hardware. Master's thesis, North-West University (South Africa), Potchefstroom Campus, 2016.

[29] Ian Akyildiz, Su WY, Y Sankarasubramaniam, and E Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38:393–422, 03 2002.

[30] Robert Szewczyk, Alan M. Mainwaring, Joseph Polastre, John Anderson, and David Culler. An analysis of a large scale habitat monitoring application, 01 2004.

[31] Zhihai He, Wenye Cheng, and Xiao Dong Chen. Energy minimization of portable video communication devices based on power-rate-distortion optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 18:596–608, 2008.

[32] Luis Gonzalez, Glen A Montes, Eduard Puig, Sandra Johnson, Kerrie Mengersen, and Kevin Gaston. Unmanned aerial vehicles (uavs) and artificial intelligence revolutionizing wildlife monitoring and conservation. *Sensors*, 16:97, 01 2016.

[33] Sara Beery, Grant van Horn, Oisin Mac Aodha, and Pietro Perona. The iwildcam 2018 challenge dataset, 2019.

[34] Michael Tabak, Mohammad Sadegh Norouzzadeh, Steven Sweeney, Kurt Vercauteren, Nathan Snow, Joseph M Halseth, Paul A Di Salvo, Jesse Lewis, Michael White, Ben Teton, Raoul Boughton, Bethany Wight, Eric Newkirk, Eric A Odell, Ryan Brook, Anna Moeller, Elizabeth G Mandeville, Jeff Clune, Ryan Miller, and Peter Schlichting. Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, pages 1–6, 11 2018.

[35] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *CoRR*, abs/1705.05065, 2017.

[36] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[37] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[38] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[39] Alexander Gómez, Augusto Salazar, and Jesús Francisco Vargas-Bonilla. Towards automatic wild animal monitoring: Identification of animal species in camera-trap images using very deep convolutional neural networks. *CoRR*, abs/1603.06169, 2016.

[40] Mohammed Sadegh Norouzzadeh, Anh Nguyen, Margaret Kosmala, Ali Swanson, Meredith Palmer, Craig Packer, and Jeff Clune. Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning, 2017.

[41] Microsoft AI for Earth. Microsoft's camera trap models. https://github.com/microsoft/CameraTraps.

[42] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[43] Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective. *CoRR*, abs/1902.01724, 2019.

[44] DeepMind. Using machine learning to accelerate ecological research. https://deepmind.com/blog/article/using-machine-learning-to-accelerate-ecological-research.

[45] Github. Python is the most used programming language for machine learning. https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning.

[46] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[47] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[48] Google Cloud. Google cloud platform pricing. https://cloud.google.com/pricing.

[49] Thomas Paine, Hailin Jin, Jianchao Yang, Zhe Lin, and Thomas Huang. Gpu asynchronous stochastic gradient descent to speed up neural network training, 12 2013.

[50] Google. Protocol buffers. https://developers.google.com/protocol-buffers/.

[51] Google. Tensorflow models. https://github.com/tensorflow/models.

[52] Google. Colaboratory. https://colab.research.google.com/.

[53] Gitpod. One-click online ide for github. https://www.gitpod.io.

[54] hypraptive. Facenet for bears. https://hypraptive.github.io/2017/01/21/facenet-for-bears.html, 2017.

[55] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.