Automatic Detection of Endangered Species in Aerial Images using Deep Learning

by

Daniel Torres Cirina

Bachelor's Thesis. Specialization in Computing
April 2021

Director: Enrique Romero Merino
Co-Director: Ludwig Houegnigan

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
FIB

Thank you to my family Javier, Maria Antonietta, Daniela and Anshu for your support.
To my directors Enrique Romero and Ludwig Houegnigan for your guidance.
And to my friends McJavi, Daleska, Pablo and Pedro-Rafael for your helping hand.

Contents

List of figures

Figure 55. Number of false positives and false negatives per iteration of the experiment non-cumulative training (Type A) - 10K training steps validated on the dataset eTlZjK4VvmY.
Figure 56. Inference on validation dataset eTlZjK4VvmY with non-cumulative training (Type A) - 10K training steps model.
Figure 57. Number of true positives, false positives and false negatives per iteration of the experiment non-cumulative training (Type A) - 10K training steps validated on the dataset 1dbyKDw27h8.
Figure 58. Inference on validation dataset 1dbyKDw27h8 with non-cumulative training (Type A) - 10K training steps model.
Figure 59. From top to bottom, brightness changes of; -127, -85, +85, +127 inside the bounding boxes.
Figure 60. Number of true positives, false positives and false negatives per experiment for non-cumulative training (Type A) - 10K training steps validated on the brightness variated datasets of 1dbyKDw27h8.

List of tables

1. Context and Scope

1.1. Introduction and contextualization

This is a Bachelor Thesis of the Computer Engineering Degree, specialization in Computing, done at the Facultat d'Informàtica de Barcelona of the Universitat Politècnica de Catalunya directed by Enrique Romero Merino and co-directed by Ludwig Houegnigan.

1.1.1. Context

Close and regular monitoring is a key task to carry out for the conservation of endangered species. Surveys allow us to know if these endangered populations are surviving or vanishing and what could be the best course of action to take to help preserve them.

The species that we will focus on during this project's introduction will be in the genus of Blue, Fin, Grey, Humpback and Right whales. Since 1946, when the International Convention for the Regulation of Whaling was signed by 15 nations, there has been an effort to limit the whale catch and thus try to avoid its extinction. To accomplish this, many surveys have been done but even then there is uncertainty about the results obtained.

The main problem is that the whale population is divided into many species. And at the same time those species have different populations, subpopulations and then subgroups. For example, Right Whales is a genus composed of three different species: North Atlantic Right Whales, North Pacific Right Whales and Southern Right Whales. Each one with its own difference in migration and atmospheric conditions presented in some of its seasonal habitats. This means that the number of surveys increases and that some whale populations that are in the same species are divided into different areas and groups. As has happened with the North Atlantic right whale and the Southern Right whale. Each population was estimated at 490 and 12.000 respectively. But in reality, within that same species, one group is close to extinction and the other one is thriving [1].

RIGHT WHALES

	Year(s) to which estimate applies	'Best' estimate	Approximate 95% CI
Southern Hemisphere Total	2009	12,000	
- Southwest Atlantic	2009	3,300	
		Rate of increase around 7%	
- Southern Africa	2009	3,900	
		Rate of increase around 7%	
- Sub-Antarctic New Zealand	2009	2,700	
- South central and Western Australia	2009	2,000	
		Rate of increase around 7%	
North Atlantic	2010	490	

- Table 1. Estimated number of right whales [1].

Surveys are usually done using aerial footage from planes, drones or even ships. Among the most recent technology to survey whales is satellite imagery thanks to progress sub-meter

resolution and the increase in satellite coverage. Nonetheless, those first few methods have some serious problems. First, it is very costly. Second, the use of satellites generates big volumes of imagery that can't be manually analyzed and needs a significant amount of automation. Furthermore, there still are many holes in the information that is already available due to the migration and the atmospheric conditions presented in some of the whales' seasonal habitat that lead to few and costly surveys. One example of such would be the rank made by the International Union for Conservation of Nature (IUCN) in which it ranked six whale species as "Endangered" and "Vulnerable". And it also ranked twenty-one species as "Data Deficient" [2].

There are many techniques that have been developed for the purpose of detecting whales in satellite imagery like standard methods based on CNNs such as RetinaNet and Convolutional Neural Networks architectures. But we still need to improve the accuracy of it. This way we'll be able to train better models to detect whales in the imagery from drones and transfer that knowledge to get better outcomes on the satellite imagery. And unlike a human driven task, we will also be able to handle much more data, cover more sea space which will lead to having more information at our disposal about the whale populations and their characteristics.

1.1.2. Problem to be resolved

For this project there isn't much of a problem to solve but an opportunity to improve. Improve upon the project that was left which tried to choose the appropriate architecture, build a completely new dataset and figure out what are the best parameters in order to achieve certain goals.

Not having publicly available labelled datasets that contain images related to the field in which one is trying to develop a deep learning model is a big handicap. That's why continuing to collect more data, labelling it, feeding that data to the model to verify the previous findings or make new tests to improve said model is an opportunity to improve a field that has not had the chance to be developed further.

Having a robust machine learning model, multiple experiments and a big dataset can help the main stakeholders of this project. Be it directly by solving a specific problem like improving the detection of whales in drone and satellite imagery or indirectly by leaving a well documented procedure that addresses the main topics of object detection for other people interested in the field to use in the future.

1.1.3. Stakeholders

As common as it may sound, and in a general vision, everybody will use and benefit from the development of this and other similar projects. As we have mentioned before, there are multiple institutions like the IUCN or the International Convention for the Regulation of Whaling which are carrying out this type of research. An improvement of this technology will be used by these kinds of groups to further theirs. Another example would be the The Big Pixel Initiative which is developing geospatial capacity to address the world's greatest challenges at scale. This initiative wants to grow a learning laboratory related to everything

spatial, to investigate and design best practices in geospatial data visualization, user experience interfaces and design techniques for scientific discovery and decision-making [3].

When this technology leads to those improvements then that newly acquired knowledge and techniques will benefit all of us by providing a better global environment.

1.2. Justification

1.2.1. Previous studies

Recommended as a starting point by the director and the co-director to develop this project we will mostly use the software and datasets developed by Rubén Martín Pinardel on his master's thesis submitted to the Faculty of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona [4].

In his thesis' conclusions, Rubén established that among the different network architectures he used (RetinaNet ResNet 50, Faster R-CNN Inception-ResNet V2 and Faster R-CNN ResNet 101) and with a dataset of 4212 images containing 6760 ground-truth boxes, the best performing one was Faster R-CNN Inception-ResNet V2. It obtained an average precision of 0.6570 for aerial images which is an improvement of 6% over the second best model (Faster R-CNN ResNet 101). He also explained how feature extractors obtain the most relevant regions from mages and the effects that synthetic images have during the development of the different architectures.

The results on satellite images are similar to the ones obtained by Emilio Guirado et al. [5]. Although he says that those results are quite satisfactory, it is also mentioned that they can be improved to make a more precise detection. The worst problem is described as the existence of false negatives (the undetected whales) since false positives like boats or rocks can be easily discarded through an image classification detector.

1.2.2. Justification

All the research mentioned above presents methods and compares their performance with the different datasets. However, it also proposes new approaches that can be used to improve upon them. Plus now we can count on the suggestions coming from the same director and co-director on which path to take.

When dealing with the data, it is very important to add more in those areas that the previous study lacks. This addition may directly affect the performance in the training part of the machine learning technique used. With all this information, future researchers could decide which method to use depending on its specific requirements.

1.3. Scope

1.3.1. Objectives and sub-objectives

The main objective is to use the aforementioned thesis as a starting point and improve upon it although it is difficult to know if the performance will actually increase and the success of this depends on multiple parameters. What is most fundamental to the success of the project is to increase the amount of labelled data. There is a huge amount of data available online that has not been fully exploited, therefore we give priority to the following three key tasks:

- The fundamental understanding of detection algorithms and in particular Faster R-CNN.
- Increase the amount of labelled data compared to Rubén Martín (i.e. images with estimated bounding box around animal) to create a very large dataset.
- The evolution of the performance metrics as a function of the number of labelled data in the training set.

Some more in depth remarks on Faster R-CNN would be to use a two-step approach in which:

- We improve the backbone of the CNN. More specifically the feature extractor mentioned in section 4.4.1.
- Improve detection metrics.

Finally, we can mention some of the risks that the project will face. The earliest one is the proper labelling of the images. While looking at the work previously done by Rubén Martín [4] I noticed that he used MATLAB's Image Labeler in order to draw a rectangle around every image that would be used to train the network. This was done by adding a 'whale' label and then separating each image into different folders depending on their genus. The genus was also added as a sub-label. This could either improve or deteriorate the results compared to past ones. This would be a risk of this implementation. Another would be the decision of adding more categories besides the 'whale' one. Such as 'rock' or 'boat'. This extra work for every new picture that will be introduced in the dataset could give no benefit in return. Of course there are more risks present but most of them can be categorized as 'time invested to improve on the result' which are a regular occurrence in the research field.

1.4. Methodology and rigor

1.4.1. Methodology

This project will allow us to get a much deeper theoretical introduction to machine learning. We will focus on explaining the different components of the current available architectures such as CNNs.

We will also have at our disposal the already available dataset of whales. In this case, the dataset is composed of 4212 aerial images and 2168 synthetic images created in order to achieve a better performance when doing "transfer learning" (a method where a model

developed for a task is reused as the starting point for a model on a second task) into satellite images.

Through different software architecture we will go in depth about the usual structures of one and two stage object detectors, providing a detailed description of the feature extractors. These are the three previously optimized selected architectures: Faster R-CNN ResNet 101, Faster R-CNN Inception-ResNet V2 and RetinaNet ResNet 50, performing a 5-fold cross validation model selection of the three architectures with the dataset composed of drone and aerial images. From which we already know that the best architecture for detecting whales on medium and high quality images is the Faster RCNN Inception-ResNet v2.

We have decided to implement the agile methodology, that subdivides the global work into subtasks one/two weeks. Each subtask (longevity to be defined) may be the study and implementation or experimentation of a step required to train or execute the network. So we will set subtasks for each architecture (Faster R-CNNs, RetinaNet, ...), to then analyze the results and make the comparison among them.

1.4.2. Monitoring tools and validation

We will use a Google Drive folder to share every document necessary from previous projects. The FBM server to store and execute the code since it already contains all the modules necessary. The code will be privately available to both director and co-director of the project. Plus, a weekly meeting will be scheduled with the director and co-director of the project for consultations and to establish the tasks to be accomplished the following week.

2. Project Planning

This project lasts approximately 168 hours, distributed in 70 days starting from September 2020 until December 2020. That represents an average of 2.4 hours per day, 5 days per week. The number of hours dedicated to each task need to be prepared in order to ensure a performance throughout the thesis.

2.1. Task definition

In this section we will show the tasks that will be done during the project. Each task is described taking into account the dependencies with other tasks and the duration of each one and are sorted by the starting date.

2.1.1. Project planning

Since a lot of importance seems to have been placed in the final planning, we'll present all the tasks that will be done over the project management.

- Contextualization and project scope. Definition of the project's scope. Indicates the objective of the thesis, the relevance of the area and how it is going to be developed.
- Temporal planning. Planning of the entire execution of the work, providing a description of the phases, resources and requirements needed.
- Economic management and sustainability. Analysis of the sustainability and the economic dimension of the project.
- Meetings. Meetings with the director and co-director will be scheduled every week with the purpose of ensuring that the thesis is evolving correctly and fulfilling the time plan.
- Building the final document. Bring together all the deliveries, correcting the parts that were wrong.

2.1.2. Study of the state of the art

Considering that one of the main purposes of the thesis is to improve upon a previously done thesis, it's fundamental to have a solid theoretical background to understand the experiments already done and to ensure the consistency of new ones. However, before that, it's important to familiarize myself with Convolutional Neural Networks (CNNs). As it is mandatory to study the architectures extensively, we are going to perform the following tasks for each one.

- Describe every architecture that is going to be used.
- Comparison of image classification architectures. More importantly, the accuracy versus the number of operations comparison.
- Comparison of object detection and instance segmentation architectures using the COCO metric.

2.1.3. Practical implementation

In order to compare the architectures, we have to learn how they work and repurpose some of the code in order to develop new experiments.

- Learn Tensorflow. It is necessary to learn this league since the network is implemented using it.
- Learn MATLAB. One of the most important parts of the experiments is having new data to train the models and improve the results. This data has to be labeled using MATLAB's Image Labeler.
- Convert all the data to the required format.
- Retrain the networks.

2.1.4. Experimentation, analysis and conclusion

Providing a good and coherent analysis is an important part of the thesis.The following are necessary for a correct conclusion to the project.

- Select the source data. Given that part of the project is to be able to transfer the knowledge from the drones to the satellites, it is important to gather the proper pictures. This means obtaining and labelling pictures from drones with a proper angle and also from the different species such as right whales or blue whales.
- Selecting the benchmarks. It is important to have a standard metric to compare the architectures in order to avoid bias during the interpretation.
- Experimentation. We have to experiment with each architecture over the data set in order to deepen the analysis of the results and improve upon them.
- Conclusion. Analyze the results obtained and develop a conclusion.

2.1.5. Summary

In the following table 2 we summarize the project planning for every task.

ID	Task	Time (hours)	Dependencies
T1	Project planning	25	
T1.1	Contextualization and project scope	5	
T1.2	Temporal planning	5	
T1.3	Economic management and sustainability	5	
T1.4	Meetings	5	
T1.5	Building the final document	5	
T2	Study of the state of the art	42	
T2.1	Describe architectures	14	
T2.2	Comparison of architectures	14	
T2.3	Comparison of object detection and instance segmentation architectures	14	
T3	Practical implementation	67	T2
T3.1	Learn Tensorflow	16.75	
T3.2	Learn MATLAB	16.75	
T3.3	Convert data	16.75	T3.2
T3.4	Retrain networks	16.75	T2.1, T2.2, T2.3
T4	Experimentation, analysis and conclusion	33	T3
T4.1	Select data source	8.25	
T4.2	Selecting the benchmarks	8.25	
T4.3	Experimentation	8.25	
T4.4	Conclusion	8.25	

- Table 2. Summary of the project planning.

2.2. Resources

Every project needs resources to be able to organize it properly and carry out its development. We have divided them in three different categories: human, hardware, and software.

2.2.1. Human resources

The main sources of information are Enrique Romero Merino and Ludwig Houegnigan. They are the ones who proposed the thesis and they are knowledgeable about the current situation, what the path ahead looks like and the actions required to get there.

2.2.2. Hardware resources

The most essential resource for this project is a good GPU to be able to train the models. I have been given access to a remote server that employs:

- NVIDIA TITAN Xp: 3840 CUDA cores, GPU memory of 12 GB GDDR5X, memory bandwidth up to 547.6 GB/s and 12.15 TFLOPs.
- NVIDIA QUADRO P6000: 3840 CUDA Cores, GPU memory of 24 GB GDDR5X, memory bandwidth up to 432 GB/s and 12.63 TFLOPs.

Plus my personal computer equipment.

2.2.3. Software resources

For now I'll just write and itemised list of everything being used:

- Python and multiple modules (I'll write them all down once the project is complete).
- MATLAB's Image Labeler.

2.3. Risk management: alternative plans

Some risks might appear during the development of the project that could negatively affect its progress. In this section, we will present some potential risks, how they would affect the project and how we could solve them by introducing alternative tasks.

2.3.1. Deadline of the project

This section is mostly affected by the miscalculation in the number of hours dedicated to each task. This is because it is mandatory to schedule the whole project before starting it and this may cause that we divert some time into tasks that would be needed in another and it makes us arrive short handed in the end. It's impact can be between medium to severe. Especially if it happens near the end of the project when there's less time to react accordingly.

The proposed solution would be to adapt the project plan to the new situation and redistribute the hours needed for each task. And if we are still running behind the schedule then a new task would be created. This task would represent the urgency of the situation by adding 5, 10, 15 or 20 hours. As needed.

2.3.2. Inexperience in the field

Given that CNNs are a new topic for me, reaching the level in which I am capable of implementing the architectures or even modifying the already existing ones may be more

time consuming than expected. The impact would be low to medium since we have already experienced situations like these during the classes in the bachelor's degree.

The proposed solution would be to dedicate more hours to study CNNs. Even though there still is the potential risk of not acquiring all the required knowledge for the project. This could be fixed by decreasing the amount of hours dedicated to explaining the most challenging concepts in the thesis. Since it would be better to reflect said knowledge in the conclusions than just explaining them in the introduction.

2.3.3. Coronavirus

Not everybody lives in the same conditions and some situations are tighter than others. In this case, there are no possible solutions to situations such as not being able to work because of the physical condition or because of more important appointments. The impact may vary from low to severe depending on the situation.

3. Budget and Sustainability

The budget and sustainability report is a regular requirement nowadays for most computer related companies' projects. Prestigious organisations such as Global Reporting Initiative (GRI), UN Sustainable development Knowledge Platform or Electronics Watch measure the impact on the sustainability of IT companies' products and services. For example, the organisation GRI defines standards for reports' guidelines on the impact of a project on climate change, human rights, transparency or quality of life, among other aspects.

In that spirit, we will describe the elements to consider when doing the budget estimation, which includes some personal, staff, generic and other related costs to the thesis. Moreover, we will define management control mechanisms in order to explain and control deviations that may appear due to unforeseen setbacks.

3.1. Budget

3.1.1. Personal and staff costs per activity

We define the cost of one task as the sum of the cost of one hour of labor multiplied by the number of hours that it will take to develop said task. If we were in a company we would sum the salary of each employee that contributes to the project. This would involve employees like the members of the GEP course, the director and the co-director of the project. Besides them I would be the only other person working on the project as junior research and developer, tester and analyst.

Role	Budget in € / hour
Manager	22.57
Researcher	12.66
Developer	10.71
Tester	11.33
Analyst	13.28

- Table 3. Budget for the different project roles [6], [7], [8], [9], [10].

Task	Total time (hours)	Manager (hours)	Researcher (hours)	Developer (hours)	Tester (hours)	Analyst (hours)
Project planning	25	25	0	0	0	0
Meetings	70	14	14	14	14	14
Study of the state of the art	42	0	35	7	0	0
Practical implementation	67	0	0	63	4	0
Experimentation, analysis and conclusion	33	15	0	0	0	18

- Table 4. Estimated time per section.

Task	Cost in €
Project planning	564.25
Meetings	987.7
Study of the state of the art	518.07
Practical implementation	720.05
Experimentation, analysis and conclusion	577.59

- Table 5. Estimated cost per section.

3.1.2. Generic costs

The amortizations in this project are mainly hardware because most of the software used is already free to use. The amortization costs are shown below.

Equipment	Price (€)	Time of use or life expectancy
NVIDIA Quadro P6000	2500	6 years
NVIDIA TITAN Xp Graphics Card	1500	6 years
MATLAB Image Labeler	2000	1 year

- Table 6. Estimated hardware cost.

To this we could add:

- Internet cost. The cost varies from carrier to carrier but the market average can be situated at around €30 per month.
- Travel cost. It would be €30.25 for 5 meetings but all the meetings until now have been done over video calls.

3.1.3. Generic cost of the project

Below there is a summarization of the generic cost of the project, not including the amortization of the hardware resources but the full cost of them because of the nature of the experiments.

Section	Cost in €
Hardware	6000 [Table 5]
Internet	120 [3.1.2. Generic costs]
Travel	181.5 [3.1.2. Generic costs]
Payroll	3367.66 [Table 4]
Total	9669.16

- Table 7. Total cost of the project.

3.2. Management control

In every project there are some potential budget deviations. To try to represent them, we will show them below. It will be a resemblance of a procedure to detect any alteration in the project's cost. To do so, while we do each one of the tasks in the planning section, we will calculate the individual deviation from the estimated cost. In order to calculate this deviation we will use the following procedures and terms:

- Estimated cost. Estimated cost of each task as described in sections 3.1.1., 3.1.2. and 3.1.3.
- Real cost. The result of calculating again the costs and possible expenses from complications for each task. Doing so helps identify which part of the task has been miscalculated in order to plan properly for the next project. This may come from parts of the project that require more or less time. Or from necessary hardware of software that needs to be bought to complete the project.
- Deviation. It will be the difference between the estimated cost and the real cost of the task.

Therefore, the deviation factor indicates how much the real cost varies from its estimation. A positive difference by the end of the project would mean that it was done under budget. On the other hand, if it's negative, changes will have to be made in order to identify and correct the mismanaged parts of the project.

3.3. Sustainability

3.3.1. Self-assessment

The poll has revealed that there are many fields to take into account when developing a project. It mentions things like proposing solutions and strategies to promote IT projects consistent with deontological principles related to sustainability and assessing the economic

viability of an IT project and its compatibility with the environmental and social dimensions of sustainability. These kind of topics were known to me although

3.3.2. Economic dimension

PPP: Reflection on the estimated cost for the completion of the project
From my point of view, the budget is unattainable for a student and could not be carried out without the help of the director who put the hardware at my disposal.

Useful Life: How are currently solved economic issues related to the problem that you want to address (state of the art)?
Whales detection is normally carried out from costly sighting surveys, acoustic surveys or through high-resolution images [1]. These methods include arranging planes and ships to rent and carry specific personnel to carry out the task of surveying the whale population.

Useful Life: How will your solution improve economic issues with respect to other existing solutions?
If after the project is finished we are able to detect whales in drone images and transfer that knowledge to detect them in satellite images then there is the potential to save tens of thousand of euros in vehicles, personnel and renting expenses.

3.3.3. Environmental dimension

PPP: Have you estimated the environmental impact of the project?
I have not estimated the environmental impact of the project. The project mainly has a high electricity consumption when training the models instead of material resources. Some of the models take up to 11 hours to train. Some other examples can be seen in Table 7. That's why there will be a high waste of electricity.

Architectures	Training times	Epochs
RetinaNet ResNet 50	1h 34m	19 epochs
RetinaNet ResNet 50	**4h 17m**	**50 epochs**
Faster R-CNN Inception-ResNet v2	**11h 23m**	**65000 steps - 19.3 epochs**
Faster R-CNN ResNet 101	**3h 30m**	**65000 steps - 19.3 epochs**

- Table 8. Training times of three architectures. Table taken from [4].

PPP: Did you plan to minimize its impact, for example, by reusing resources?
Since electricity is the only resource being used excessively the only thing I could do is make a project that is as efficient as possible so that the consumption of electricity decreases.
Either that or I could campaign for the green party so that laws are put in place for the companies responsible to generate said electricity are forced to do it while minimizing the environmental impact.

Useful Life: How is currently solved the problem that you want to address (state of the art)?

Currently ships and planes are being used to be able to do the surveys. These two vehicles are quite known for being one of the biggest polluters methods of transportation. Even more if they are only deployed for a specific task.

3.3.4. Social dimension

PPP: What do you think you will achieve -in terms of personal growth- from doing this project?
I will gain the experience and knowledge that I didn't have the opportunity to achieve through specific courses in the university because there aren't enough resources to be able to enroll all the students that ask for it and instead design a system in which the blame is put on the students. It will also give me the chance to collaborate with experts in the field and get first hand knowledge from them.

How will your solution improve the quality of life with respect to other existing solutions?
Better surveys means that we would be able to take better care of it which would make the Earth a better place for all.

Useful Life: Is there a real need for the project?
Yes. As we stated in the introduction the survey and control of the population of this species is critical to the survival of the current environment.

4. Theoretical background

Machine learning is used in order to construct an unknown function that takes some reference data to make future predictions or decisions, in such a way that it is more efficient and they are also not explicitly programmed to do so [11]. The unknown function transforms an input ***I*** into an output ***O***. This input and output can take many forms depending on the type of application:

Type of application	Input ***I***	Output ***O***
Spam filtering	An email	{spam, non-spam}
Face recognition	An image	Identified faces
Machine translation	A sentence in language A	A sentence in language B
Speech recognition	A speech signal	A (text) sentence
Data mining	A financial transaction	{fraud, non-fraud}
Robot motion	Sensory data	Motor control

- Table 9. Types of applications and their respective inputs and outputs [12].

The algorithm will use a subset ***T***, called training data, to obtain the desired function. And the output can be structured in different ways such as classification responses, bounding boxes, data structures such as clusters, etc. Thus, machine learning algorithms try to create an optimal function $\boldsymbol{F_T} : \boldsymbol{I} \rightarrow \boldsymbol{O}$.

Based on the desired outcome, machine learning algorithms are organized into taxonomy. Common types include:

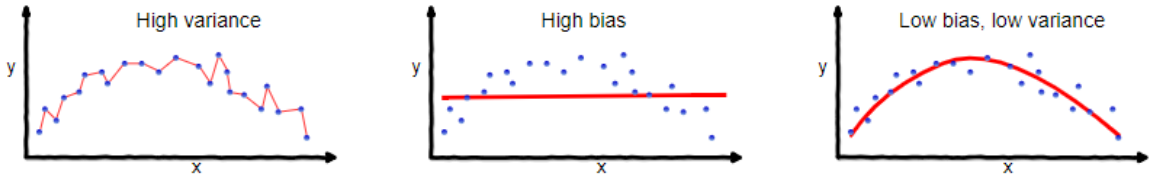
- Supervised learning. The algorithm generates a function that maps inputs to desired outputs. The classification problem belongs in this category. By looking at examples of correct pairs of input and output the function will predict one of several classes. The one that approximates as much as possible.
- Unsupervised learning. It models a set of inputs without labeled examples. That is to get a possible underlying structure of the input data.
- Semi-supervised learning. It combines labeled and unlabeled data to generate the function.
- Reinforcement learning. Learns a policy of how to act given the input data via a cumulative reward that has to be maximized by making the correct actions.
- Transduction. Unlike the supervised learning, it doesn't explicitly construct a function. Instead, it tries to predict new outputs based on training inputs, training outputs and new inputs.

Supervised learning has been used throughout the project since the task in hand is to do object detection. The cost function is the main feature of the algorithm. This is because it calculates the error between the predicted outcome and the actual outcome. With an optimization algorithm that makes use of these error values it will be possible to make the necessary changes to the model so that it minimizes the cost function. In order to be able to extract the main features of the training data it is important to use the appropriate cost

functions. It's what will allow us to generalize an answer. Cross-entropy, mean squared error (MSE) and mean absolute error (MAE) are some of the main types of functions to use when training models.

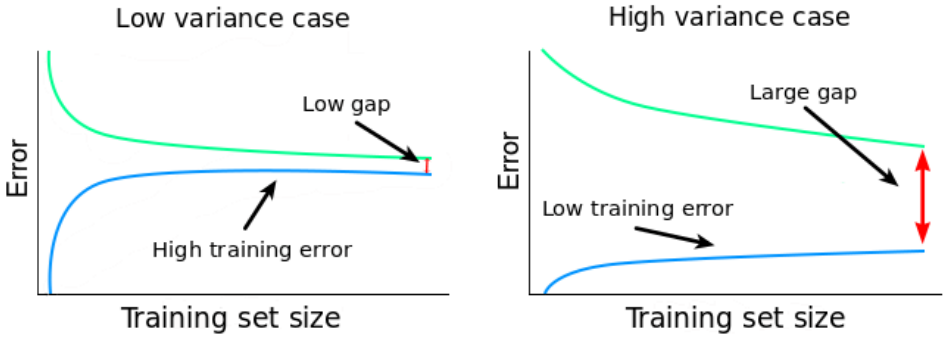
The difficulty resides in finding one that is able to handle the most difficult aspects; random noise that may be present, mislabeled examples, irrelevant or missing values and errors from the training data that biases the results. In any case these circumstances make for an imperfect function.

Trying to find the correct balance between filtering the unnecessary information and letting too much of it out is a problem called bias–variance tradeoff. This property of the model is a tradeoff between underfitting and overfitting it. Underfitting produces high bias and overfitting high variance. High bias can cause an algorithm to miss the relevant relations between features and target outputs. While high variance can cause an algorithm to model the random noise in the training data.



- Figure 1. (Left) High variance and overfitting of the data. (Center) High bias and underfitting of the data. (Right) Low bias, low variance and a correct fitting of the data.

A validation set is used to avoid choosing the incorrect complexity of the model. This validation set is an input subset which will allow us to test the trained model. If the learning curves are plotted against the size of the training set we can observe how different phenomena occur.

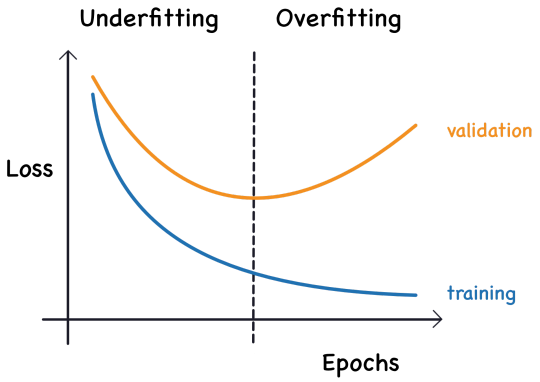


- Figure 2. Two possible training curves evolving alongside the size of the training set and a display of the gap between the errors.

As more training and validation examples are introduced, the training error increases and the validation error decreases. In the case of the training error it's because as more data is introduced, the more the model has to adapt and fit to it. Which loosens the fittingness of the previous data. Conversely, for the validation the model learns more features and is able to tighten the fitting. Having high values for the training and validation errors and a small gap

between them means that the model has low complexity. Which translates to having high bias. In this case adding more training examples won't decrease the validation error. In the case of low value for the training error and a large gap between the errors, which can be seen in the right of the figure, means that there's high complexity and, therefore, high variance. By adding more training examples it could decrease the validation error.

Another aspect of which the complexity depends is the amount of training. A model with enough complexity that it's trained too much will incur into overfitting. And oppositely, if it's trained too little it will produce an underfitted model. Displaying the training loss against the number of epochs helps identify where is the appropriate point to stop the training.



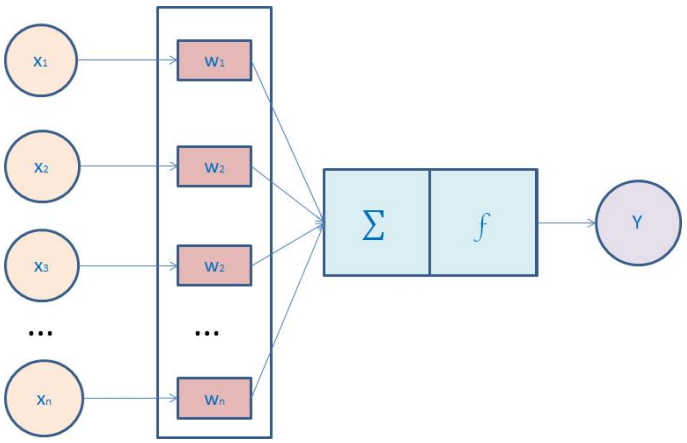
- Figure 3.Training and validation loss displayed for every instance in which the model has been trained with the complete training set one time.

Nonetheless, with a fixed training and validation set the training error shall fall for every epoch. As it's seen from figure 3 the validation error decreases when the model is learning from the training data and then increases when it reaches the point of learning irrelevant features. The dotted lined pinpoints the optimal region which is the minimum of the validation cost function. By generating a model with a good enough complexity for the size of the training set and training it for the appropriate amount of time it is possible to correctly solve the bias-variance tradeoff problem.

4.1. Neural Network

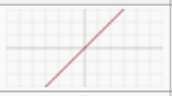
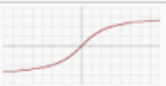
A neural network (NN) consists of many neurons, each producing a sequence of activations which output a real value. Input neurons located in the input layer get activated through an input, other neurons like the ones found in the hidden layers get activated through weighted connections from previously active neurons and output neurons in the output layer produce the output variables.

These inputs and weights are received by the aggregation function which combines them. The way that it's combined varies depending on the type of the network's architecture. But generally a bias is added to the output of the aggregation function. Finally, in order to obtain an output, it is necessary to apply an activation function which is a linear or nonlinear function that transforms the aggregation function.



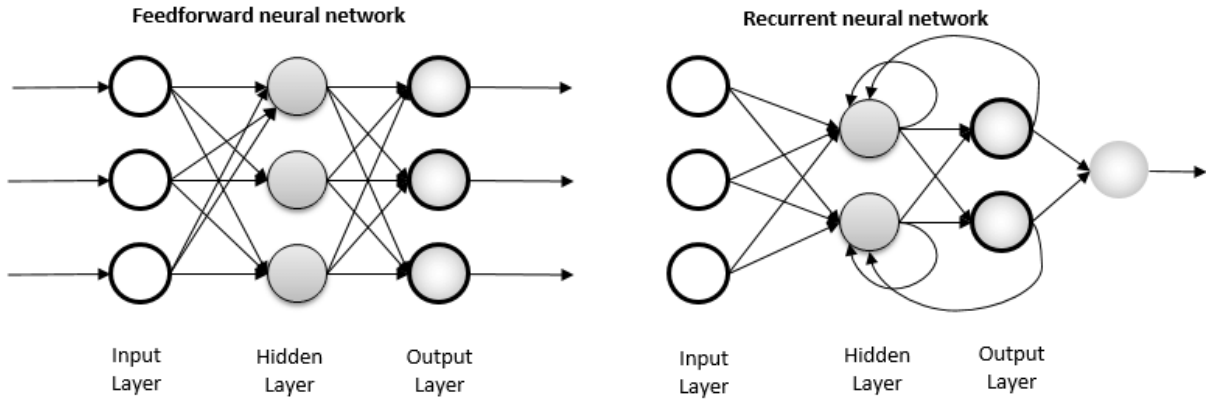
- Figure 4. Basic components of a NN. The set $\{x_1,...,x_n\}$ represents the input, $\{w_1,...,w_n\}$ the weights, $\sum$ the aggregation function, f the activation function and Y the output.

The most used types of activation functions are nonlinear because they allow complex transformations of the inputs. Using only linear functions would make the whole network just a linear combination of bias and weights. No matter how many hidden layers, the last layer would just be a linear function of the first layer. Instead, nonlinear functions allow stacking of multiple layers which translate to being able to perform backpropagation [13].

Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU)		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$

- Table 10. Some common nonlinear functions used as activation functions, their plots and equations [14].

Beside the input layer and output layer there are many others connected with the whole network in a specific way. These are called hidden layers and they perform transformations to the input layer and proceed to make an output for the next layer. The specific way in which the three types of layers are connected creates a structure. Stacking many hidden layers between the input and output layers creates a deep neural network (DNN). Deep Learning is about accurately assigning credit across many such stages in order to improve the output [15].



- Figure 5. Structure of a feedforward neural network (left) and a recurrent neural network (right).

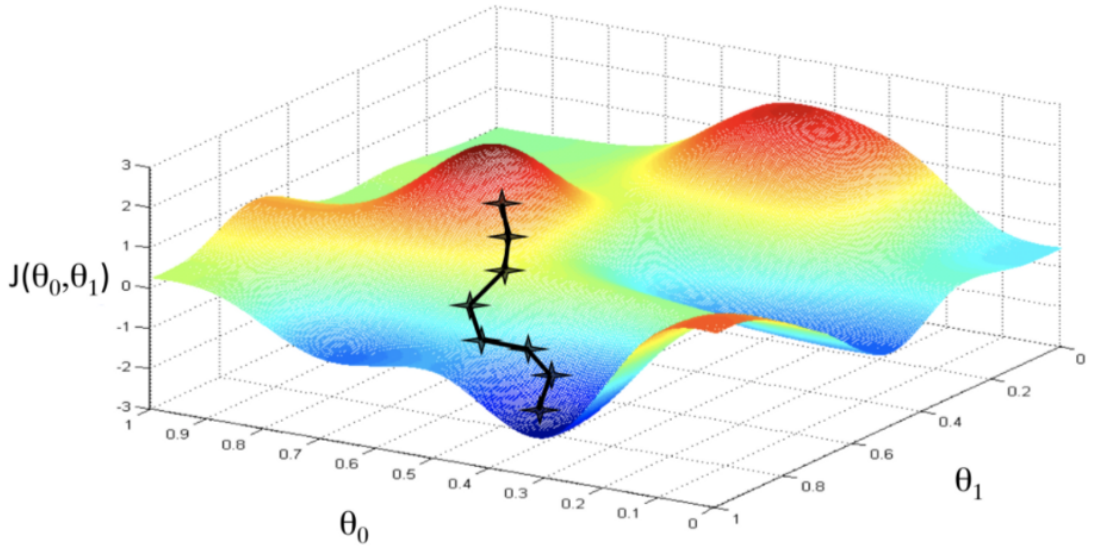
Two of the most used connection structures are Feedforward neural networks (FNNs) and Recurrent neural networks (RNNs) [16]. In FNNs there's only one direction of propagation. That means that the layers only received input connections from its immediately preceding layer and their output connections go to the immediate following layer. Also, the input and the output layer are fixed. While in RNNs recurrent connections that create cycles are allowed. These cycles can be between the same layer or with previous layers. And the input and output layers usually aren't determined. Possibly making the neurons be one or several of the types: input, hidden and output.

Learning is about finding weights that make the neural network exhibit desired behavior. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages, where each stage transforms the aggregate activation of the network.

4.1.1. Optimization Techniques

Training a neural network to approximate a mapping function to a certain result can be treated as an optimization problem of a cost function. The optimization usually is done to obtain a minimum.

Cost functions calculate the difference between the predicted result and the expected one. These calculations depend on the input data, the network's structure and the parameters given. As it has been said before, there are many types of cost functions depending on the problem (cross-entropy, mean squared error (MSE), mean absolute error (MAE)).



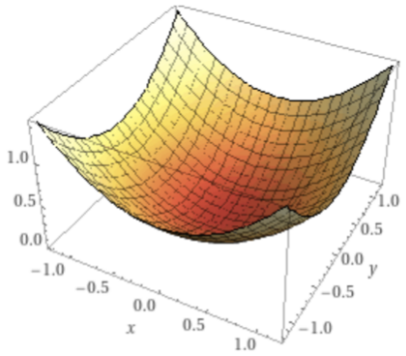
- Figure 6. Path between a random point coordinate and the global minimum displayed on top of a hyperspace.

Based on the results of the cost function the parameters of the network are adjusted. This can be done via many different kinds of algorithms. Gradient descent back propagation (BP) with the appropriate neural network topology is one of the most efficient and commonly used optimization techniques [17].

4.1.1.1. Gradient

A gradient is a vector that is tangent to a function. It points in the direction of greatest increase of said function. The gradient has a value of zero when it's at a local maximum or minimum because there isn't a direction to which it will increase. For example, for the function:

$$f(x,y) = \frac{1}{2}(x^2 + y^2) \qquad \nabla f(x,y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = (x,y)$$



- Figure 7. (Top left) Scalar-valued differentiable function f of several variables, (top right) vector field ∇f and (down) plot of f.

From the plot we can see that the minimum is located at the coordinates (0, 0). The x component of the gradient is the partial derivative with respect to x and the y component the partial derivative with respect to y. A formal definition would be that the gradient of the

scalar-valued differentiable function f of several variables is the vector field ∇f whose value at a point *p* is the vector whose components are the partial derivatives of f at *p*.

4.1.1.2. Gradient Descent and Backpropagation

A negative gradient is a vector pointing at the greatest decrease of a function. Therefore, we can minimize a function by iteratively updating the network's parameters. This is done by subtracting the gradient of the cost function. That means moving in the direction of a negative gradient. Each parameter is updated with its derivative of the cost function [18]. To control the rate of change (ROC), a constant called learning rate (λ) is added.

$$(\theta_1^{t+1}, \theta_2^{t+1}, \dots, \theta_M^{t+1}) = (\theta_1^t, \theta_2^t, \dots, \theta_M^t) - \lambda \left(\frac{\partial J(D, \boldsymbol{\theta}_t)}{\partial \theta_1^t}, \frac{\partial J(D, \boldsymbol{\theta}_t)}{\partial \theta_2^t}, \dots, \frac{\partial J(D, \boldsymbol{\theta}_t)}{\partial \theta_M^t} \right)$$

- Figure 8. Gradient Descent. Left of the equation is the network's parameters $\boldsymbol{\theta}_t$. To the right λ is the learning rate, *J(D, $\boldsymbol{\theta}_t$)* is the cost function for a dataset D and *t* indicates the discrete time steps. Figure taken from [4].

But this method is not efficient since it requires evaluating *M* data examples and the gradient might end up at a local minimum instead of the global one. That's why it is standard procedure to introduce randomness such as the initialization of the starting point. This will increase the odds of avoiding a local minimum.

Another technique to improve the efficiency is to use stochastic gradient descent (SGD). This approach consists in changing the parameters of the network more frequently which allows more intensive search in the hypersurface that represents the cost function (like the one seen in figure 7). For this, a subset of *D* will be used to compute the cost function. This subset is called mini-batch (*mB*) and it can have different sizes. Once a mini-batch size is selected the network's parameters will be updated *(size(D)/size(mB))* times instead of per epoch. The randomness of calculating the gradient on these subsets will contribute positively in the search of the global minimum [4].

Backpropagation is an algorithm for calculating the gradient of a loss function with respect to variables of a model [19]. When training neural network models, backpropagation is used to calculate the gradient for each weight in the network model. Later, the gradient is used by an optimization algorithm to update the model's weights.

The algorithm allows the information from the cost to then flow backwards through the network, in order to compute the gradient. This means that it calculates the gradients of variables in graph structures working, backward from the output of the graph toward the input of the graph, propagating the error in the predicted output that is used to calculate the gradient for each variable [20]. Stochastic gradient descent (SGD) has become an important optimization method in machine learning.

4.1.1.3. Momentum

Momentum is a method applied in stochastic gradient descent so that it takes into account the previous gradient updates and uses it to determine the next iteration gradient as a linear combination between them [21].

$$\Delta w := \alpha \Delta w - \eta \nabla Q_i(w) \quad (1)$$

$$w := w + \Delta w \quad (2)$$

- Figure 9. Momentum. In (1) w is the parameter that minimizes $Q(w)$, α is an exponential decay factor between 0 and 1 that determines the relative contribution of the current gradient and earlier gradients to the weight change and η is the learning rate [22] [23]. In (2) the weight update.

Because of this contribution by the previous gradients, using momentum helps decrease oscillations during the execution of stochastic gradient descent.

4.1.2. Vanishing gradient problem

But the optimization techniques don't always perform correctly. On some neural networks they perform unsatisfactorily because the gradients attain near zero values in some layers. This multiplication with such a small value causes the gradients to vanish. More specifically, this problem is encountered when training artificial neural networks with gradient-based learning methods and backpropagation and it's called the vanishing gradient problem. For those networks that do present this problem can be fixed by not training the first layers.

There are a variety of ways to avoid said problem such as; using an activation function like ReLu (seen in figure 5), using functions with high values for their derivatives, using Jürgen Schmidhuber's multi-level hierarchy of networks [24] or making connections between layers that are not immediately close allowing local gradients to connect even though their layers are situated far apart.

But on the flip side, if an activation function's derivative takes a very large value then it may enquire on the exploding gradient problem.

4.2. Deep learning techniques

4.2.1. Transfer Learning

Building a completely new dataset for a specific task has a very high cost in time and resources. It requires finding appropriate sources, collecting the data and manually labelling it. The training datasets are necessary to be able to fit the model. But using very large datasets also requires a large amount of time to train.

That's why the transfer learning method is used. It repurposes the knowledge acquired by a previous machine learning model to apply it to our task. This means using the final weights from the previous model as the initial weights for ours instead of randomly initializing them.

4.2.2. Data Augmentation

Data Augmentation consists of synthesizing new data by applying specific transformations to the available data. It is very useful when having small datasets and difficulties to expand

them. Its effectiveness and ability to increase the performance of deep neural networks have been proved [25].

- Figure 10. Some common data augmentation transformations.

The data augmentation transformations used on our project dataset are the ones that came by default from the last configuration:

- Horizontal flip.
- Vertical flip.
- Brightness changes.
- Saturation changes.
- Contrast changes.

Online augmentation is used instead of applying each transformation to every single image. It randomly performs the transformations only in the mini-batch feed to the model and avoids large training times. The specific configuration can be seen later in section 6.4.2.

4.3. Convolutional Neural Networks

Convolutional neural networks use convolution which is mathematically described as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx$$

- Figure 11. Mathematical description of convolution.

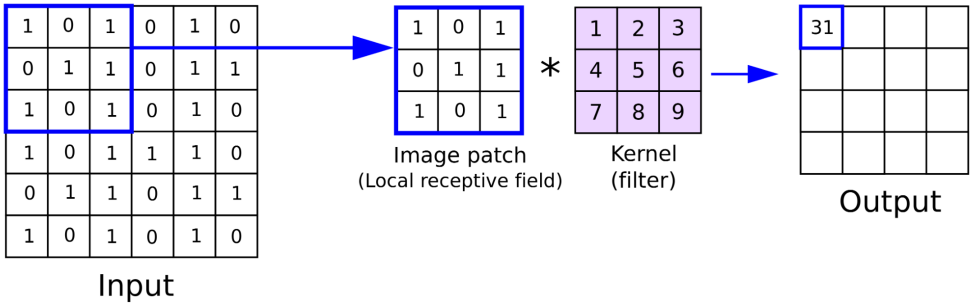
It's the composition of the two functions *f* and *g* at every space position *t*. With this formula it is possible to apply to an image (*f*) a function that represents a spatial feature (*g*) and see how much of that spatial feature is present in the image [4].

We have a discrete space composed of pixels for image detection. Every image has a width and height and is composed of three channels (RGB) with ranging values from zero to 255 so the convolution has to be discretized:

$$C_{K_f}(n,m) = \sum_{i=1}^{A}\sum_{j=1}^{B}\sum_{k=1}^{3} I(n+i, m+j, k)\, K_f(i,j,k)$$

- Figure 12. Convolution to be discretized.

In figure 12 I is an image, n and m are its dimension, 3 is because of the three channels, K_f is a filter of dimensions A, B and 3 and the result is stored in the feature map at position (*n*,*m*). When a filter K_f is convoluted with the input layer of size n × m × k to generate a feature map to determine the features to be extracted is called a convolutional layer inside a convolutional neural network.



- Figure 13. Example of convolutional layer where the filter is convoluted through the image patch to produce the output.

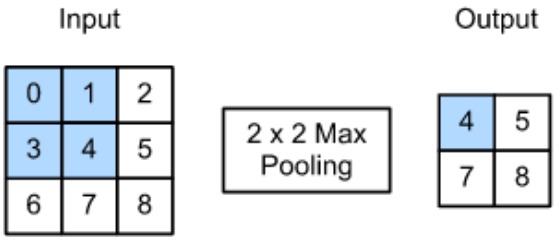
This kernel is applied to every region inside the image and the region depends on the size of the kernel window and the stride. The stride determines the number of pixels the window has to move per convolution. Some techniques such as padding may have to be applied in order to avoid problems like losing pixels on the perimeter of the image [26].

For every convolution more than one kernel filter can be applied, each one generating its feature map. The next kernel must have the same number of channels as the number of feature maps that it receives. This way, by varying the number of feature maps to be outputted, is possible to make every layer learn different features and at the same time learn more features in the whole network.

4.3.1. Pooling Layers

A pooling layer helps extract more common features by doing pooling operations and is usually applied after nonlinearity layers. Just like the kernel from the convolutional operations, the pooling operation also has a size and stride. The pooling operation will divide the pixels in subsections and take only one value from every every section to generate a

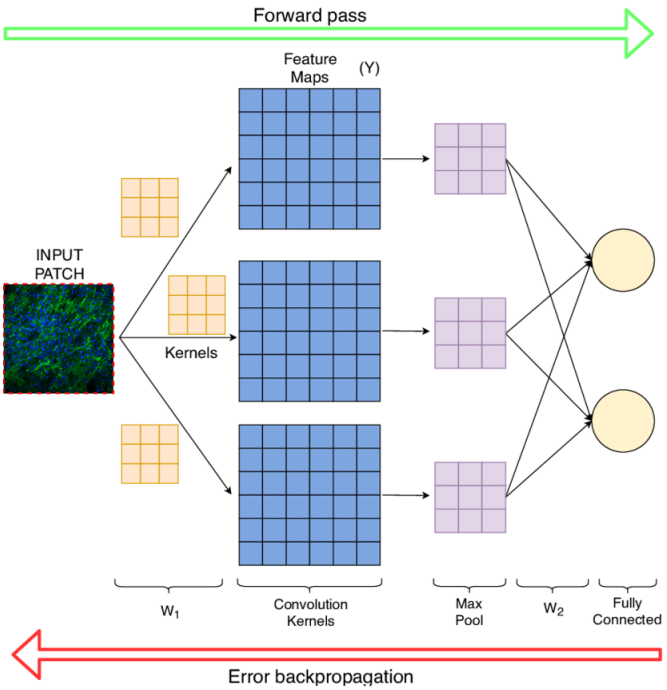
new smaller grid. The value to be taken depends on the function applied. It can be maximum or average pooling where the maximum and average value of the subsection is extracted.



- Figure 14. Output from applying maximum pooling to a subsection of size 2x2 of the original input.

4.3.2. Backpropagation in convolutional neural networks

As we have seen until now, the image is decomposed into pixels which are given as input to the input layer. In the layers one or more operations are made to extract one or more feature maps and these operations depend on the weights in the network. The convolutional operations act as aggregation functions and the operations after the convolution such as the ones seen in table 10 are the activation functions.



- Figure 15. Backpropagation process divided into the forwards pass and the error backpropagation [27].

The backpropagation algorithm will update work to update the weights that affect these filters. Except for the pooling layers aren't trained as they don't have any parameters. Nevertheless, it is still necessary to backpropagate the gradient through all of them.

4.4. Object Detection

Object detection consists of localizing objects inside an image and classifying them. Nowadays the most common architectures for object detection are convolutional neural networks. These convolutional neural networks can be categorized in one stage or two stage detectors. The localization of the objects in the two stage detectors is done by a region proposal network (RPN) which generates regions of interest, where there might be an object, for the second stage to classify said regions and optimize the predicted position of the bounding box [28]. Unlike the two stage detectors, the one stage detectors don't have a region proposal network. They directly generate bounding boxes and predicted classes from the images with a degree of probability or have fixed regions of interest that slightly change with the regression (like AlexNet and VGGNet). That's why one stage detectors are usually less precise but require less inference time. Although there are some architectures, such as YOLO, which perform better than two stage detectors like Fast R-CNN and also architectures like Faster R-CNN that are more efficient in inference time than a one stage detector like YOLO [29]. But everything ends up depending on the task at hand.

Either way, all these architectures have in common that they use a feature extractor. A feature extractor is a convolutional neural network that generates the feature maps by extracting the important features from images so that object detection and classification can be done correctly.

4.4.1. Feature Extractors

This section only takes into consideration the convolutional neural network section of the most used and accurate feature extractors that have been used in the object detection architecture of the project.

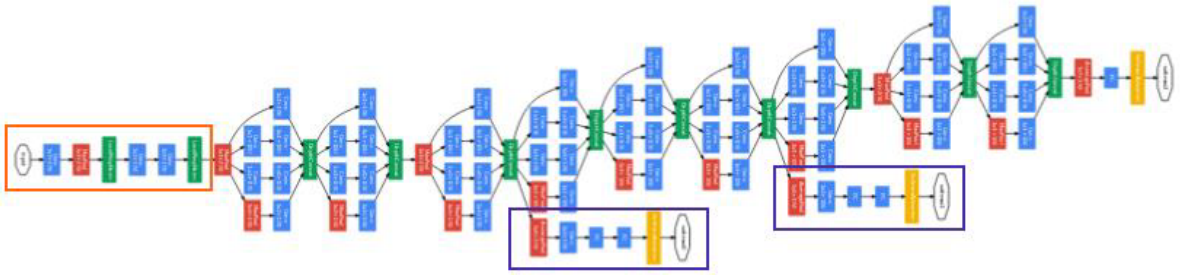
4.4.1.1. Inception

As mentioned in section 4.3. the size of the filter decides the subsections of the image from which the features will be extracted in an iterative way. But the optimal size of the kernel varies depending on the objects of the images.

- Figure 16. Two images of our dataset with objects to be detected of very different sizes.

Inception uses a variety of kernel sizes at each level to gather information at different object scales making the network wider instead of deeper. The Inception V1 network is mainly composed of one module which is stacked at different depth levels [30]. The outputs of each

layer are concatenated at the end of the module. Then the whole output is given as an input to the following layer or modul. All the convolutional layers inside and outside the inception modules use ReLU as activation function.

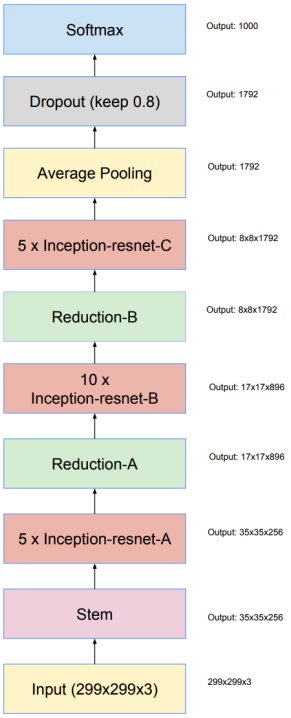


- Figure 17. The Inception V1 network [30].

Inception V2 is an improvement upon V1 that gives a better performance both in accuracy and time. An example of that improvement would be the change from one 5x5 convolutional layer to two 3x3 convolutional layers.

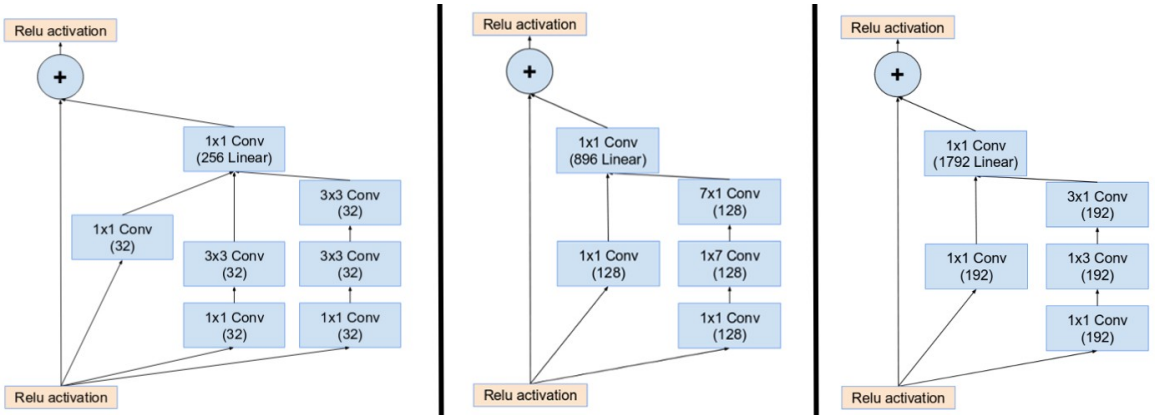
4.4.1.2. Inception-ResNet

Inception-ResNet is an architecture that incorporates residual connections on Inception modules by replacing the filter concatenation stage of the Inception architecture. Szegedy et al [31] propose two different variations of the same architecture: Inception-ResNet V1 and V2. While using the same modules, these two variations have changes like the number of filters applied and some other hyperparameters.



- Figure 18. Scheme for Inception-ResNet V1 and V2.

The scheme from figure 18 applies to both networks but the underlying components differ. The stem refers to the initial operations performed before introducing the Inception blocks (A, B and C). Reduction blocks are used to change the width and height of the grid which earlier versions didn't explicitly have.



- Figure 19. Inception modules A,B and C in an Inception-ResNet.

In figure 19 the pooling layer was replaced by the residual connection, and also the additional 1x1 convolution before addition [32].

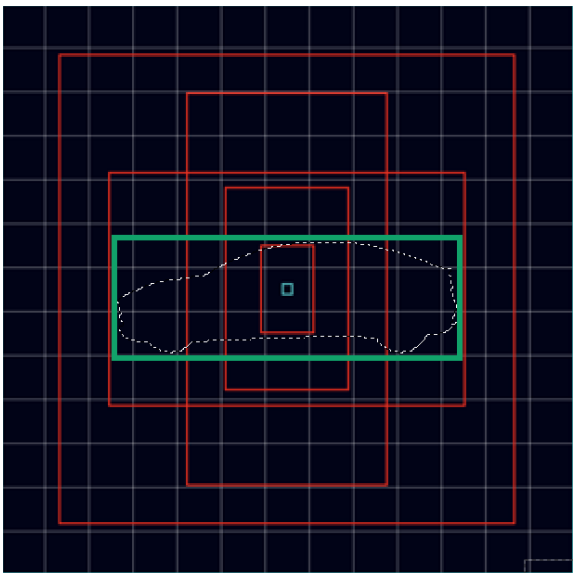
4.4.2. Faster R-CNN

Faster R-CNN is a two-stage object detector advanced from the R-CNN and Fast R-CNN detectors with the difference among them being the region proposal network [33]. The last two run a selective search algorithm on every image and generate around 2000 proposed regions for each one. Those 2000 proposed regions are used on the input images by R-CNN and then it gives them to the feature extractor. In contrast, Fast R-CNN hands the image to the feature extractor, obtains the extractor's feature map and global stride and finally applies those regions from the feature map using only once the feature extractor.

But both options have a very high cost in time which is why Shaoquing Ren et al. [33] came up with the idea for Faster R-CNN. In their design they replace the selective search algorithm with a network that will learn the region proposals and then give it to the Fast R-CNN detector. Besides the feature extractor, Faster R-CNN's structure is divided into two more sections: its region proposal network and the detector.

4.4.2.1. Region Proposal Network

Anchors are a fixed number of bounding boxes with predefined dimensions specifically situated on an image. They are used so it is not necessary to generate bounding boxes and instead they can be adjusted to better fit the objects. The feature extractor produces the feature map and pixels of the feature map define the centers of the anchors.



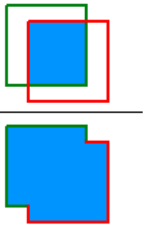
- Figure 20. Anchors situated to detect a car with the green bounding box being the best fitting for the object to be detected.

Then a certain number of anchors are taken by the region proposal network and it returns the probability that those anchors contain an object and changes to be made to them so that each anchor that is not discarded because it contains some object fits it better.

As for the structure of the region proposal network, it is fully convolutional without employing fully connected layers at the output. It also has to be trained to correctly predict the anchors' offsets and the probabilities of containing an object. Which means having to use a cost function to determine the correctness of the prediction. Since this features inside the supervised learning category, we will give it the correct answers when training. The correct answer will be given in the form of a ground-truth box manually generated. It contains the class and the coordinates of the answer.

Intersection over union evaluates the overlap between ground-truth box (gt) and the predicted box (pd) [34].

$$\text{IoU} = \frac{\text{area}(gt \cap pd)}{\text{area}(gt \cup pd)} \qquad IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

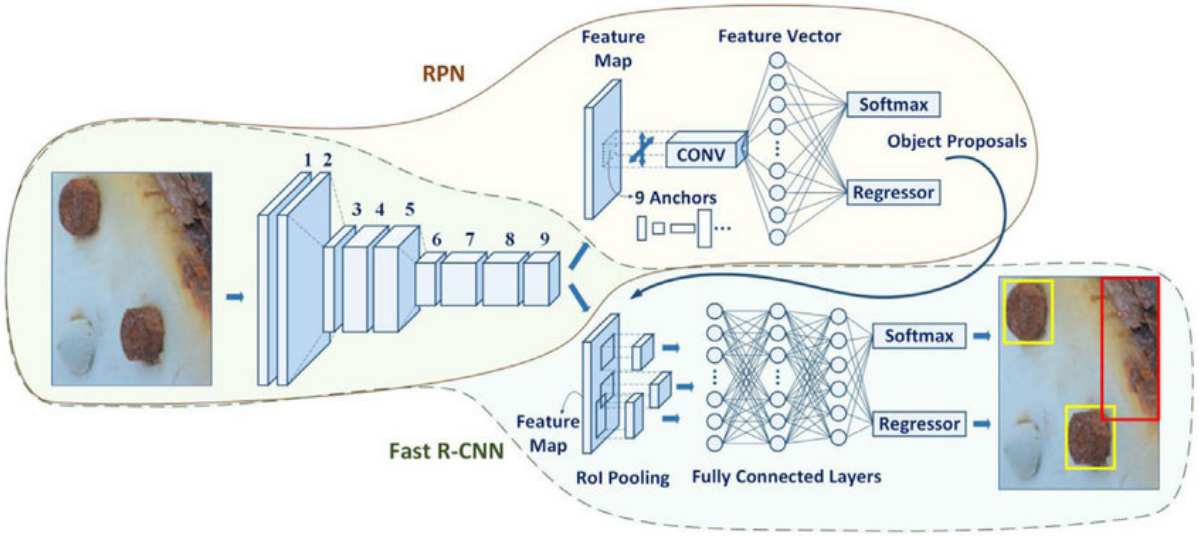


- Figure 21. Intersection over union metric.

Only the anchors that have an IoU over the established threshold are accepted. If a large enough number of anchors pass the threshold the training mini-batch tries to select an equal number of accepted and discarded anchors to not introduce bias. Finally, the anchors are used by the region proposal network cost function and the trained network will output the regressed anchors as predicted regions of interest. Although not perfect as more than one region may be for the same object. That's why non-maximum suppression is applied which merges windows that might belong to the same object [35].

4.4.2.2. Fast R-CNN Detector

The Fast R-CNN detector also has a region proposal network which shares its convolutional layers. Via bounding box regression it will classify the objects by classes using the region proposal network's output.



- Figure 22. Faster R-CNN schematic structure [36].

As it can be seen in figure 22 the region of interest (RoI) pooling layer then takes different regions out of the feature map to apply max pooling to them and given as an input to the fully connected layers. For this project the detector was already trained by Rubén Martín the same way as the region proposal network in section 4.4.2.1. Without taking into consideration the classes and using the same cost function. But it could also be trained differently.

4.5. Metrics

To evaluate the results of the architecture we will use the MS COCO metrics [37]. Of the metrics that it gives the most important ones are the precision (P) and the recall (R). They can be computed manually using the ground-truth boxes and the predicted bounding boxes.

First of all we have to categorize the predicted output:

- True Positive (TP) is a valid detection.
- False Positive (FP) is an invalid detection.
- False Negative (FN) is a ground-truth missed by the model.
- True Negative (TN) this metric shouldn't appear because there are infinitely many instances of non-ground-truth boxes that should not be detected as objects.

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{TP}{\text{all detections}}$$
$$R = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{TP}{\text{all ground-truths}}$$

- Figure 23. Calculation of precision and recall.

We will use the previous categorization for the confusion matrices and the following MS COCO metrics [38] to evaluate the architectures:

- Average precision (AP or mAP):
    - AP is based on the precision-recall curve, is the interpolated precision averaged across all unique recall levels if the IoU is between 0.5 and 0.9. The interpolated precision p_{interp} at a certain recall level r is defined as the highest precision found for any recall level $r' \geq r$:

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

- Figure 24. Interpolation of precision.

    - AP @ IoU = 0.50. Which is the AP for bounding boxes that have predictions of 0.50 or higher.
    - AP @ IoU = 0.75. Which is the AP for bounding boxes that have predictions of 0.75 or higher.
    - AP @ area = small. Which is the AP for small objects with an area smaller than 32^2 pixels.
    - AP @ area = medium. Which is the AP for medium objects with an area bigger than 32^2 and smaller than 96^2 pixels.
    - AP @ area = large. Which is the AP for large objects with an area larger than 96^2 pixels.
- Average recall (AR or mAR):
    - AR is the recall averaged over all IoU in the range between 0.50 and 1.0. It can be computed as two times the area under the recall-IoU curve.

$$AR = 2 \int_{0.5}^{1} recall(o)$$

- Figure 25. Averaged recall where *o* is IoU and *recall(o)* is the corresponding recall.

    - AR @ maxDets = 1. Which is AR given 1 detection per image.
    - AR @ maxDets = 10. Which is AR given 10 detection per image.
    - AR @ maxDets = 100. Which is AR given 100 detection per image.

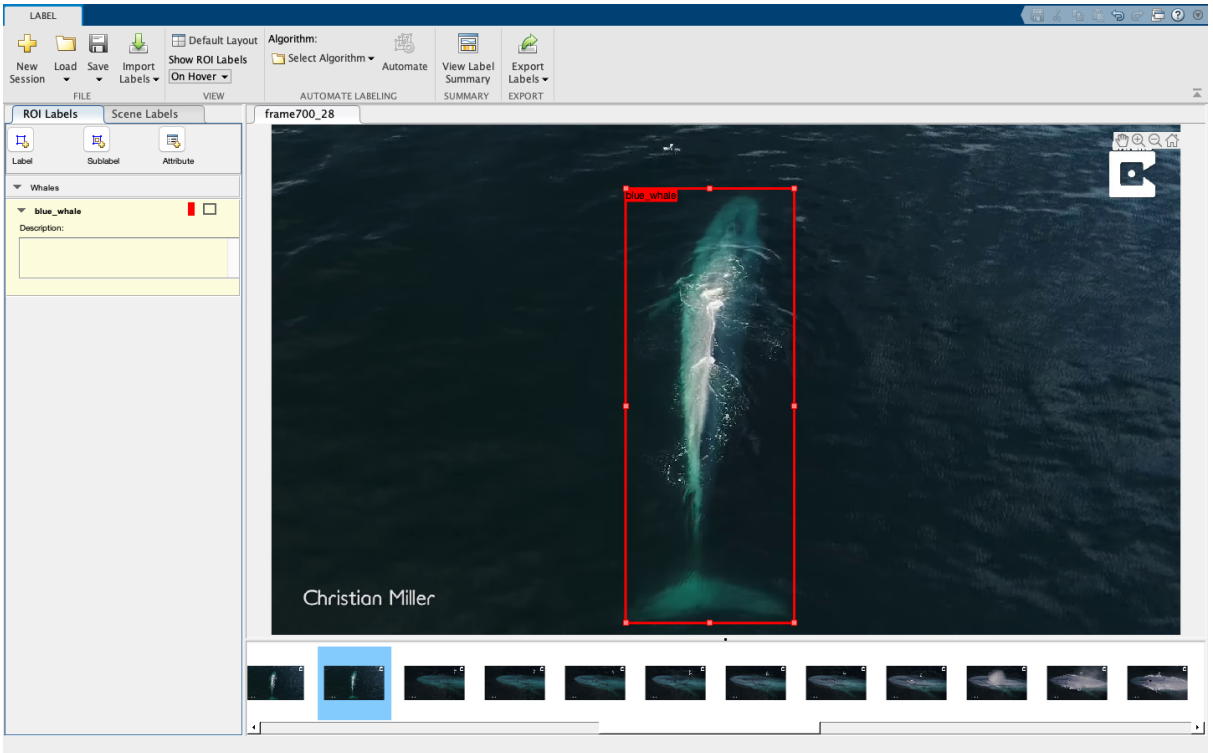
Plus more combinations with area and maxDets for both AP and AR.

5. Building a new dataset

A dataset is composed of images and their added bounding box annotations that define the objects' class. There must be one or more objects present in the image and the bounding box indicates where the ground-truth lies within the image.

The main difficulty for building the new high quality datasets is finding images that haven't been used before. It's needed to build new datasets since there are none publicly available of these characteristics. By 'these characteristics' we mean aerial whale pictures taken at a 90 degree angle. Which is quite scarce to come across considering the good job done by Rubén at scavenging the internet for said images for his master's thesis.

In order to build it, I have used YouTube and Vimeo videos, mostly from this year to be able to avoid overlaps with the previous datasets. These videos are of different quality and have been recorded from drones at different heights so there is a variety of resolutions and scales. From each video one frame has been extracted for every second. Of this set of images a subset of 50 is selected which will be used for either training or validation. All images have been manually labelled using MATLAB's Image Labeler.



- Figure 26. MATLAB's Image Labeler session.

In the labelling, the label's name reflects the object's class and a sublabel has been added which specifies the species of the whale. Although the species is important in the development of the project, the sublabel doesn't affect the estimation of the algorithm.

The datasets are composed of 1500 images from 30 videos with 2183 ground-truth boxes:

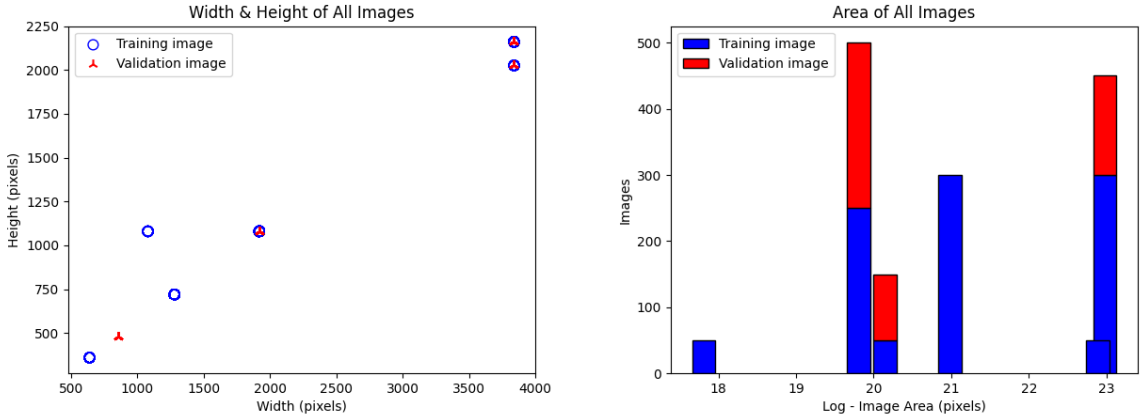
Whale species	Dataset	Number of ground-truth boxes
Blue whale	Training	158
Blue whale	Validation	135
Fin whale	Training	309
Fin whale	Validation	192
Grey whale	Training	368
Grey whale	Validation	130
Humpback whale	Training	253
Humpback whale	Validation	117
Right whale	Training	350
Right whale	Validation	171

- Table 11. Composition of the training and validation datasets.

The difference between the number of images and the number of bounding boxes is due to the presence of multiple whales inside the same frame. We have videos where two, three or four whales are present in several frames. It's not exclusively one ground-truth box per frame.

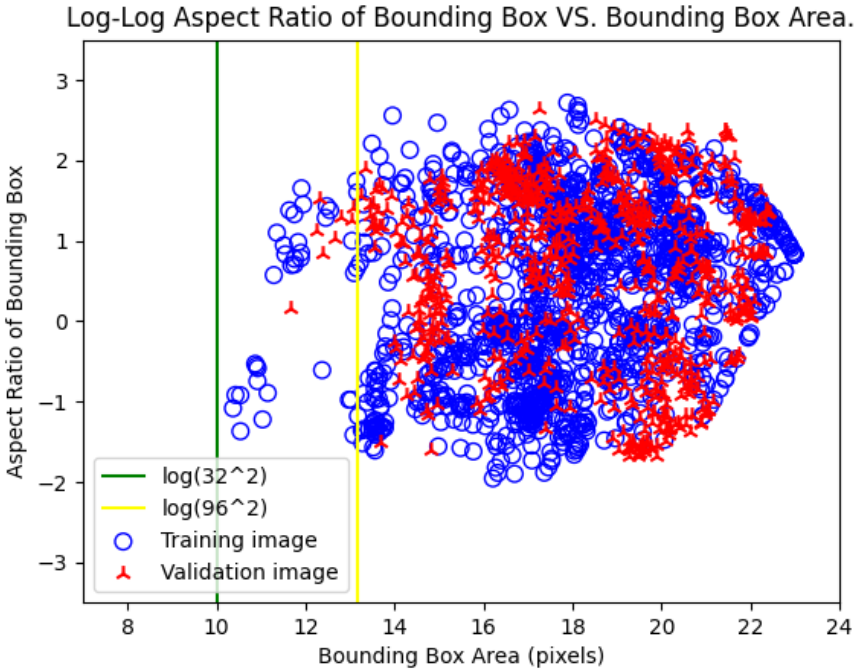
- Figure 27. Image from our dataset with two and three ground-truth boxes.

That will make 1438 ground-truth boxes from 1000 images for training and 745 ground-truth boxes from 500 images for validation. The training and validation images were grouped in 20 subsets of 50 images each for the experiments. This way it's easier to introduce them for training or validation at the necessary moment during the experiment.



- Figure 28. (Left) Width and height of all images in the training and validation datasets. (Right) Number of images by logarithm of the area.

In figure 28 can be seen the size of the image and also the cumulative number of images per area. The 2183 ground-truth boxes are depicted in figure 29 where the aspect ratio of them are plotted against the bounding box area. Since MS COCO metrics are used for this project and they differentiate between small, medium and large objects (whose areas are the number of pixels in the segmentation mask; smaller than 32^2, between 32^2 and 96^2, bigger than 96^2 respectively) two lines are plotted to delimit them. In both figures the logarithmic scale has been used to respond to skewness towards large values in the area of the images and the bounding boxes.



- Figure 29. Log-Log plot of the aspect ratio of a bounding box against its area. Small objects fall left of the green line, medium ones between the green and yellow lines and large objects to the right of the yellow line.

Most of the images used are of the size Full HD (1920×1080) and 4K (3840×2160) which logarithmic values of the area are between 19 and 24. This region is also where around a third of the bounding boxes areas fall. Leaving only a few in the region which COCO considers to be of medium size.

This dataset can be compared to the one used in the project by Emilio Guirado et Al. [5] which is composed of 700 images that contain 945 ground-truth boxes.

5.1. Annotation format

As said before, all annotations have been made using MATLAB's Image Labeler. The app, through the interface seen in figure 26, let's the user create a region of interest label, sublabels and add attributes. For this project the type of region of interest is a rectangle, the main label is 'Whales', the sublabel is a whale species and the whale's posture has been added as an attribute. But in the end neither the sublabel nor the posture have been used.

After all the images have been labeled a ground-truth object is generated from which the necessary information can be extracted alongside the original picture. The data is then stored in a CSV file in which the values are separated by commas. The final structure of the data is as follows:

filename	width	height	class	xmin	ymin	xmax	ymax
/home/danieltorres/dataset/TrainFasterV2/Blue Whale/Validation/1dbyKDw27h8/subset/frame116_4.jpg	1327	392	Whales	1361	1564	2688	1956

- Figure 30. First row of the dataset containing the information of a bounding box from the video with identifier 1dbyKDw27h8.

It indicates the filename with a path to the source image, the bounding box's width, height, class, top left corner (coordinates xmin and ymin) and bottom right corner (coordinates xmax and ymax). It is necessary to have the width and height to comply with the COCO format.

Finally, the CSV file is transformed to TFRecord required by the Tensorflow framework. The TFRecord format is a simple format for storing a sequence of binary records [39]. Because of its lower memory usage and quicker access to the data it improves the performance of the algorithm.

6. Understanding the architecture

The project's development consists on; understanding the machine learning architecture done by Rubén Martín Pinardel in his master's thesis [4], finding and labelling new images to build a new dataset and executing the appropriate experiments.

The dataset must not contain repeated images and the experiments must reflect, among other things, the different approaches to training a network based on the timeline in which the data is introduced. To check if it's better to build a big dataset as soon as possible or continuously collect data and keep training the network. And to test how the number of training steps affect the performance.

6.1. Architecture

The purpose of this project was to continue the work previously done by Rubén Martín Pinardel. In his thesis he worked with three types of architectures. Faster R-CNN with ResNet-101 as feature extractor, Faster R-CNN Inception-ResNet V2 as feature extractor (which uses atrous convolutions) and RetinaNet with ResNet-50 as feature extractor. In this case the project will be developed using the second option, Faster R-CNN Inception-ResNet V2 which was the one that Ruben obtained the best results with. These three architectures were already among the best performing ones be it either by precision or inference time, as shown in figure 31.

This doesn't mean that the performance that we will obtain is going to be the same one as seen in figure 31, since those metrics were obtained using a standard dataset, running on specific hardware for a specific task.

The Faster R-CNNs had already been downloaded from the Tensorflow Object Detection API [41] and it had been initialized with the weights coming from its previous training on the COCO dataset. It was also modified in order to work with a lower number of classes. One final modification was to allow fine-tuning so the weights can change during training.

Method	Data	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Fast R-CNN[29]	train	VGG-16	19.7	35.9	–	–	–	–
Faster R-CNN[8]	trainval	VGG-16	21.9	42.7	–	–	–	–
OHEM[40]	trainval	VGG-16	22.6	42.5	22.2	5.0	23.7	37.9
ION[41]	train	VGG-16	23.6	43.2	23.6	6.4	24.1	38.3
OHEM++[40]	trainval	VGG-16	25.5	45.9	26.1	7.4	27.7	40.3
R-FCN[42]	trainval	ResNet-101	29.9	51.9	-	10.8	32.8	45.0
CoupleNet[43]	trainval	ResNet-101	34.4	54.8	37.2	13.4	38.1	52.0
Faster R-CNN G-RMI[44]	–	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN+++[25]	trainval	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN[15]	trainval35k	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w TDM[45]	trainval	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
Deformable R-FCN[38]	trainval	Aligned-Inception-ResNet	37.5	58.0	40.8	19.4	40.1	52.5
$umd_$det[46]	trainval	ResNet-101	40.8	62.4	44.9	23.0	43.4	53.2
Cascade R-CNN[47]	trainval35k	ResNet-101-FPN	42.8	62.1	46.3	23.7	45.5	55.2
SNIP[48]	trainval35k	DPN-98	45.7	67.3	51.1	29.3	48.8	57.1
Fitness-NMS[49]	trainval35k	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5
Mask R-CNN[11]	trainval35k	ResNeXt-101	39.8	62.3	43.4	22.1	43.2	51.2
DCNv2+Faster R-CNN[39]	train118k*	ResNet-101	44.8	66.3	48.8	24.4	48.1	59.6
G-RMI[44]	trainval32k	Ensemble of Five Models	41.6	61.9	45.4	23.9	43.5	54.9
YOLOv2[30]	trainval35k	DarkNet-53	33.0	57.9	34.4	18.3	35.4	41.9
YOLOv3[32]	trainval35k	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
$SSD300$*[10]	trainval35k	VGG-16	25.1	43.1	25.8	6.6	22.4	35.5
RON384+++[50]	trainval	VGG-16	27.4	49.5	27.1	–	–	–
SSD321[34]	trainval35k	ResNet-101	28.0	45.4	29.3	6.2	28.3	49.3
DSSD321[34]	trainval35k	ResNet-101	28.0	46.1	29.2	7.4	28.1	47.6
SSD512*[10]	trainval35k	VGG-16	28.8	48.5	30.3	10.9	31.8	43.5
SSD513[34]	trainval35k	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513[34]	trainval35k	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet500[33]	trainval35k	ResNet-101	34.4	53.1	36.8	14.7	38.5	49.1
RetinaNet800[33]	trainval35k	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
M2Det512[35]	trainval35k	VGG-16	37.6	56.6	40.5	18.4	43.4	51.2
M2Det512[35]	trainval35k	ResNet-101	38.8	59.4	41.7	20.5	43.9	53.4
M2Det800[35]	trainval35k	VGG-16	41.0	59.7	45.0	22.1	46.5	53.8
RefineDet320[36]	trainval35k	VGG-16	29.4	49.2	31.3	10.0	32.0	44.4
RefineDet512[36]	trainval35k	VGG-16	33.0	54.5	35.5	16.3	36.3	44.3
RefineDet320[36]	trainval35k	ResNet-101	32.0	51.4	34.2	10.5	34.7	50.4
RefineDet512[36]	trainval35k	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4
RefineDet320+[36]	trainval35k	VGG-16	35.2	56.1	37.7	19.5	37.2	47.0
RefineDet512+[36]	trainval35k	VGG-16	37.6	58.7	40.8	22.7	40.3	48.3
RefineDet320+[36]	trainval35k	ResNet-101	38.6	59.9	41.7	21.1	41.7	52.3
RefineDet512+[36]	trainval35k	ResNet-101	41.8	62.9	45.7	25.6	45.1	54.1
CornerNet512[51]	trainval35k	Hourglass	40.5	57.8	45.3	20.8	44.8	56.7
NAS-FPN[18]	trainval35k	RetinaNet	45.4	-	-	-	-	-
NAS-FPN[18]	trainval35k	AmoebaNet	48.0	-	-	-	-	-

- Figure 31. Detection results on the MS COCO test-dev dataset of some typical baselines [40].

6.2. Tensorboard

TensorBoard provides the visualization and tooling needed for machine learning experimentation [42]:

- Tracking and visualizing metrics such as loss and accuracy
- Visualizing the model graph (ops and layers)
- Viewing histograms of weights, biases, or other tensors as they change over time
- Displaying images, text, and audio data

But most important is that it interactively displays the learning rate, the evaluation loss and the training loss.

6.3. GPU

For this project a NVIDIA TITAN Xp GPU with 3840 NVIDIA CUDA cores and 12 GB of GDDR5X memory running at over 11 Gbps. The cores run at 1.6 GHz and the GPU can achieve 12 TFLOPS of computing power.

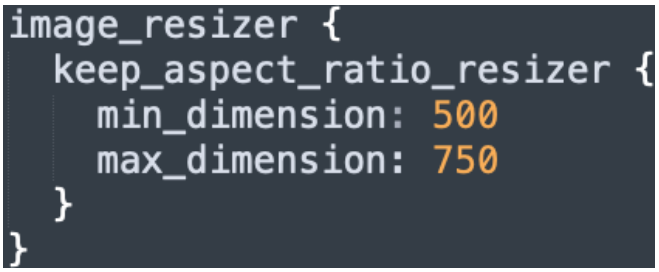
One hundred updates of the parameters which uses one mini-batch as an input of the Faster R-CNN ResNet-101 architecture with only one CPU would take three hundred seconds, while performing the aforementioned task on a GPU would only take 15 seconds.

6.4. Optimization

The architecture has many hyperparameters that can be optimized. Most had already been optimized for standard datasets so only a few of them were altered by Rubén during the development of his thesis. In the next sections of 6.4. we will show how Rubén configured them and the results that he obtained.

6.4.1. Input Image Size

The input image size of the raw input images has been changed to homogenize them. It has been done in a way so it doesn't affect the aspect ratio of the bounding boxes. This was done by resizing the image until it meets the targets and it continues to do so until the aspect ratio has the same original value too.



- Figure 32. Resizer in the configuration file containing the minimum and maximum dimensions for images.

6.4.2. Data Augmentation

The transformations applied were changes on saturation, contrast and brightness:

```
data_augmentation_options {
  random_horizontal_flip {
  }
  random_vertical_flip{
  }
  random_adjust_saturation{
    min_delta: 0.95
    max_delta: 1.05
  }
  random_adjust_contrast{
    min_delta: 0.9
    max_delta: 1.1
  }
  random_adjust_brightness{
    max_delta: 0.1
    #delta randomly picked in the interval [-max_delta, max_delta)
  }
}
```

- Figure 33. Data augmentation options in the configuration file.

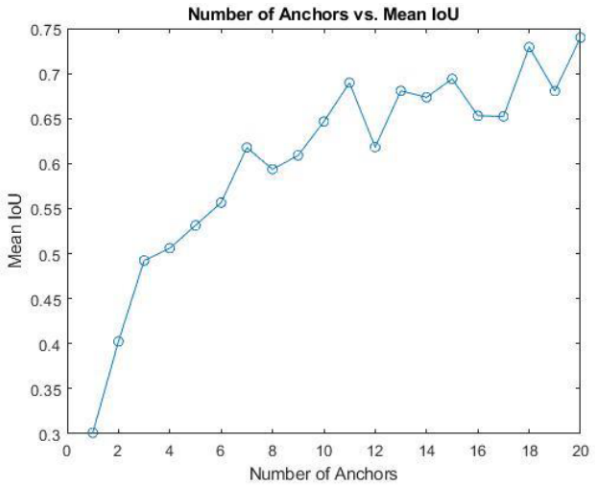
The illumination and contrast changes were picked randomly but are also delimited for what was commonly seen in previous projects [43]. This way the values didn't drastically change the image.

6.4.3. Anchors

The number of anchors, their scale and their aspect ratio can be changed but altering them too much can incur in an increase of the training and inference time. They way the anchors were chosen was by using the k-mean clustering algorithm and the IoU as metric distance.

```
first_stage_anchor_generator {
  grid_anchor_generator {
    scales: [0.1, 0.5, 1.0, 1.5]
    aspect_ratios: [0.5, 1.2, 2.7]
    height_stride: 8
    width_stride: 8
  }
}
```

- Figure 34. Twelve anchors in the configuration file for scales and aspect ratios.



- Figure 35. The mean of IoU against the number of anchors that validate the values in figure 34. Table taken from [4].

Finally it is concluded that the anchors can be more optimized but it would have a high cost in time and low precision benefits.

6.4.4. Learning Rate

The momentum optimizer used on the architecture uses the learning rate as the main parameter.

```
momentum_optimizer: {
  learning_rate: {
    manual_step_learning_rate {
      initial_learning_rate: 0.0016
      schedule {
        step: 900000
        learning_rate: .00003
      }
      schedule {
        step: 1200000
        learning_rate: .000003
      }
    }
  }
  momentum_optimizer_value: 0.9
}
```

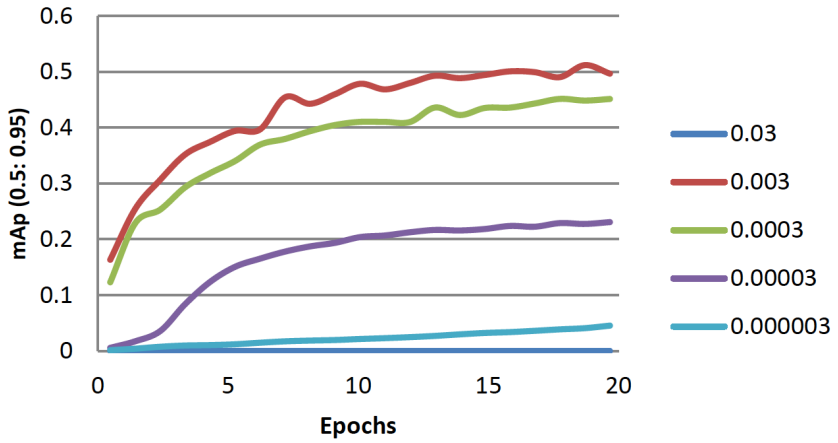
- Figure 36. Momentum optimizer used in the configuration file.

Rubén performed a learning rate optimization of the architecture once they had optimized the anchors, the image input size and selected the desired data augmentation transformations according to their dataset. Then he evaluated the optimized architecture to see how it performed on the task.

6.4.5. Fold Cross Validation

A k-fold cross validation consists of dividing a dataset on k folds, using k-1 folds for training and 1 fold for validating. Using all the folds one time for validating. To calculate the final value the arithmetic mean is made from the k results.

6.4.6. Final setup



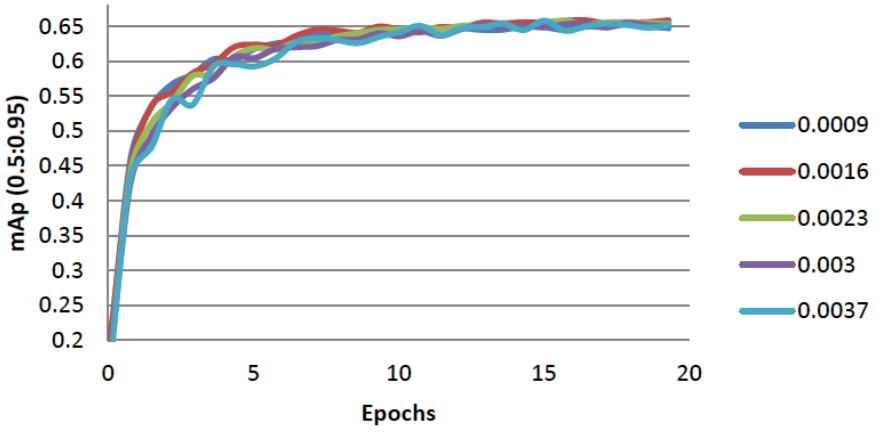
- Figure 37. Faster R-CNN Inception-ResNet V2 logarithmic grid learning rate search with 15% of dataset A and 3-fold cross validation. Figure taken from [4].

As seen in figure 37, the default learning rate of 0.0003 and a learning rate 0.003 have the best results among the possible options. From here, Rubén focused on obtaining the learning rate which optimizes the architecture. Through a grid search around the two best learning rate values, using 5-fold cross validation and the whole dataset of aerial/drone images he obtained the result seen in table 12. Which is the same one that can be seen in the configuration file on figure 36.

Architecture	Learning rate
Faster R-CNN Inception-ResNet v2	[3.7E-03, 3E-03, 2.3E-03, 1.6E-03, 9E-04]

- Table 12. Learning rate tested on the 5-fold cross validation grid search. Table taken from [4].

And out of the possible five values the one with the biggest mAP calculated using the MS COCO metrics over the last 6 epochs is chosen.



- Figure 38. Faster R-CNN Inception-ResNet V2 grid learning rate search with complete dataset A and 5-fold cross validation. Figure taken from [4].

Lr = 0.0009	**Lr = 0.0016**	Lr = 0.0023	Lr = 0.003	Lr = 0.0037
0.6490777	**0.65705483**	0.65534803	0.65242583	0.64923251

- Table 13. Faster R-CNN Inception-ResNet v2 learning rates final accuracy over the last 6 epochs. Table taken from [4].

As we can see from table 13 in bold letters, the learning rate with a value of 0.0016 is the best performing one and it was chosen as the initial learning rate for the configuration of the network (figure 36).

Finally, Rubén concluded that the Faster R-CNN Inception-ResNet V2 is the architecture that achieves the highest mAP on dataset A compared to RetinaNet ResNet 50 and Faster R-CNN ResNet 101. The only downside is the higher inference and training time (Table 7).

Network architecture	Learning rate	Best MS COCO mAP
RetinaNet ResNet 50	0.00005	0.61320246
Faster R-CNN Inception-ResNet V2	0.0016	0.65705483
Faster R-CNN ResNet 101	0.001	0.61938728

- Table 14. The best mAP obtained by the different network architectures and their learning rates.

7. Experiments and results

The experiments consist in seeing how the network progresses alongside the input data. To do so, a pre-trained checkpoint is loaded but we will have to decide which pre-trained model would be the most suitable for this task or if any at all. The training images were already grouped in 20 subsets of 50 images and the validation images were grouped in 10 subsets of 50 images. We used one video for each subset. When training we will introduce two training groups at a time. These moments when we introduce the two training sets will be called iterations since they are sequential during the training. Every time we introduce two subsets for training we will also validate at the end of it. The validation will be done: first separately, one validation subset of 50 images at a time and also with the ten validation sets at the same time (from now on when mentioning 'the validation dataset' we will be referencing the combination of all ten validation subsets). This way we can see the general performance of the network with the validation done in all ten subsets and also see how each validation video performs with its own subset's metrics at every iteration. For the training we will vary the number of training steps and the next checkpoint to be loaded depending on the type of experiment.

For every experiment, the dataset used in Rubén Martín's thesis to train the network is also loaded on top of our data. This way, it allows the network to continue training on it and avoid losing features already learned. After adding Rubén's 6760 ground-truth boxes, we will have a total of 8198 ground-truth boxes for training.

The experiments are divided into eight sections:

- The first one to choose which pre-trained model to load.
- The second (type A) and third (type B) will load the best checkpoint from the first experiment and apply the different variations in input data and training steps.
- The fourth seeing how the model performs with unrelated images.
- The fifth to check the results from the confusion matrices.
- The sixth to look into the effects of the validation images' area and bounding boxes area.
- The seventh to try some data augmentation techniques.
- And finally the eight to predict at which iteration the model will achieve certain mAP values.

And in section 8. we will indicate in table 35 the final metrics obtained and training time for every architecture.

7.1. Choosing a pre-trained model

Loading a pre-trained model is very useful since these kinds of networks often have eight or more layers and over a million parameters. As we have seen, to train them requires big datasets and a lot of time. Before making any experiments we had to see how different checkpoints being loaded affect the performance of the network. The first option is to not load any pre-trained model to see the effects that it has, the second option is to load a detection model pre-trained on the COCO dataset provided for TensorFlow 1 Detection

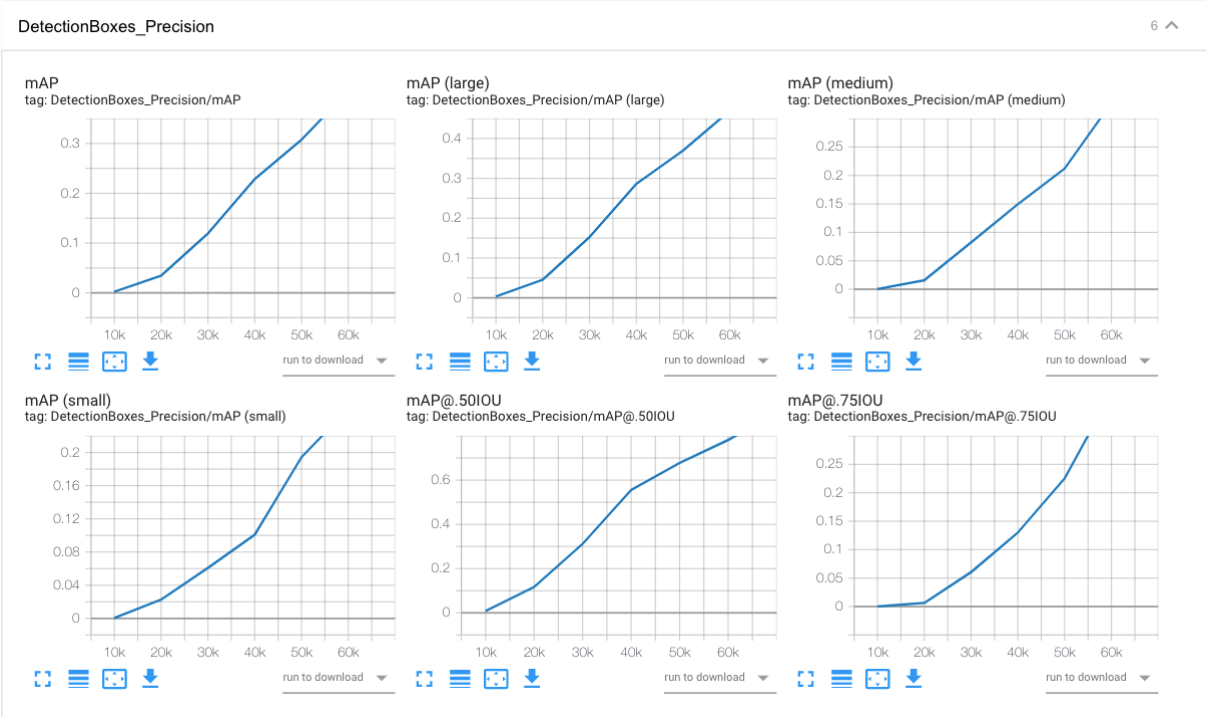
Model Zoo and the third option is to load the same model that was stored in the server from Ruben's work which is also used as a starting point for his training.

Loading previous models can be useful for out-of-the-box inference if we are interested in categories already in those datasets. They are also useful for initializing models when training on novel datasets. It's not recommended not to use any checkpoint because training a network completely from scratch is very costly in time and produces poor results.

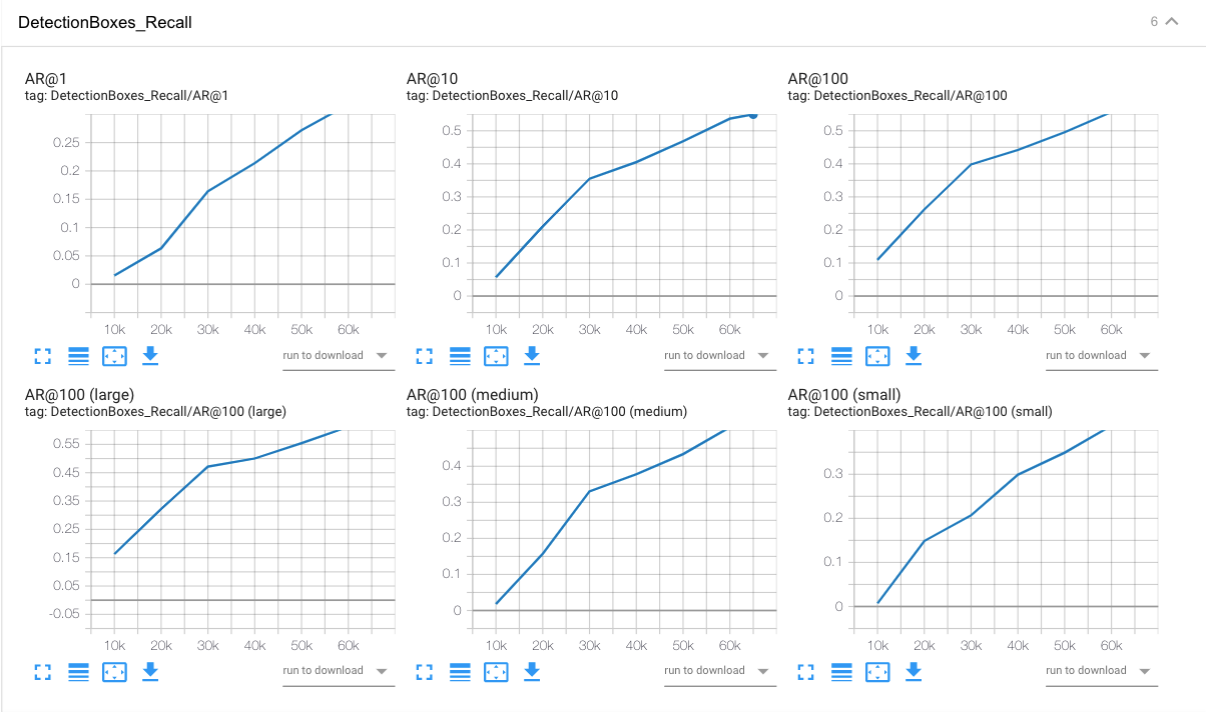
The training and validation datasets used for this will be in both cases the one built by Ruben. So the objective would be to achieve an almost perfect score for all the values of the MS COCO metrics. All three networks will be trained for 65000 steps (19.3 epochs). The rest of the setup is the same as the previous project (section 6.4.).

7.1.1. No checkpoint loaded

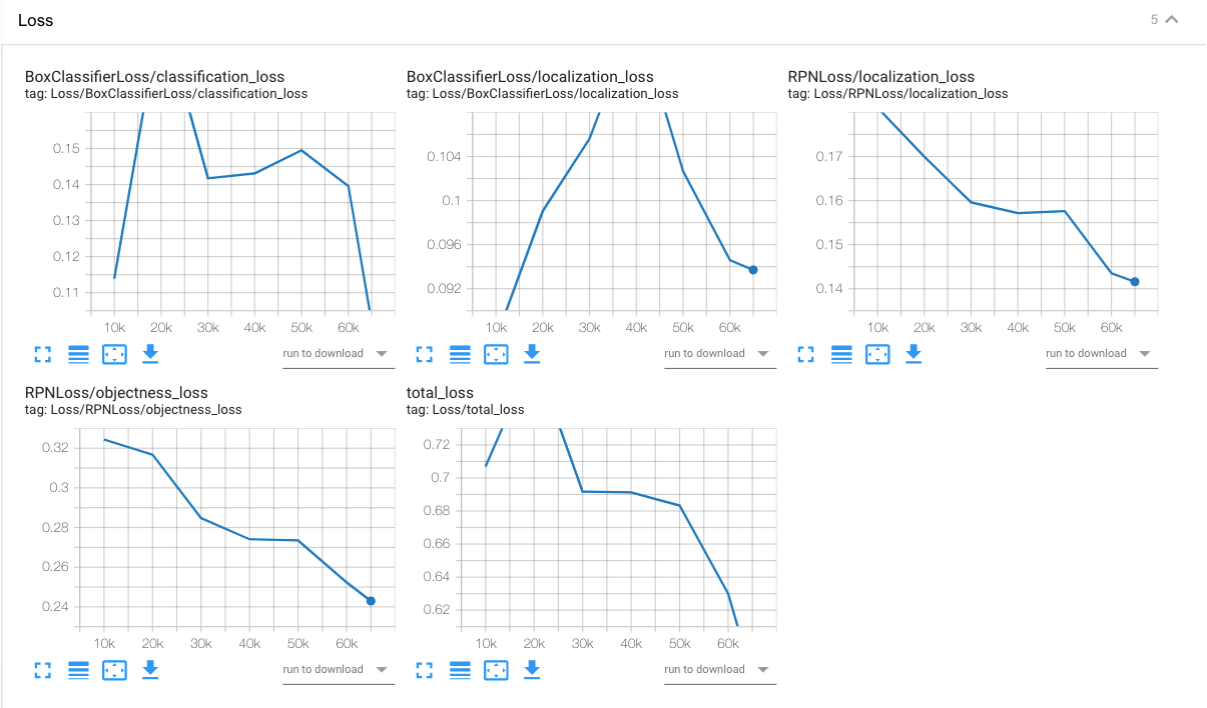
Here we can confirm that not loading any kind of pre-trained model for the checkpoint produces very low values of precision. After the 65K training steps the mAP obtained from testing with the training dataset is 0.439 which is not much of an improvement compared to the 0.37 that offers the pre-trained model from TensorFlow 1 Detection Model Zoo [41]. And specially given the amount of time it took to train and that it's testing with the same images that it has been learning from. Further below we can see how the mAR doesn't reach 0.600



- Figure 39. Precision results from training the network without loading a starting checkpoint.



- Figure 40. Recall results from training the network without loading a starting checkpoint.

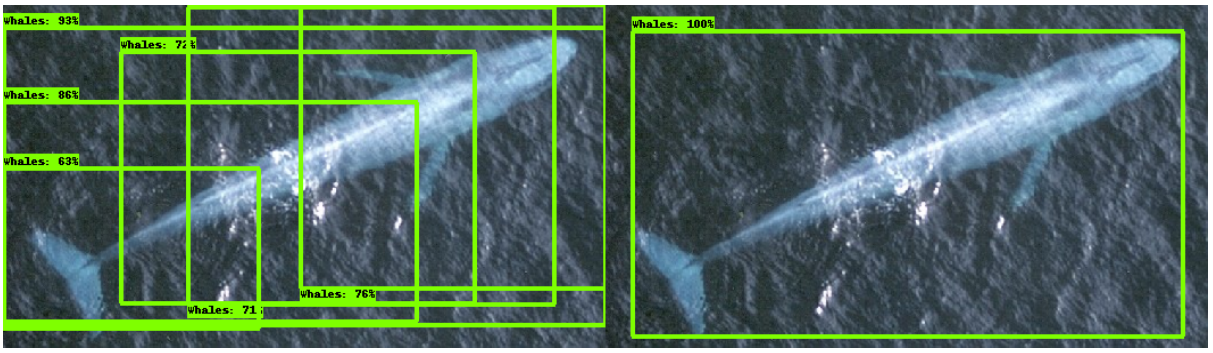


- Figure 41. Loss results from training the network without loading a starting checkpoint.

Table 15 is the output that the finished training gives of the most important MS COCO metrics instead of the graphic representations.

MS COCO Metrics	Result
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.439
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.843
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.408
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	0.273
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.373
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.506
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.337
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.550
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.566
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	0.412
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.519
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.622

- Table 15. MS COCO Metrics from training the network without loading a starting checkpoint.

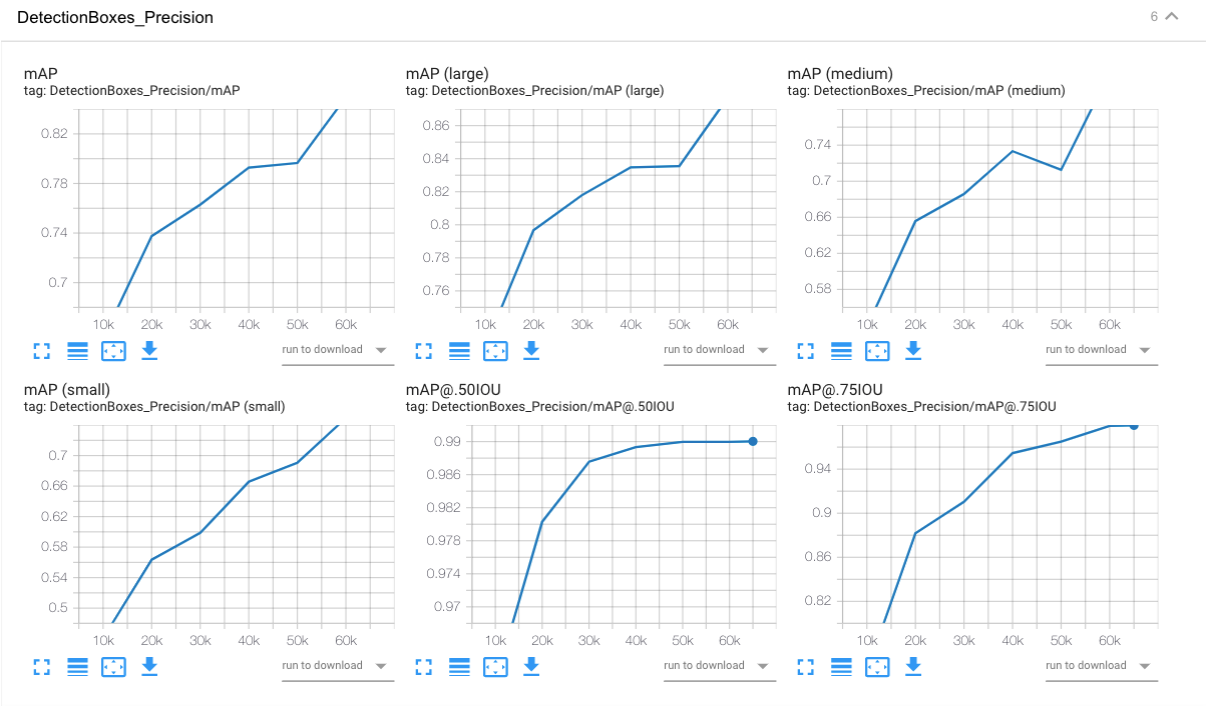


- Figure 42. (Left) Object detections inferred from the no checkpoint loaded model. (Right) Ground-truth bounding box.

Figure 42 shows that the model is giving many FP which contributes negatively to the mAP. In this case the whale pictured is a Blue whale which is known to have a higher difference in contrast and brightness compared to the ocean. The multiple bounding boxes drawn may represent the ability that it has to recognize these differences. But at the same time it doesn't seem that it has learned the features necessary to know the shape. Thus, it draws multiple boxes from the tail to multiple points along the spine of the whale.

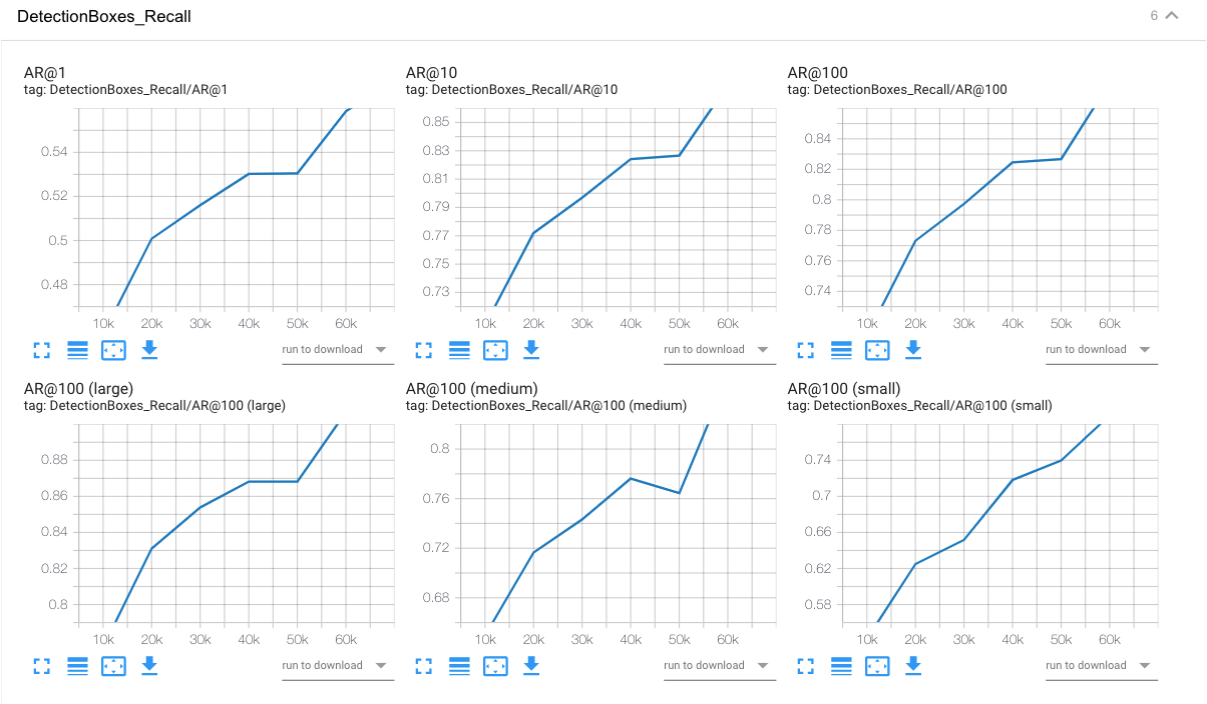
7.1.2. Model pre-trained on the COCO dataset

Previous to the training with Rubén's pre-trained model, and since we didn't know the exact source of Ruben's pre-trained model or the version used we decided to repeat the process seen in 7.1.1. with the model available to download from TensorFlow 1 Detection Model Zoo [41].

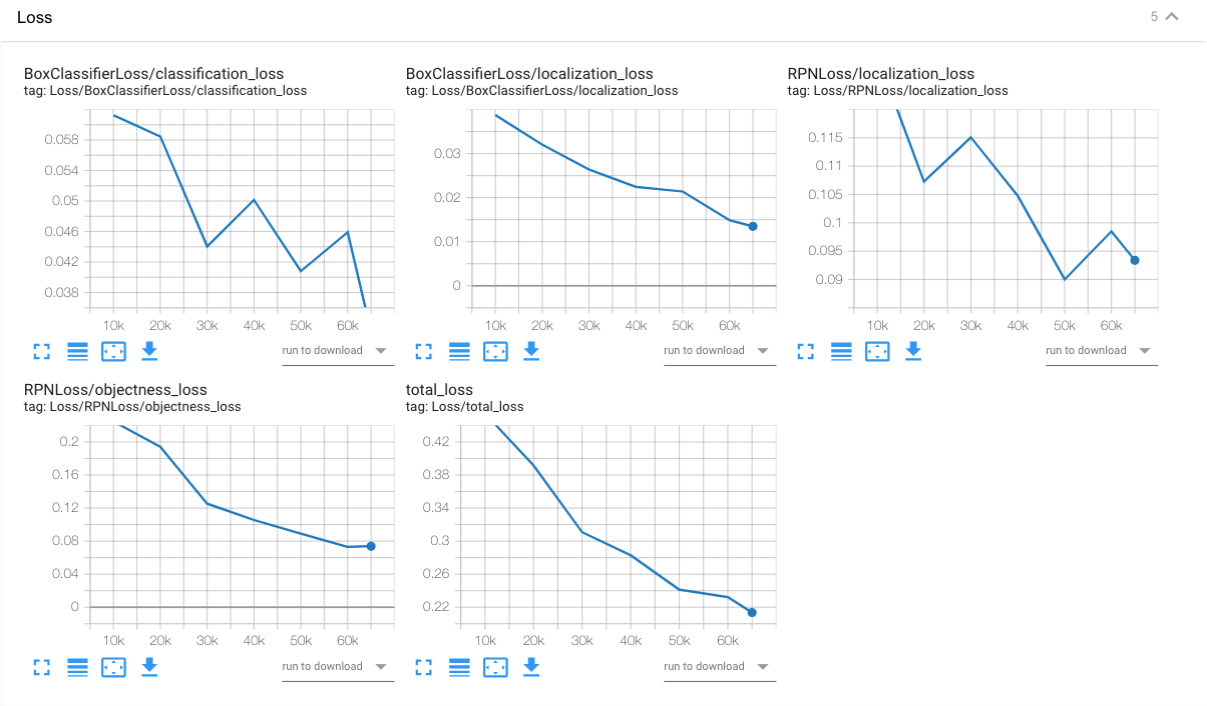


- Figure 43. Precision results from training the network using the pre-trained model on the COCO dataset as a starting point.

In this case the trained model gives a mAP of 0.863 in the training set and an almost perfect 0.990 for IoU equal to 0.50. Which means it's capable of detecting and classifying correctly all the data with which has been trained.



- Figure 44. Recall results from training the network using the pre-trained model on the COCO dataset as a starting point.



- Figure 45. Loss results from training the network using the pre-trained model on the COCO dataset as a starting point.

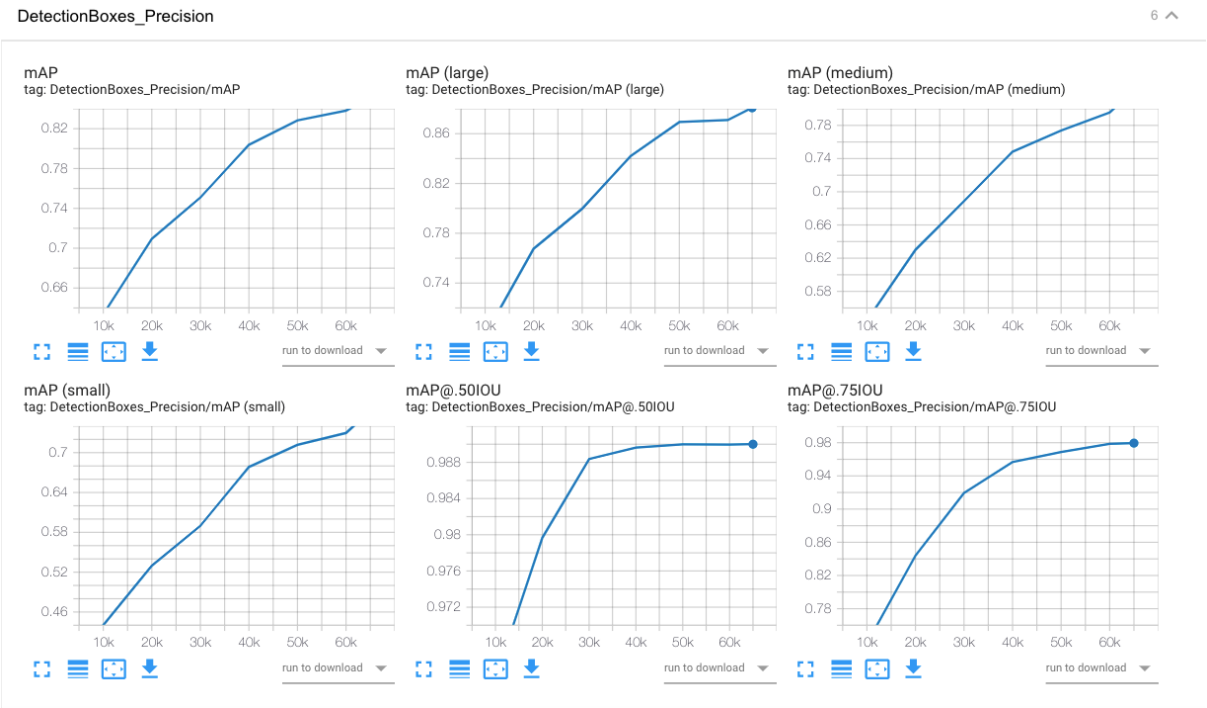
MS COCO Metrics	Result
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.863
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.990
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.980
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	0.772
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.829
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.890
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.566
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.886
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.886
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	0.808
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.862
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.915

- Table 16. MS COCO Metrics from training the network using the pre-trained model on the COCO dataset as a starting point.

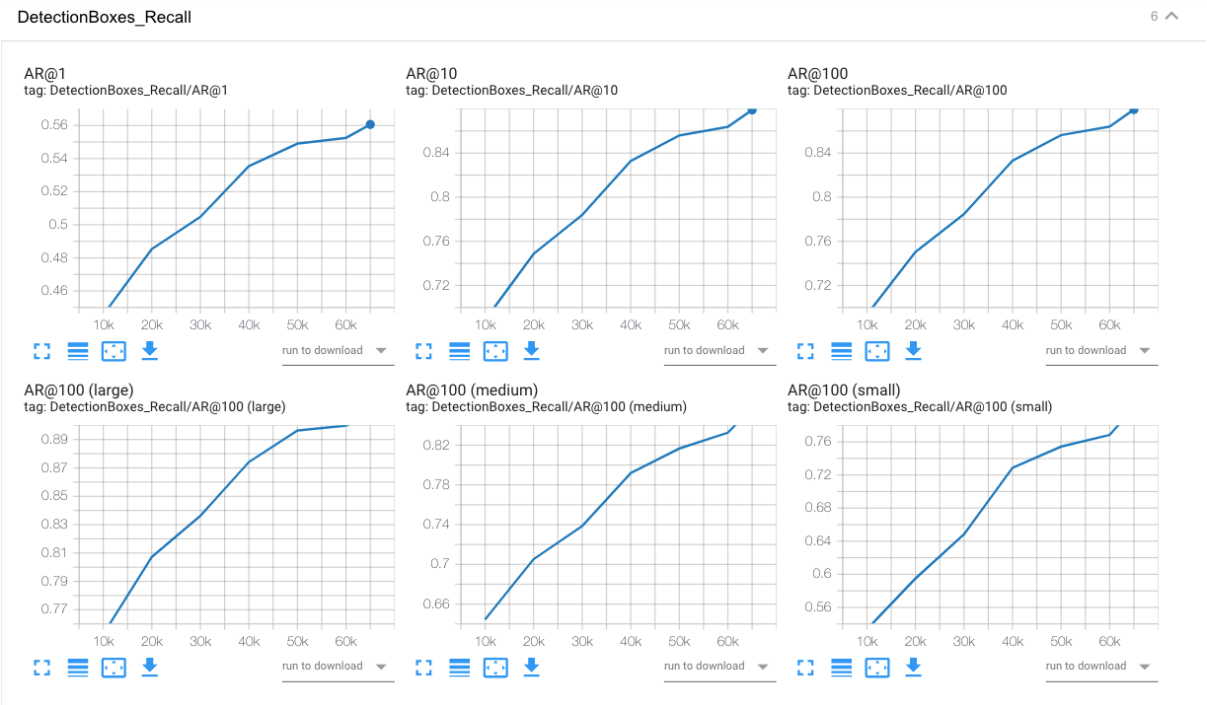
7.1.3. Previous network starting checkpoint

For this task we decided to load the pre-trained model used by Ruben to see if the results would coincide with the one on section 7.1.2. If the results are similar it would mean that he also used a pre-trained model whose training was based on novel or generalized datasets. Otherwise, if the results show that the mAP curve performs better then the training used a

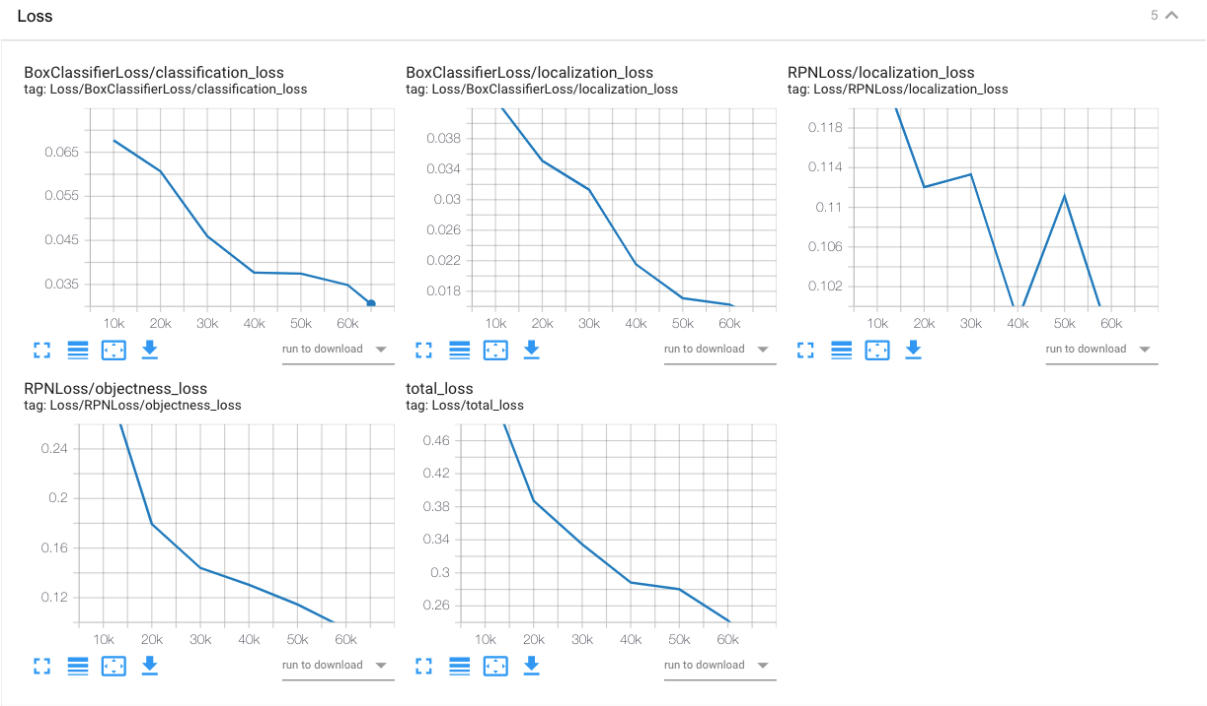
more specific dataset that includes the same type of data as the one that we are using now and will produce better results.



- Figure 46. Precision results from training the network using the same starting checkpoint as Rubén.



- Figure 47. Recall results from training the network using the same starting checkpoint as Rubén.



- Figure 48. Loss results from training the network using the same starting checkpoint as Rubén.

MS COCO Metrics	Result
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.852
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.990
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.980
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	0.763
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.822
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.881
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.561
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.879
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.879
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	0.802
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.857
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.908

- Table 17. MS COCO Metrics from training the network using the same starting checkpoint as Rubén.

The results from table 17 don't differ much from the one on the previous experiment in table 17. The mAP for Iou equal to 0.5 is the same and there's only a tiny difference of 0.011 improvement using this model. This indicates that the pre-trained model used by Ruben is probably the same that's available online and the difference between them it's due to the initialization of the layers during the training. For this reason and for continuity we have

decided to choose this pre-trained model, the one that Rubén also used as a starting point, to load as the initial checkpoint and begin training our experiments.

From now on, every time we show the performance of a new model we will also show a column with the iteration 0. This iteration 0 is the result of validating this model from section 7.1.3. with the validation dataset. So we can compare the results obtained from the training with the base value.

7.2. Non-cumulative training (Type A)

For this experiment we will train the network from the checkpoint mentioned in the previous section (6.1.3.) and do different amounts of training steps (3K, 10K and 30K). Every time we proceed to train the network we will use Ruben's complete dataset and two times the number of the iteration of our training datasets. Each training dataset is composed of 50 images from a single video. This way, the first iteration will train on Ruben's dataset and two of our training datasets. And when getting to the tenth iteration we will be training the network for the number of training steps mentioned in each section, with Ruben's complete dataset and all of our 20 training datasets.

These three setups for the experiments would simulate at a smaller scale the process of creating a dataset for a specific task and trying to obtain a functional model as soon as possible and without further development.

7.2.1. 3K training steps

For each iteration we will train for 3K training steps from the pre-trained model. Making it a total of 68K training steps for every column. Each iteration takes a training time of 31.52 minutes and a total of 5.25 hours for the whole experiment.

MS COCO Metrics / Iteration	0	1	2	3	4	5	6	7	8	9	10
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.445	0.462	0.480	0.477	0.475	0.483	0.455	0.508	0.512	0.510	0.521
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.763	0.798	0.798	0.793	0.801	0.809	0.795	0.834	0.841	0.824	0.848
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.461	0.461	0.494	0.489	0.498	0.495	0.430	0.522	0.518	0.531	0.562
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.002	0.001	0.001	0.002	0.002	0.005	0.007	0.004	0.035	0.016	0.054
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.458	0.473	0.489	0.485	0.482	0.491	0.462	0.517	0.517	0.515	0.526
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.393	0.406	0.415	0.415	0.413	0.416	0.399	0.430	0.426	0.427	0.438
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.544	0.553	0.561	0.561	0.556	0.577	0.553	0.593	0.597	0.587	0.601
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.567	0.573	0.580	0.579	0.585	0.606	0.574	0.607	0.617	0.608	0.619
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.225	0.050	0.138	0.138	0.087	0.225	0.150	0.250	0.338	0.188	0.325
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.571	0.579	0.585	0.584	0.591	0.610	0.578	0.611	0.620	0.612	0.623

- Table 18. MS COCO metrics for 3K training steps of non-cumulative training.

The results in table 18 show that the AP improves by 0.076 (17%) and the AR improves by 0.052 (9%) from the initial point. Although the learning seems to regress for iteration 6. The

results are also better if we only consider large objects given that most of our bounding boxes were located in the ‘large’ side of the line in figure 29. The -1.000 values are because there were no objects that met the criteria for that specific metric.

It seems like this is a good starting point now that we could check that the network learns progressively during the whole training and the increase looks substantial given the little time dedicated to the task compared to the following ones.

7.2.2. 10K training steps

For each iteration we will train for 10K training steps from the pre-trained model. Making it a total of 75K training steps for every column. Each iteration takes a training time of 1.75 hours and a total of 17.51 hours for the whole experiment.

MS COCO Metrics / Iteration	0	1	2	3	4	5	6	7	8	9	10
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.445	0.454	0.468	0.483	0.473	0.498	0.507	0.506	0.522	0.530	0.538
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.763	0.795	0.795	0.809	0.799	0.820	0.842	0.837	0.851	0.849	0.846
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.461	0.451	0.484	0.495	0.476	0.519	0.510	0.507	0.535	0.564	0.569
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.002	0.002	0.002	0.002	0.004	0.003	0.010	0.008	0.031	0.015	0.064
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.458	0.464	0.480	0.492	0.482	0.509	0.514	0.513	0.528	0.538	0.543
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.393	0.398	0.411	0.418	0.414	0.428	0.419	0.425	0.434	0.435	0.449
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.544	0.543	0.557	0.559	0.569	0.584	0.597	0.588	0.598	0.611	0.620
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.567	0.568	0.576	0.582	0.591	0.609	0.612	0.612	0.610	0.631	0.631
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.225	0.150	0.150	0.113	0.325	0.163	0.300	0.212	0.263	0.362	0.325
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.571	0.573	0.581	0.587	0.594	0.614	0.615	0.616	0.614	0.634	0.634

- Table 19. MS COCO metrics for 10K training steps of non-cumulative training.

The results in table 19 show that the AP improves by 0.093 (21%) and the AR improves by 0.064 (11%) from the initial point. Which is a 4% and 2% increase respectively compared to the previous experiment (section 7.2.1.). In this case we can see that it has disappeared the regresion from iteration 6 and instead is present at iteration 4. Meaning that the different amount of training steps also change the way that the features are learned during the training. The values for large objects are once again higher than the average of all areas.

7.2.3. 30K training steps

For each iteration we will train for 3K training steps from the pre-trained model. Making it a total of 95K training steps for every column. Each iteration takes a training time of 5.25 hours and a total of 52.53 hours for the whole experiment.

MS COCO Metrics / Iteration	0	1	2	3	4	5	6	7	8	9	10
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.445	0.456	0.469	0.487	0.478	0.458	0.494	0.489	0.496	0.524	0.521
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.763	0.788	0.798	0.805	0.812	0.776	0.837	0.819	0.823	0.840	0.831
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.461	0.461	0.481	0.494	0.483	0.455	0.497	0.482	0.505	0.559	0.544
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.002	0.002	0.002	0.003	0.002	0.002	0.005	0.001	0.002	0.052	0.017
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.458	0.465	0.479	0.495	0.490	0.471	0.500	0.499	0.504	0.530	0.527
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.393	0.398	0.408	0.422	0.410	0.404	0.416	0.416	0.417	0.435	0.439
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.544	0.536	0.562	0.560	0.570	0.542	0.583	0.579	0.576	0.608	0.597
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.567	0.567	0.586	0.581	0.591	0.571	0.598	0.595	0.597	0.626	0.615
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.225	0.100	0.175	0.138	0.163	0.138	0.138	0.138	0.188	0.300	0.225
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.571	0.572	0.591	0.585	0.596	0.576	0.603	0.600	0.601	0.630	0.620

- Table 20. MS COCO metrics for 30K training steps of non-cumulative training.

The results in table 20 show that the AP improves by 0.076 (17%) and the AR improves by 0.048 (8%) from the initial point. The performance is the same for AP and worse for AR compared to the first experiment (section 7.2.1.). It's also irregular between iteration2 and 9. This makes clear that the 5.25 hours that it took to train this model for the tenth iteration is not worth it. And unlike the last two experiments this one seems to be affected negatively by going over the features of the last two training sets for 30K steps.

7.3. Cumulative training (Type B)

This experiment will be similar to type A except that every iteration will use the previous iteration's model. This way we will check how training for more steps over the same datasets reinforces or not the features learned. At the same time this means that the videos' dataset that are introduced the earliest will be the ones that more features are extracted from.

This experiment would simulate at a smaller scale the process of importing the necessary network architecture for a specific task and focusing the development of the project in constantly adding new data and fine tuning the network's parameters along the way.

7.3.1. 3K training steps

For each iteration we will train for 3K training steps from the pre-trained model of the previous iteration. Making it 65K training steps plus 3K training steps for every column and 95K training steps at the tenth iteration. Each iteration takes a training time of 31.52 minutes and a total of 5.25 hours for the whole experiment.

MS COCO Metrics / Iteration	0	1	2	3	4	5	6	7	8	9	10
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.445	0.468	0.476	0.488	0.475	0.510	0.521	0.509	0.525	0.529	0.525
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.763	0.801	0.802	0.815	0.807	0.843	0.863	0.854	0.851	0.857	0.845
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.461	0.471	0.478	0.497	0.494	0.508	0.505	0.509	0.557	0.549	0.537
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.002	0.001	0.002	0.004	0.003	0.007	0.016	0.010	0.017	0.074	0.047
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.458	0.477	0.485	0.495	0.484	0.516	0.527	0.515	0.531	0.533	0.530
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.393	0.406	0.413	0.416	0.414	0.418	0.426	0.422	0.426	0.437	0.439
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.544	0.559	0.564	0.565	0.561	0.591	0.605	0.587	0.601	0.612	0.603
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.567	0.582	0.586	0.588	0.580	0.612	0.622	0.603	0.620	0.629	0.620
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.225	0.150	0.100	0.087	0.163	0.300	0.287	0.175	0.237	0.438	0.250
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.571	0.587	0.592	0.593	0.585	0.615	0.626	0.607	0.624	0.631	0.624

- Table 21. MS COCO metrics for 3K training steps of cumulative training.

The results in table 21 show that the AP improves by 0.08 (18%) and the AR improves by 0.053 (9%) from the initial point. It is 1% better than the other experiment (section 7.2.1.) that uses 3K training steps. It also shows better results for large objects, 0.004 more and 0.076 overall. This continues to make the experiment from section 7.2.3. the worst option. As for the recall right now it's the best second model.

7.3.2. 10K training steps

For each iteration we will train for 10K training steps from the pre-trained model of the previous iteration. Making it 65K training steps plus 10K training steps for every column and 165K training steps at the tenth iteration. Each iteration takes a training time of 1.75 hours and a total of 17.51 hours for the whole experiment.

MS COCO Metrics / Iteration	0	1	2	3	4	5	6	7	8	9	10
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.445	0.480	0.465	0.477	0.471	0.479	0.493	0.482	0.484	0.510	0.496
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.763	0.816	0.785	0.808	0.804	0.809	0.846	0.845	0.840	0.852	0.837
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.461	0.497	0.484	0.498	0.469	0.471	0.481	0.461	0.491	0.539	0.519
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.002	0.003	0.001	0.001	0.003	0.004	0.005	0.007	0.042	0.060	0.113
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.458	0.490	0.477	0.486	0.479	0.489	0.500	0.488	0.489	0.516	0.501
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.393	0.414	0.401	0.417	0.414	0.414	0.433	0.433	0.427	0.442	0.443
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.544	0.570	0.548	0.559	0.561	0.577	0.598	0.593	0.595	0.609	0.604
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.567	0.589	0.573	0.581	0.583	0.604	0.612	0.611	0.614	0.626	0.616
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.225	0.163	0.125	0.087	0.188	0.388	0.250	0.300	0.275	0.275	0.250
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.571	0.594	0.578	0.587	0.588	0.607	0.616	0.615	0.618	0.630	0.620

- Table 22. MS COCO metrics for 10K training steps of cumulative training.

The results in table 22 show that the AP improves by 0.051 (11%) and the AR improves by 0.049 (9%) from the initial point. This is the worst model until now in terms of both precision and recall. Seems like until iteration 6 it's going over the same images and it's not capable of improving its performance with the features from those training datasets.

It can be compared to both section 7.2.2. which does the same amount of training steps and section 7.3.1 which does 7K less steps with the same method yet both perform much better than this one.

7.3.3. 30K training steps

For each iteration we will train for 30K training steps from the pre-trained model of the previous iteration. Making it 65K training steps plus 30K training steps for every column and 365K training steps at the tenth iteration. Each iteration takes a training time of 5.25 hours and a total of 52.53 hours for the whole experiment.

MS COCO Metrics / Iteration	0	1	2	3	4	5	6	7	8	9	10
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.445	0.474	0.451	0.479	0.430	0.441	0.462	0.462	0.470	0.487	0.484
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.763	0.795	0.798	0.812	0.773	0.796	0.822	0.826	0.829	0.839	0.829
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.461	0.491	0.462	0.478	0.431	0.421	0.450	0.438	0.455	0.486	0.487
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.002	0.003	0.002	0.003	0.002	0.003	0.001	0.001	0.007	0.018	0.065
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.458	0.484	0.459	0.487	0.441	0.450	0.471	0.470	0.477	0.492	0.488
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.393	0.411	0.411	0.424	0.403	0.417	0.420	0.431	0.431	0.439	0.441
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.544	0.562	0.553	0.571	0.549	0.578	0.591	0.590	0.596	0.609	0.606
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.567	0.581	0.573	0.601	0.573	0.599	0.604	0.603	0.612	0.620	0.617
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	0.225	0.163	0.100	0.150	0.125	0.212	0.062	0.087	0.237	0.225	0.250
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.571	0.585	0.578	0.606	0.578	0.603	0.610	0.609	0.616	0.624	0.621

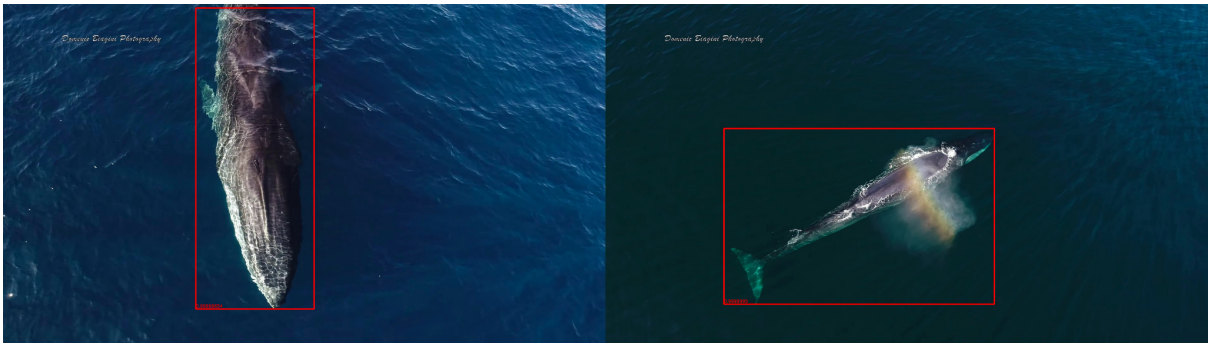
- Table 23. MS COCO metrics for 30K training steps of cumulative training.

The results in table 23 show that the AP improves by 0.039 (9%) and the AR improves by 0.05 (9%) from the initial point. This becomes the worst performing model. For multiple iterations its performance drops below the starting metrics. Even the average for large objects falls under 0.500.

To conclude the experiment from section 7.2.2., (Type A) - 10K training, is on average the best performing one. It archives very good results on three of the videos with above 0.79 AP and it only deeps below 0.445 (which is the initial average at iteration 0) for two videos. We can see how it perform for the rest of the videos:

Species	Video ID	AP at 10
Blue	1dbyKDw27h8	0.499
Blue	GmK1XjaUB_4	0.808
Fin	EOyywJx760w	0.852
Fin	Q-W9VD8UZ6k	0.370
Fin	Rrd-TSH-WG4	0.797
Grey	Bd6dwO83U_4	0.292
Grey	eTlZjK4VvmY	0.580
Humpback	SR0mznc3gg0	0.491
Right	OL4qTmNhw8E	0.509
Right	QscfACfi2EI	0.546

- Table 24. Performance of experiment non-cumulative training (Type A) - 10K training steps per video of the validation dataset.



- Figure 49. Detections done by model non-cumulative training (Type A) - 10K training steps on video EOyywJx760w.

7.4. Non-whale images

For this experiment we chose the best performing model, the non-cumulative training (Type A) - 10K training steps, and used it to infer over a dataset composed of non related images. In this case we used one the best covers ever made in the history of music. Glove and Boots' cover of the Beatles' "All Together Now" [44].

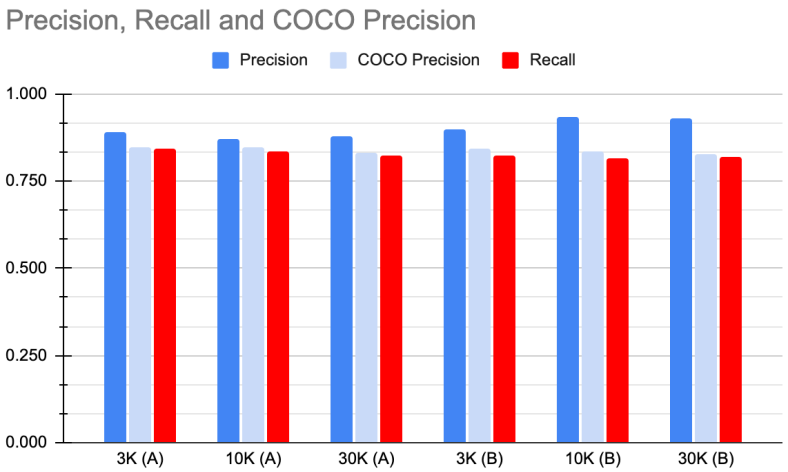
We extracted one frame for every second of the video and randomly selected 50 of them to be inferred upon.

- Figure 50. Doing inference with the best performing model non-cumulative training (Type A) - 10K training steps over non-whale images.

In an ideal situation, no false positive should be predicted. However, we obtained 94 false positives out of the 50 images given on the validation set. As seen in figure 50 it mostly detected objects such as the puppets, the cow and Biz Markie as whales. This seems to indicate that it is capable of recognizing shapes of large objects and at least it doesn't infer false positives out of random areas of the images. Nonetheless, the percentages for object detection are quite high. Ranging from 0.8 to 0.99 in letters, 0.6 to 0.99 on the puppets and 0.5, 0.94 and 0.99 for a leg, the cow's horns and a couch respectively.

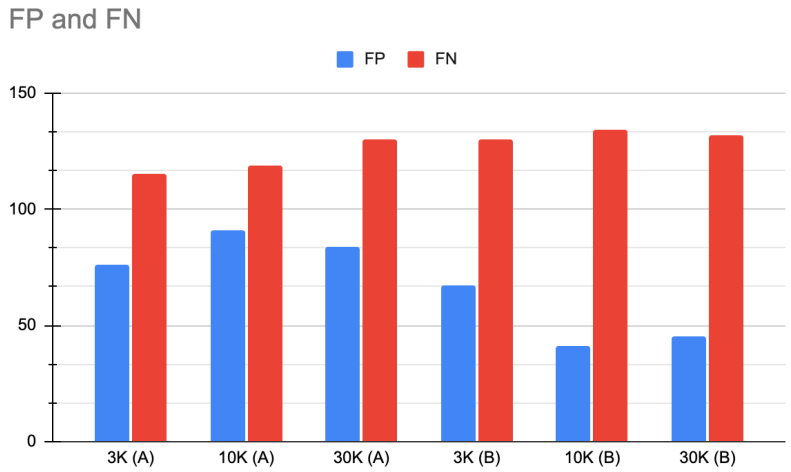
7.5. Confusion matrices

Besides the MS COCO metrics we also generated the confusion matrices in order to understand why each network performs the way it does. We want to check how the number of true positives, false positives and false negatives behave for each model on their last iteration. This way we can observe in a Layman's approach if the ground-truth boxes are being detected for a IoU value of 0.5.

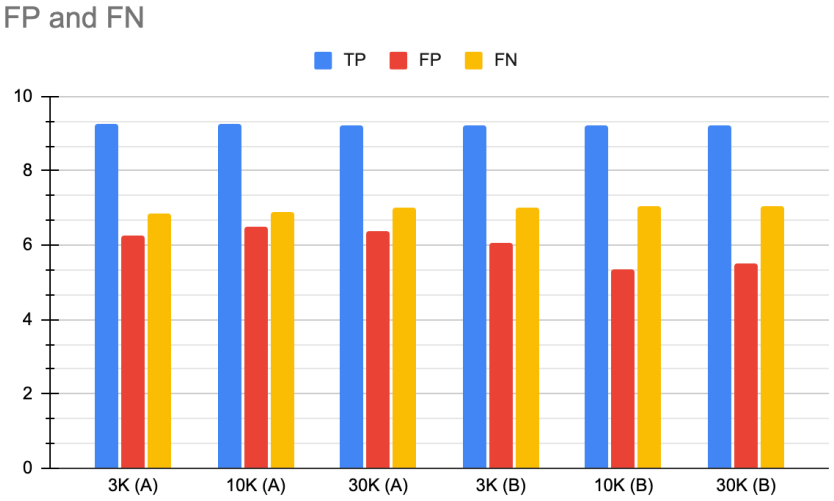


- Figure 51. Precision and recall calculated with the confusion matrices and MS COCO precision per experiment.

Because the number of true positives are much higher than the number of false positives and false negatives the following figure only depicts the last two aforementioned variables. To include the true positives we will display them applying the logarithmic scale to all three in figure 53.



- Figure 52. Number of false positives and false negatives per experiment.



- Figure 53. Number of true positives, false positives and false negatives per experiment on a logarithmic scale.

On figure 51 we can see that by calculating the precision and recall only using the confusion matrices it shows a different perspective than the MS COCO Metrics. For instance, the networks perform much better with the cumulative training method described in section 7.3. This is because the number of true positives doesn't vary much (figure 53) and as denoted on figure 52, the number of false positives decrease for those methods and reaches its minimum point with the cumulative training (Type B) - 10K training steps. Even though that network ranked fifth for COCO precision it is the best one at not misclassifying.

Nonetheless the non-cumulative training (Type A) - 10K training steps continues to be the best model. Here by only looking at the confusion matrices we are not taking into account

the area of intersection between the ground-truth box and predicted bounding box nor the interpolated precision averaged across all unique recall levels.

7.6. Validation images area and performance

We know that if the ratio between the image and the bounding box areas increase too much, as it does for satellite images which have a very high ratio, it affects the general performance of a network.

Validation ID	Bd6dwO83U_4	Q-W9VD8UZ6k	SR0mznc3gg0	1dbyKDw27h8	OL4qTmNhw8E	QscfACfi2EI	eTlZjK4VvmY	Rrd-TSH-WG4	GmK1XjaUB_4	EOyywJx760w
Logarithm of Image area	22.98	20.98	22.98	22.98	20.98	18.64	20.98	22.98	22.89	22.98
Logarithm of average bounding box area	17.97	16.07	20.58	20.22	16.89	15.80	19.36	20.17	20.65	21.56
Ratio (Image area:Bounding box)	1.28	1.31	1.12	1.14	1.24	1.18	1.08	1.14	1.11	1.07
(A) 10K (AP) @[0.50:0.95 | all | 100]	0.292	0.370	0.491	0.499	0.509	0.546	0.580	0.797	0.808	0.852
(A) 10K (AR) @[0.50:0.95 | all | 100]	0.461	0.515	0.573	0.581	0.628	0.618	0.658	0.844	0.852	0.875

- Table 25. Logarithm of the image area, average bounding box area, ratio and performance for each validation set.

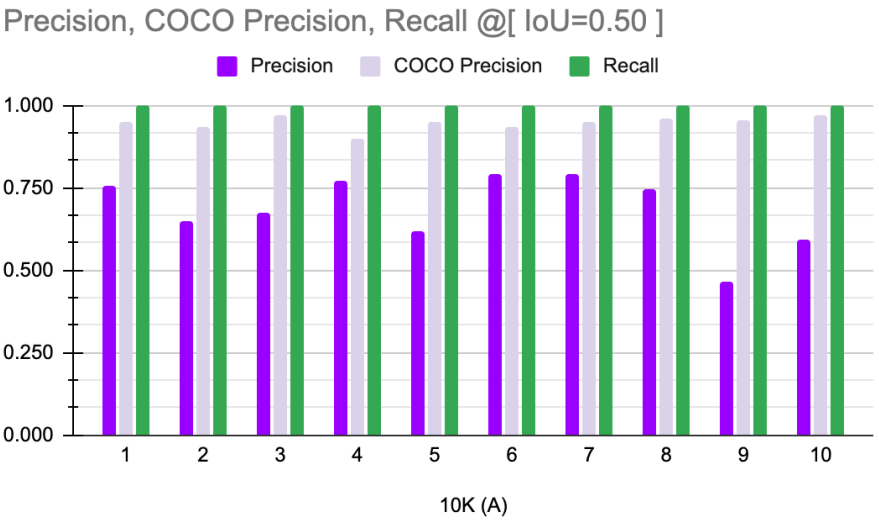
For this experiment we wanted to see if the ratio between the average area of the bounding box and the area of the images affected the performance. From table 25 we can observe how the two sets with the biggest ratio perform the worst but it doesn't mean it’s a direct cause. Because for every other validation set the values for the ratio are inside a small range but the performance isn’t. Differences of 0.4 in mAP and 0.3 in mAR can be caused in this case because of the areas.

Nevertheless, the experiment should be repeated with videos whose difficulty and species don’t change and with a larger variation of ratios in order to prove a unique facto.

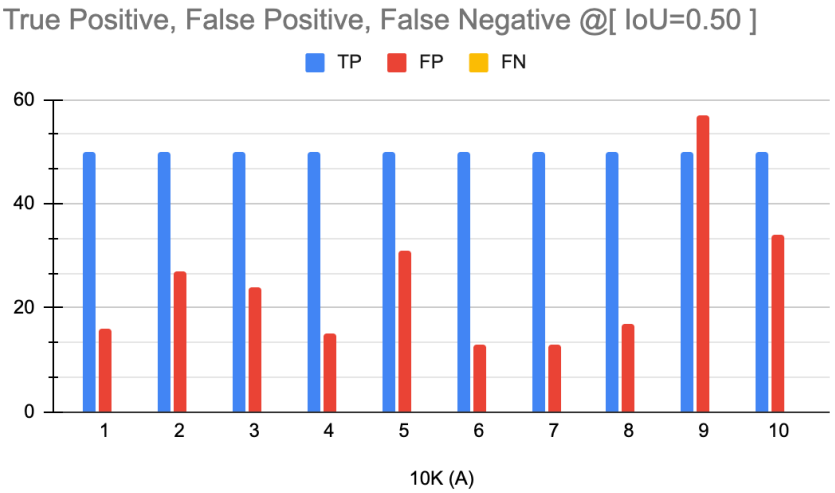
7.7. Data augmentation

Now that we know that the ratio between the image area and the bounding boxes wasn’t a big contributing factor we decided to take another approach by looking at specific cases of the validation sets.

One particular case was the validation video with ID eTlZjK4VvmY. After disseminating the performance by iteration and obtaining the values for the confusion matrix for the non-cumulative training (Type A) - 10K training steps we can see the following.



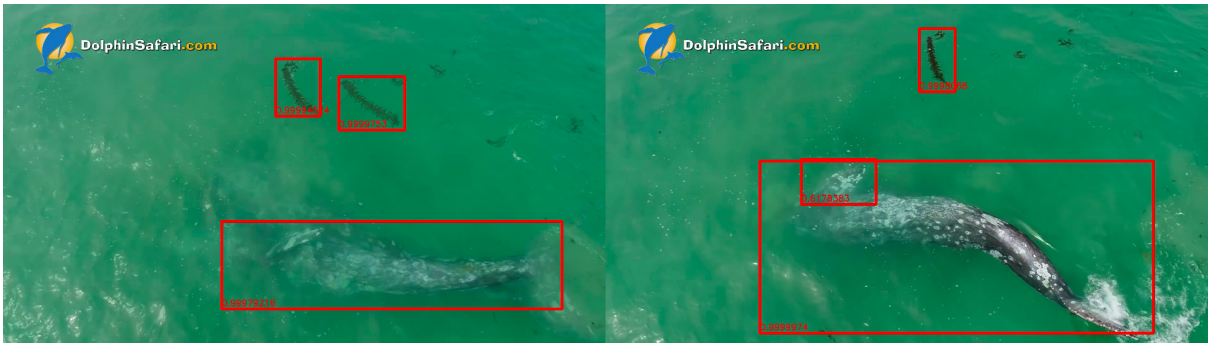
- Figure 54. Precision and recall calculated with the confusion matrix and MS COCO precision per iteration of the experiment non-cumulative training (Type A) - 10K training steps validated on the dataset eTlZjK4VvmY.



- Figure 55. Number of false positives and false negatives per iteration of the experiment non-cumulative training (Type A) - 10K training steps validated on the dataset eTlZjK4VvmY.

The model is correctly detecting all the true positives for every iteration since 50 is the total number of bounding boxes in the dataset and there are no false negatives in whales not being detected. But what's making the model perform badly is the number of false positives.

As it turns out this was because out of the 50 images there were algaes and the whale's flipper being detected as a whale with pretty high confidence.



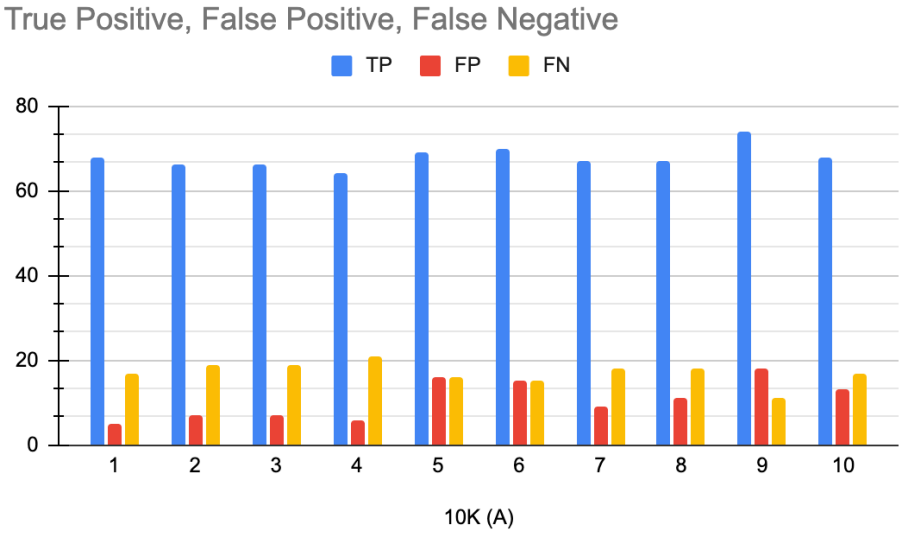
- Figure 56. Inference on validation dataset eTlZjK4VvmY with non-cumulative training (Type A) - 10K training steps model.

As can be seen in figure 56 the algaes are detected with a 0.99 level of confidence and the flipper with 0.61. Although this behaviour could be avoided with more pre-processing of the images to remove the algaes.

Another validation dataset that presenten some problems was 1dbyKDw27h8. In this case we had that during the execution and validation with the model with non-cumulative training (Type A) - 10K training steps the performance regresses quite a bit for multiple instances (iterations 7 and 10). The purple colours in the iterations row denotes the introduction of a training set of the same species. The more intense purple means that both the training sets were of blue whales and the plain purple that only one of the two training sets introduced was of a blue whale.

MS COCO Metrics / Iteration	0	1	2	3	4	5	6	7	8	9	10
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.432	0.438	0.444	0.448	0.449	0.460	0.510	0.450	0.472	0.514	0.499
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.691	0.712	0.713	0.722	0.736	0.757	0.818	0.769	0.785	0.829	0.776
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.413	0.469	0.525	0.516	0.530	0.500	0.606	0.443	0.533	0.554	0.529
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.432	0.438	0.444	0.448	0.449	0.460	0.510	0.450	0.473	0.514	0.499
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.365	0.352	0.345	0.354	0.366	0.379	0.387	0.359	0.352	0.382	0.388
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.513	0.505	0.513	0.516	0.532	0.549	0.579	0.552	0.549	0.598	0.575
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.522	0.516	0.516	0.518	0.533	0.558	0.584	0.562	0.561	0.613	0.581
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.522	0.516	0.516	0.518	0.533	0.558	0.584	0.562	0.561	0.613	0.581

- Table 26. MS COCO metrics for non-cumulative training (Type A) - 10K training steps validated on the dataset 1dbyKDw27h8.

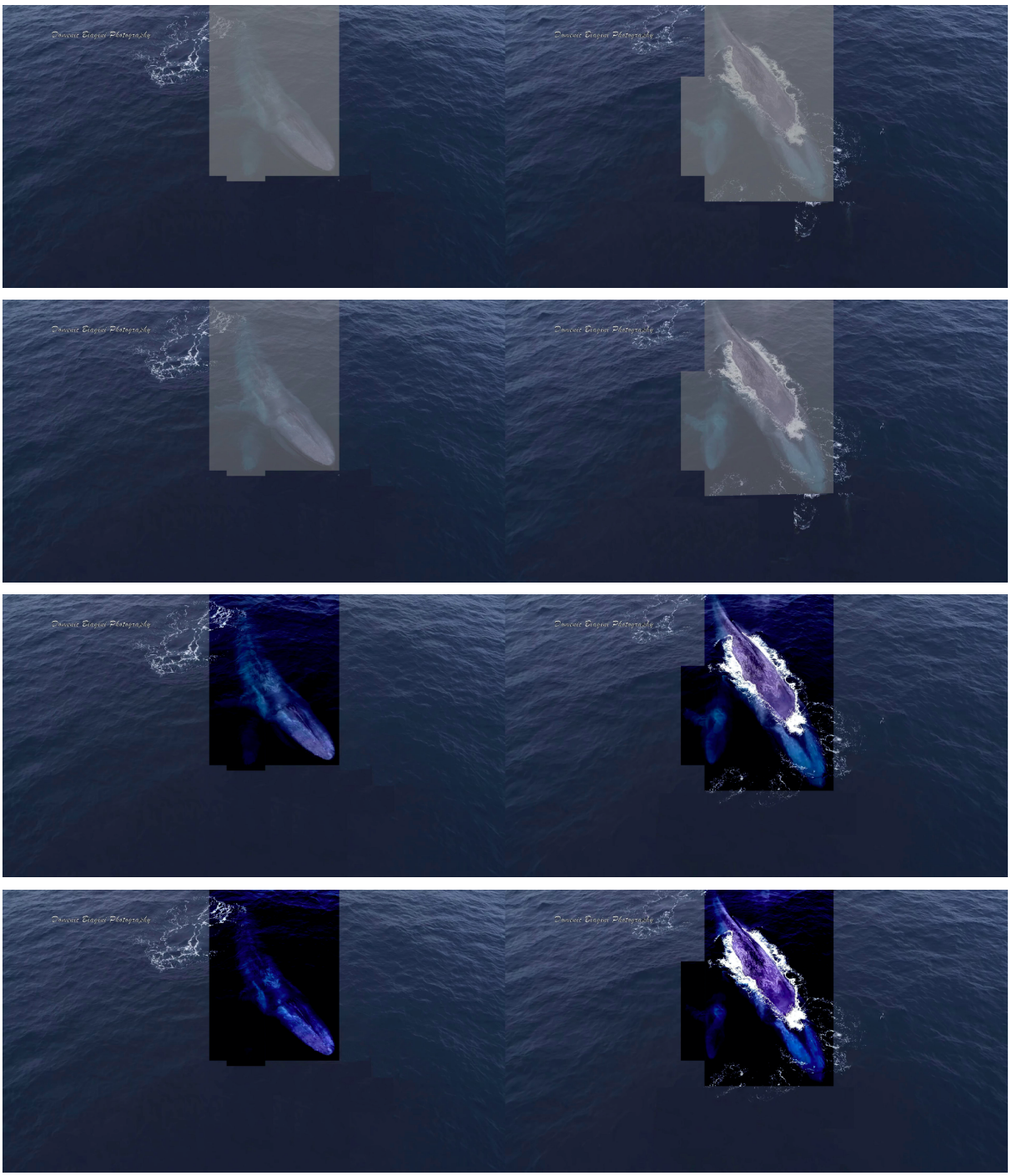


- Figure 57. Number of true positives, false positives and false negatives per iteration of the experiment non-cumulative training (Type A) - 10K training steps validated on the dataset 1dbyKDw27h8.

Taking into account that these were pictures in the open ocean it seemed strange that there were that many false positives again plus the false negatives remaining at a constant level.

- Figure 58. Inference on two images of the validation dataset 1dbyKDw27h8 with non-cumulative training (Type A) - 10K training steps model.

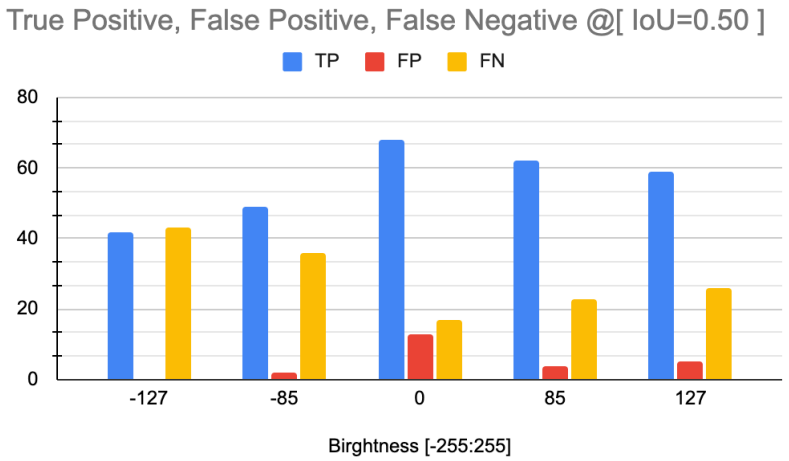
On the left image dolphins are being detected as whales and on the right one of the whales that remains underneath is not being detected. It is nonetheless good that the network counts the dolphin as whales as a very good generalization from the training data. But also inconsistent because the dolphins are not being detected in all the images in which they appear. Since this situation repeated itself across every iteration we decided to use it to cut the dolphins out of the images and apply contrast variations inside the bounding boxes. The contrast variations could be done in a range of [-255:255] where the value 0 would be how the image is as default. We altered every validation image by -⅔, -⅓, +⅓ and +⅔.



- Figure 59. From top to bottom, brightness changes of; -127, -85, +85 and +127 inside the bounding boxes.

MS COCO Metrics / Brightness change in range [-255:255]	-127	-85	0	85	127
Average Precision (AP) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.378	0.454	0.499	0.476	0.253
Average Precision (AP) @[IoU=0.50 | area= all | maxDets=100]	0.679	0.733	0.776	0.753	0.533
Average Precision (AP) @[IoU=0.75 | area= all | maxDets=100]	0.382	0.488	0.529	0.513	0.237
Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1	-1	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100]	-1	-1	-1.000	-1.000	-1.000
Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.379	0.454	0.499	0.476	0.253
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 1]	0.333	0.389	0.388	0.413	0.264
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets= 10]	0.467	0.532	0.575	0.591	0.406
Average Recall (AR) @[IoU=0.50:0.95 | area= all | maxDets=100]	0.511	0.559	0.581	0.594	0.486
Average Recall (AR) @[IoU=0.50:0.95 | area= small | maxDets=100]	-1	-1	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100]	-1	-1	-1.000	-1.000	-1.000
Average Recall (AR) @[IoU=0.50:0.95 | area= large | maxDets=100]	0.511	0.559	0.581	0.594	0.486

- Table 27. MS COCO metrics for non-cumulative training (Type A) - 10K training steps validated on the brightness variated datasets of 1dbyKDw27h8.



- Figure 60. Number of true positives, false positives and false negatives per experiment for non-cumulative training (Type A) - 10K training steps validated on the brightness variated datasets of 1dbyKDw27h8.

On table 27 the results don't improve in comparison to the validation done on the original dataset. All four types of brightness change produce more false negatives meaning that the number of bounding boxes not being detected continues to increase. On the other hand, one aspect that does improve is the number of false positives seen in figure 60 which decreases after cropping out the dolphins. Meaning that it's worth the cost of applying a pre-process of the images.

7.8. R^2 prediction

R^2 is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model. It explains to what extent the variance of one variable explains the variance of the second variable.

For this experiment we have obtained the different R^2 values for every possible model. This will help us get an idea on how many iterations would it take to achieve a specific target for a 0.90 mAP value. For every model the projections are done following a lineal, 2nd and 3rd degree polynomial. The values shown in the table are the number of iterations that it would take so the model can achieve 0.50 and 0.90 mAP.

Type	x^3	x^2	x^1	x^0	R^2	0.50	0.90
Lineal			6.45E-03	4.52E-01	0.716	7.42	69.39
2nd Polynomial		2.56E-04	3.89E-03	4.56E-01	0.725	7.56	34.71
3rd Polynomial	1.66E-04	-2.23E-03	1.34E-02	4.50E-01	0.752	8.05	17.59

- Table 28. R^2 prediction for 3K training steps of non-cumulative training.

Type	x^3	x^2	x^1	x^0	R^2	0.50	0.90
Lineal			9.19E-03	4.47E-01	0.970	5.75	49.27
2nd Polynomial		-8.74E-05	1.01E-02	4.46E-01	0.971	5.66	-
3rd Polynomial	2.74E-05	-4.98E-04	0.0116	0.445	0.972	5.69	26.20

- Table 29. R^2 prediction for 10K training steps of non-cumulative training.

Type	x^3	x^2	x^1	x^0	R^2	0.50	0.90
Lineal			6.85E-03	4.49E-01	0.795	7.43	65.86
2nd Polynomial		2.58E-04	4.27E-03	4.53E-01	0.804	5.661	34.19
3rd Polynomial	1.96E-04	-2.68E-03	1.55E-02	4.46E-01	0.840	8.110	17.02

- Table 30. R^2 prediction for 30K training steps of non-cumulative training.

Type	x^3	x^2	x^1	x^0	R^2	0.50	0.90
Lineal			7.99E-03	4.57E-01	0.875	5.33	55.39
2nd Polynomial		-6.00E-04	1.40E-02	4.48E-01	0.913	4.59	36.49
3rd Polynomial	-3.32E-05	-1.02E-04	1.21E-02	4.50E-01	0.914	4.57	-

- Table 31. R^2 prediction for 3K training steps of cumulative training.

Type	x^3	x^2	x^1	x^0	R^2	0.50	0.90
Lineal			4.22E-03	4.59E-01	0.678	9.70	104.53
2nd Polynomial		-1.33E-04	5.55E-03	4.57E-01	0.684	10.25	74.70
3rd Polynomial	9.89E-05	-1.62E-03	1.12E-02	4.54E-01	0.704	9.63	21.12

- Table 32. R^2 prediction for 10K training steps of cumulative training.

Type	x^3	x^2	x^1	x^0	R^2	0.50	0.90
Lineal			2.75E-03	4.49E-01	0.238	18.74	164.44
2nd Polynomial		8.11E-04	-5.37E-03	4.61E-01	0.400	10.252	26.81
3rd Polynomial	1.20E-04	-9.87E-04	1.49E-03	4.56E-01	0.426	10.410	18.43

- Table 33. R^2 prediction for 30K training steps of cumulative training.

Both tables X+17 and X+19 have a '-' value for 0.90 because certain polynomials give a negative number of iterations which are not valid.

So, to achieve 0.90 mAP with the lineal projections would require:

Method	Iterations	Training time (hours)	Training images
Type A - 3K training steps	69.39	36.45	6,939
Type A - 10K training steps	49.27	86.22	4,927
Type A - 30K training steps	65.86	345.76	6,586
Type B - 3K training steps	55.39	29.09	5,539
Type B - 10K training steps	104.53	182.92	10,453
Type B - 30K training steps	164.44	863.31	16,444

- Table 34. Lineal projections to reach 0.90 mAP per method.

The results show that the highest mAP of the type A - 10K training steps method has had its costs in the time. If this project was focused only in optimizing the time dedicated to creating the dataset then the best option would continue to be the type A - 10K training steps method because it requires 612 fewer images than any other method. But if the main goal was to reduce the training time and energy consumption then the type B - 3K training steps would have been the best choice since it would have required a third or the training time.

8. Conclusions

First, by investigating different kinds of neural networks, machine learning techniques and seeing an up-and-running network previously developed this project has allowed me to deepen my theoretical knowledge of the machine learning field. The theoretical explanation is focused on computer vision where we can see a description of the state of the art, the usual structure of the one-stage and two-stage object detectors and the usual metrics used on computer vision.

Second, we have built a complete dataset of whales from scratch. The training dataset is composed of 1438 ground-truth boxes (21% increased from Rubén's original training dataset) from 1000 drone images (23% increase) of 20 videos. This helps stakeholders such as other students that continue to develop their own networks and benefit from having more labelled data available or surveyors. Additionally, we have learned about the optimization done to the Faster R-CNN Inception-ResNet V2 architecture.

From what we saw on section 7.:

- The pre-trained model stored in the server and used by Rubén as the starting checkpoint was the more robust among the three available models. Also having the already trained model on novelty dataset as a viable option since it obtained very similar metrics.
- The non-cumulative training (Type A) - 10K training steps method proved to be the one that got better AP and AR out of all the experiments.
- A lot of false positives were predicted on non-whale images even though in an ideal situation none should have been predicted.
- In section 7.5. we saw the difference between the MS COCO metrics and mannually calculating the AP for IoU equal to 0.5. And how the type A obtains better COCO metrics even though its false positives numbers are higher.
- Finally we estimated the amount of training that it would take to achieve an AP of 0.90. The lowest amount of time being 29.09 hours for the Type B - 3K training steps model with 5539 images. And for the lowest amount of images needed being 4927 for the Type A - 10K training steps model requiring 86.22 hours of training time.

As for ours non-cumulative training (type A) and cumulative training (type B) experiments we have seen how the results vary a lot:

Architecture	Final mAP	Training time (hours)
Type A - 3K training steps	0.521	5.25
Type A - 10K training steps	0.538	17.51
Type A - 30K training steps	0.521	52.53
Type B - 3K training steps	0.525	5.25
Type B - 10K training steps	0.496	17.51
Type B - 30K training steps	0.484	52.53

- Table 35. Final mAP and training time for every architecture.

The best performance comes from the type A - 10K training steps. The second one is the type B - 3K training steps. The difference between them for mAP is only 0.013 which represents a 2 per cent improvement. But the most important factor that I see is the difference in time to obtain the best performance which is 12.26 hours for this particular case. Taking into account that this is only with 1000 images it is a considerable amount of time. If this is scaled to fit the purposes of a company that wants to put a product out and it requires several magnitudes more of images then the training time could be unattainable. Making it worth going for the cumulative method that requires less time (around a third for this project), less energy consumption and just short of 500 more images. As long as the difference in mAP is acceptable. On the other hand, if none of these concerns are valid and the projected training times seen in section 7.8. are acceptable then the best choice is again the type A - 10K training steps.

Both sections 7.4. and 7.7. have shown us that the network is still susceptible to detecting objects as whales that are not really whales. Such as algaes and dolphins which reflect the importance of applying pre-processing to the images. Investing resources to do such a task will pay off in the final performance and it seems like it's an unavoidable thing to do. Since the presence of algaes, other mammals like seals and dolphins, large fishes like sharks or even boats and ships cannot be easily avoided when taking aerial or satellite images.

As my co-director Ludwig Houegnigan explained to me, the data augmentation experiment should have been done differently via contrast changes in order to try to obtain better results. This experiment could be repeated in order to properly alter contrast and brightness to obtain conclusive results.

Another point made with Ludwig's help is that Resnet, Inception and other architectures that have been used for some studies on novelty datasets will not always be the best feature extractors for all other kinds of datasets. Resnet and Inception, in their various forms, are actually very large networks that do not re-train well on small datasets because one cannot update all the parameters efficiently.

The main objective of the project was to get my hands on a running project and try to further develop it by increasing the data and experimenting with it. We have demonstrated how the different methods affect the results and most importantly how scaling it up affects the mAP and the costs.

Therefore, the project is a good starting point for other students that want to continue developing it or who want to have their first contact with a functioning machine learning project. We have proved that by continuing to expand the datasets and applying the proper techniques a better model can be achieved. Finally, this project provides good datasets of whale images that can be used for next projects.

Bibliography

[1] Iwc.int. 2021. Population Estimates. [online] Available at: <https://iwc.int/estimate>.

[2] Cooke, J., 2021. IUCN Red List of Threatened Species: Eubalaena glacialis. IUCN Red List of Threatened Species. [online] Available at: <https://www.iucnredlist.org/species/41712/50380891>.

[3] Bigpixel.ucsd.edu. 2021. About. [online] Available at: <https://bigpixel.ucsd.edu/about/index.html>.

[4] Martín Pinardel, R., 2020. AUTOMATIC DETECTION OF ENDANGERED SPECIES IN VIDEO AND SATELLITE IMAGES USING DEEP LEARNING. Universitat Politècnica de Catalunya.

[5] E. Guirado, S. Tabik, M. L. Rivas, D. Alcaraz-Segura, F. Herrera. “Whale counting in satellite and aerial images with deep learning”. Sci Rep 9, 14259, October 2019. DOI: https://doi.org/10.1038/s41598-019-50795-9.

[6] Sueldo: IT Project Manager en Barcelona | Glassdoor. [online] Available at:<https://www.glassdoor.es/Sueldos/barcelona-it-project-manager-sueldo-SRCH_IL.0,9_IM1015_KO10,28.htm?countryRedirect=true> [Accessed October 2020].

[7] Data Scientist Salaries in Spain - Glassdoor. [online] Available at: <https://www.glassdoor.es/Sueldos/spain-junior-researcher-sueldo-SRCH_IL.0,5_IN219_KO6,23.htm?countryRedirect=true> [Accessed October 2020].

[8] Salary: Software Engineer in Barcelona, Spain | Glassdoor. [online] Available at: <https://www.glassdoor.es/Sueldos/barcelona-junior-developer-sueldo-SRCH_IL.0,9_IM1015_KO10,26.htm?countryRedirect=true> [Accessed October 2020].

[9] Average Software Tester Salary in Spain. PayScale. [online] Available at: <https://www.payscale.com/research/ES/Job=Software_Tester/Salary/06ccb152/Barcelona> [Accessed October 2020].

[10] Sueldo: Analyst en Barcelona | Glassdoor. [online] Available at: <https://www.glassdoor.es/Sueldos/barcelona-analyst-sueldo-SRCH_IL.0,9_IM1015_KO10,17.htm?countryRedirect=true> [Accessed October 2020].

[11] Ayon Dey. “Machine Learning Algorithms: A Review”. 2016. [online] Available at: <http://ijcsit.com/docs/Volume%207/vol7issue3/ijcsit2016070332.pdf>.

[12] Pierre Lison. “Language Technology Group (LTG) Department of Informatics. An introduction to machine learning”. 2012. [online] Available at: <https://www.nr.no/~plison/pdfs/talks/machinelearning.pdf>.

[13] Medium. 2021. Activation Functions (Linear/Non-linear) in Deep Learning—ReLU/Sigmoid/SoftMax/Swish/Leaky ReLu.. [online] Available at: <https://xzz201920.medium.com/activation-functions-linear-non-linear-in-deep-learning-relu-sigmoid-softmax-swish-leaky-relu-a6333be712ea>.

[14] Medium. 2021. Activation Functions in Neural Networks. [online] Available at: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>.

[15] Juergen Schmidhuber: “Deep Learning in Neural Networks: An Overview”, 2014, Neural Networks, Vol 61, pp 85-117, Jan 2015; [http://arxiv.org/abs/1404.7828 arXiv:1404.7828]. DOI: [https://dx.doi.org/10.1016/j.neunet.2014.09.003 10.1016/j.neunet.2014.09.003].

[16] ResearchGate. 2021. (PDF) A COMPREHENSIVE REVIEW FOR ARTIFICAL NEURAL NETWORK APPLICATION TO PUBLIC TRANSPORTATION. [online] Available at: <https://www.researchgate.net/publication/315111480_A_COMPREHENSIVE_REVIEW_FOR_ARTIFICAL_NEURAL_NETWORK_APPLICATION_TO_PUBLIC_TRANSPORTATION>.

[17] Dao, Vu & Vemuri, Rao. “A performance comparison of different back propagation neural networks methods in computer network intrusion detection. Differential Equations and Dynamical Systems”. 2002. [online] Available at: <https://www.researchgate.net/publication/228902789_A_performance_comparison_of_different_back_propagation_neural_networks_methods_in_computer_network_intrusion_detection>.

[18] Bhattarai, S., 2021. “What is Gradient Descent in machine learning?”. [online] A Tech Blog. Available at: <https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/>.

[19] Brownlee, J., 2021. “Difference Between Backpropagation and Stochastic Gradient Descent. Machine Learning Mastery”. [online] Available at: <https://machinelearningmastery.com/difference-between-backpropagation-and-stochastic-gradient-descent/>.

[20] Ian Goodfellow, Yoshua Bengio, Aaron Courville. “Deep Learning (Adaptive Computation and Machine Learning series)”. Page 204. 2016.

[21] Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. “Learning representations by back-propagating errors”. October 1986. [online] Available at: <https://ui.adsabs.harvard.edu/abs/1986Natur.323..533R

[22] Sutskever, Ilya; Martens, James; Dahl, George; Hinton, Geoffrey E. “On the importance of initialization and momentum in deep learning”. June 2013. [online] Available at: <http://www.cs.utoronto.ca/~ilya/pubs/2013/1051_2.pdf>

[23] Sutskever, Ilya. “Training recurrent neural networks”. 2013. [online] Available at: <http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf>.

[24] Ieeexplore.ieee.org. 2021. Learning Complex, Extended Sequences Using the Principle of History Compression. [online] Available at: <https://ieeexplore.ieee.org/document/6795261>.

[25] A. Mikolajczyk, M. Grochowski. “Data augmentation for improving deep learning in image classification problem”. International Interdisciplinary PhD Workshop. May 2018. DOI: https://doi.org/10.1109/IIPHDW.2018.8388338

[26] D2l.ai. 2021. 6.3. Padding and Stride — Dive into Deep Learning 0.16.2 documentation. [online] Available at: <https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html>.

[27] Bazaga, A., Roldán, M., Badosa, C., Jiménez-Mallebrera, C. and Porta, J., 2021. “A Convolutional Neural Network for the Automatic Diagnosis of Collagen VI related Muscular Dystrophies”. [online] Available at: <https://arxiv.org/abs/1901.11074>.

[28] Medium. 2021. Annotated RPN, ROI Pooling and ROI Align. [online] Available at: <https://medium.com/swlh/annotated-rpn-roi-pooling-and-roi-align-6a40ac5bbe1b>.

[29] P. Soviany, R. T. Ionescu. “Optimizing the Trade-off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction”. August 2018. ArXiv: 1803.08707v3.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. “Going deeper with convolutions”. September 2014. ArXiv: 1409.4842v1.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning” August 2016. ArXiv: 1602.07261v2.

[32] Medium. 2021. A Simple Guide to the Versions of the Inception Network. [online] Available at: <https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202> [Accessed 2021].

[33] S. Ren, K. He, R. Girshick, J. Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. January 2016. ArXiv: 1506.01497v3.

[34] Medium. 2021. On Object Detection Metrics With Worked Example. [online] Available at: <https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e>.

[35] Hosang, J., Benenson, R. and Schiele, B., 2021. Learning Non-Maximum Suppression. [online] Openaccess.thecvf.com. Available at: <https://openaccess.thecvf.com/content_cvpr_2017/html/Hosang_Learning_Non-Maximum_Suppression_CVPR_2017_paper.html>.

[36] Cha, Y., Choi, W., Suh, G., Mahmoudkhani, S. and Büyüköztürk, O., 2021. Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple

Damage Types. [online] Semanticscholar.org. Available at: <https://www.semanticscholar.org/paper/Autonomous-Structural-Visual-Inspection-Using-Deep-Cha-Choi/15a4e2a15f20ed77609a70fb268cbcfafa21df54>.

[37] Cocodataset.org. 2021. COCO - Common Objects in Context. [online] Available at: <https://cocodataset.org/#detection-eval>.

[38] Zeng, N., 2021. An Introduction to Evaluation Metrics for Object Detection | NickZeng|曾广宇. [online] NickZeng|曾广宇. Available at: <https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection>.

[39] TensorFlow. 2021. TFRecord and tf.train.Example | TensorFlow Core. [online] Available at: <https://www.tensorflow.org/tutorials/load_data/tfrecord>.

[40] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, Rong Qu: “A Survey of Deep Learning-based Object Detection”, 05 September 2019; [http://arxiv.org/abs/1907.09408 arXiv:1907.09408].

[41] GitHub. 2021. tensorflow/models. [online] Available at: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md>.

[42] TensorFlow. 2021. TensorBoard | TensorFlow. [online] Available at: <https://www.tensorflow.org/tensorboard>.

[43] GitHub. 2021. fizyr/keras-retinanet. [online] Available at: <https://github.com/fizyr/keras-retinanet/> [Accessed 20 April 2021].

[44] Glove and Boots - All Together Now. (2013, November 5). [Video]. YouTube. https://www.youtube.com/watch?v=NJRUo1zmIQ4