Deep on Bogotá's streets

Esteban Galvis Gutiérrez
Advisors: José Tiberio Hernández, Pablo Arbeláez

A thesis presented for the undergraduate degree of
Systems and Computer Engineer

Department of Systems and Computer Engineering
Universidad de Los Andes
Bogotá, Colombia

December 2018

Acknowledgments

I want to dedicate this work to my parents who have given all they have for me and my brothers and sister. Hopefully I can make them proud by doing meaningful work.

To my girlfriend Daniela because she has had to live with me these pasts years and has endured my hard process of studying while working.

To José Tiberio Hernández and Pablo Arbeláez for believing in me to do such work. I have learned a lot from the both of them and hope to carry on their teachings wherever I go.

Abstract

Technology has played an extremely important role in society during the last decades. It has brought great benefits to humanity creating some tools that have automatized processes—saving men time and effort of work. One situation where this occurs is with the video cam footage that cities produce each day using its traffic cameras. Each instant, these cameras are recording information about the cities' activities which can convey relevant information about topics of a district's agenda. Yet, making a person analyze hundreds of hours of video to extract useful information for decision making, is unfeasible. The following work documents the use of computer vision to detect a variety of classes in Bogotá by using its traffic cameras to derive insights to influence local transportation government decision making. As a reference, computer vision machine learned models from a distribution of data of a city to be considered from the first world, opposed to Bogotá, is used to compare such models in a variety of urban developments.

Contents

2. Methodology

2.1. Database

2.1.1. Dataset Proposal: Bogotá

2.1.1.1. Video Selection

2.1.2. Dataset MIO-TCD: Canada

2.2. Methods

2.3. Experiments

2.4. Results

2.4.1. MIO-TCD

2.4.2. Bogota

2.5. Conclusion

3. Appendix

3.1. Experiments Repository

Chapter 1

Preliminaries

In this chapter we introduce the main ideas of the project and its objectives. The reader will be familiarize with all the concepts that the project states and implements. Additionally, she/he will be able to understand how a real world problem from a specific user can be tackle with artificial intelligence.

1.1 Introduction

Artificial intelligence has its first trail and foundation in Philosophy at the moment Plato asks Euthyphro for an algorithm to be able to discriminate between piety and non piety [4]. To make of this tale an example to better understand our case, Plato is an user that is asking Euthyphro (us) for an AI system to discriminate between some types of object classes that it sees from video images that the user has.

1.2 Problem

First of all, let's ask ourselves why would someone want an AI system to understand video? What use can it have? The answer is too broad since usually AI systems are being deployed in every industry and for some, like the pioneer artificial intelligence research scientist Andrew Ng, they are the new electricity. Usually these types of systems work great in tasks that are repetitive and expensive to perform for humans. Businesses and a lot of situations tend to have a lot of these tasks, which makes them a perfect fit to be tackled by AI systems.

1.2.1 Cities

A city is a complex system composed of several entities that interact between each other like: Streets, Neighborhoods, Economies, Citizens, and Politics just to name a few, are part of the ecosystem. Additionally, keeping track of such interactions still remains a key issue for local government in order to have better decision making.

1.2.1.1 Citi Mobility

One problem that a city has daily is the difficulty of observing its behavior in terms of mobility. Depending on the size of the city, the city has a flux of thousands or millions of vehicles and pedestrians each day and knowing how

good or bad each one of them is commuting, is nearly impossible nor viable.

Nonetheless, the cities in general are increasingly making larger efforts to obtain information on the behavior of the mobility of its traffic, or of how their road networks are used. For example, Bogotá, the capital city of Colombia, partnered with the mobile application Waze through its initiative Connected Citizens. [6] For the reader unfamiliar with Waze: it is a navigation app that shows a driver the traffic conditions in real time of where the person is currently located. This type of technology is enabled thanks to a huge communal effort since it to work, all the cities' commuters have to have the app installed and running.

This type of situation is perfect for a city's government since the app stores a vast information about its vehicular flow that the district currently lacks. If a city instantly knows when an accident happens, it can react to it quickly and mitigate any traffic problems it might cause. Also, it can help understand quantitatively recurrent traffic bottlenecks as well as areas that are not safe: important topics around urban planning. It helps plan traffic lights, traffic signaling and validate current transportation policies to name a few. Similar initiatives exist like Uber's Uber Movement [2] that wants to sell the data they have collected through their trips to third parties, different from Waze's which is free.

Broadly speaking, a city to improve its flow it is mandatory for it to optimize the infrastructure resources it has at hand the most. In order for this to happen, it is required to extract and store information in real time. Fortunately, we're in a digital age where real time is at the tip of our thumbs like in the mobile applications mentioned before. However, there are still traditional and mundane ways of acquiring real time data. According to [9]

Colombia's cities are collecting information about traffic manualling and this data is being used for road design and implementing local traffic norms. However, it also mentions that there are methods based in AI systems to automatically extract information based on video footage of the city they are analyzing.

1.2.1.2 Not Every City Is Alike

The behavior of a city is usually related to the behavior of its citizens. After all, they're the ones that use its infrastructure. Thus, it is no surprise to find that Mumbai functions different than Copenhagen, and will be crazy to state that New York City works the same way as Nairobi or even that Bogotá and Cartagena operate similar; each population has a different culture, a way of communicating and customs than reflect on how they behave each day with their peers and city. Consequently, the manner how vehicles and pedestrians interact with roads, traffic signs and traffic lights will vary accordingly. Can AI systems capture these behavior variabilities?

1.2.1.3 Business

Many of the efforts from local governments to collect information and therefore have better urban planning cost resources. Bogotá, in December 2015, invested around $10 US million dollars in order to deploy traffic cameras and sensors to gather data about traffic. [1] Normally, these type of efforts translate to city contracts for businesses which mean that they ultimately come from a restricted budget created from taxpayer money. Ultimately, businesses that manage to solve these issues a city might have,

will be rewarded. Moreover, the resources invested to these efforts should be directed to initiatives that will make use of these the best and smartest way. An opposite example of this will be for instance, cities placing public tenders for contracts to manually count vehicles or events at road intersections. It is not a scalable solution, it requires a lot of time and it is expensive.

1.2.2 Computer Vision

Now that it was discussed the difficulty and need of keeping track of the behavior of a city in terms of mobility let's dwell first with how can an AI system understand video in order to do so.

First proposed in 1975 by a group of scientists in California, the concept of artificial intelligence is based on a series of rules to program algorithms which allow computers to think and maximize their chances of success in any task. According to Professor Geoffrey Hinton—known as the godfather of modern artificial intelligence— AI is configuring a machine to imitate the human mind. [10] Yet, not so long ago did this concept become part of our reality due to great developments in computing power, more precisely thanks to the use of GPUs to train Deep Learning based models. [11] For our particular task, it is necessary that a computer mimics the human brain in regards perception to understand what it sees through plain digital images. This type of problem is related to computer vision, a sub field of artificial intelligence that seeks to emulate many of the capabilities of human vision.

1.2.2.1 Introduction to Computer Vision

In consonance, computers need to interpret and understand raw input images. Humans do it naturally, but for computers to do it is a complete challenge. These machines are built with bolts, metals and digital circuits that manage to understand no more than simple bits packed into bytes. Therefore, it is important for any image to be layout in a language native to machines.

An image is the output of an image formation process that takes as input some lighting conditions, scene geometry, surface properties and camera optics. [12] Thus, it is important for computers to be able to capture these properties, if they want to comprehend what's in an image. We have specialized fields like Physics and Mathematics to help us understand these parameters in real life through mathematical modelling. Luckily, computers are really good at computing things, which means that it if these parameters are mathematically modeled, they can be processed by them. Normally, the image formation process can either be analog or digital thanks to technological breakthroughs over time in relation to the capacity of sensors to capture light.

Thanks to these breakthroughs, we are able to enjoy of digital images that reflect real life conditions with a fixed resolution. Sensors, or more generally speaking cameras, are able to capture real life scene elements that map to a pixel -- also known as a picture element. Depending on the technical specification of each camera sensor, more or less information is captured per pixel. For instance, each picture element stores the brightness of the environment it captures in a range between [0-255] simply because the

maximum amount of information a byte can store is 8 bits, which translates to 255.

A picture can be rendered in several ways, an image rendered only with its brightness is known as a grayscale image. Also, a picture can be rendered to show the color it portrays. There are several color spaces which can be rendered with an image: CIE, RGB, XYZ and L*a*b. [12] RGB is a common color space, which stores the respective color information in terms of the amount of red, green or blue a pixel has. If a computer can only store information in 8 bits, how much red, green or blue a pixel is allowed to have? As well as with the brightness of a pixel, a pixel can represent its color in a range between [0-255] for each color component; obeying the natural restriction of computers stated before.

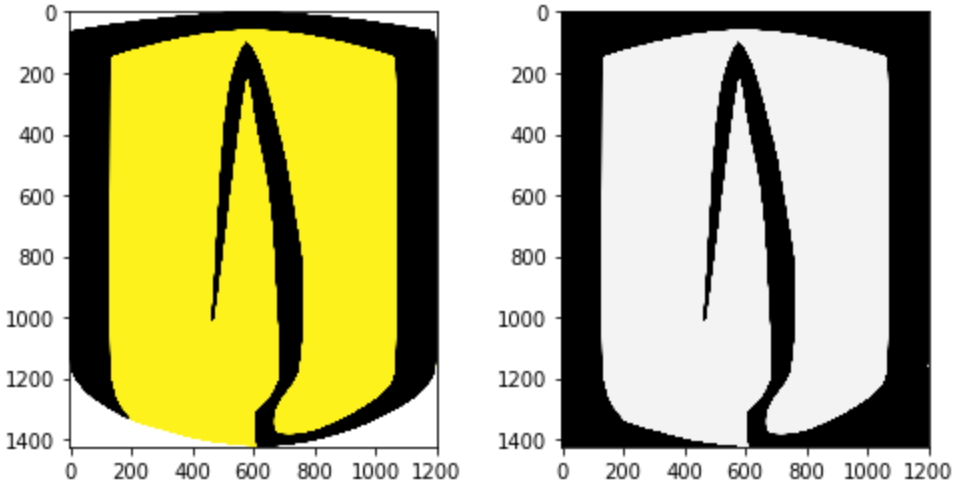

Figure 1: color RGB image (left), grayscale Image (right)

The amount of color and brightness in an image gives important cues of what is present in an image. Yet, if an image has a round red object, one might think it can be an apple, or is it a tomato? It is still confusing for computers to identify objects with only such cues. However, there are other cues that can describe an image, for instance: textons. This cues are based on

the textures of an image, and these are extracted by a pipeline described in Leung, T. and Malik, J [15].

1.2.2.2 Detection

Further methods developed for Computer Vision rely heavily in hand-features like textons. Thanks to these features or cues, Computer Vision is able to solve several problems related to recognition, detection and segmentation to name a few. The following work in this paper is a detection problem; it is of our interest to detect a class, and count how many instances of such class are present in a video frame.

This area has developed quite impressive in the last decades and is important to understand how it has occured before moving on. In 2001, Viola & Jones [13] presented a novel face detector that used a sliding window paradigm. The idea behind it is that a sliding window looks at all the image seeing if some features similar to Haar features [14] are present at each position of the window. However, the Haar like features output is sparse because there is a lot of redundant information calculated by such framework. A boosted classifier AdaBoost [15] is used to classify these features: eliminating redundancy and classifying if the image is a face or not.

With limited computing resources and infrastructure in the early 21st century, the hand designed features combined with the sliding window paradigm created by computer vision scientist were of extreme importance. Navneet Dalal and Bill Triggs from Inria in 2005 proposed a hand feature algorithm called Histograms of Oriented Gradients for the task of detecting humans [17]. The basic idea is to split the image into bins and in each bin calculate the gradients, or edge directions and bundle them into histograms to represent the block. This strategy allows to capture structural cues of the shape of the object at hand while being invariant to translation and rotation

to geometric and photometric transformations. After these features have been extracted, they are fed as input to a Linear Support Vector Machine [18] to be classified.

Using almost the same HOG features, Felzenszwalb, Ross Girshick et al in 2010 proposed a model that has these set of features into a deformable model that defines a spring-like connection between a pair of filters. These connections have as a reference a filter that will act as a root of a star graph shaped joining the deformable parts. Once the deformable parts are detected, these deformable parts will be fed to a Latent Support Vector machine for classification in order to detect an object [29].

These methods are just a brief overview of detection algorithms developed through the progression of this field. However, these methods differ greatly from current methods by not having a neural net that learns the filters that are used as feature extractors.

1.3 User

The user of this project is the Mobility District Secretary of Bogotá. However, this user could be easily generalized to any local city transportation government entity. The analytical task identified with the user corresponds in gathering data regarding the vehicle flow density through time. The main challenge in this task is how to extract this data and with it, take data-driven decisions regarding topics of safety and signaling.

The following works focuses on searching for tools to extract and produce this type of data from video cam footage of the city. In other words, we are only concentrated with the task of identifying the input of data, but not in the

decision making part of the task. Once there is a robust method to extract data, our user will be responsible of the decision making.

1.4 Related work

To be successful in this task, it is important to detect and classify correctly. Traditional computer vision methods have in some way been considered outdated since the 2012 outbreak of neural networks [3] in ImageNet compared to traditional methods.

1.4.1 Detection

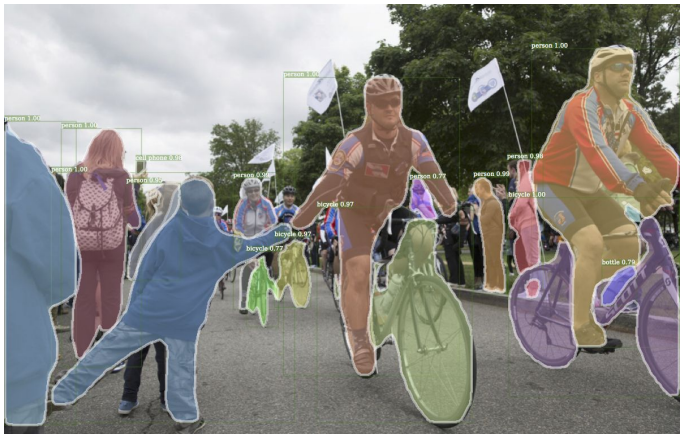

Figure 2 : An example of Mask R-CNN output

To understand where the field of computer vision is at object detection, it is recommended to look at the COCO [5] dataset, which contains 200,000 labeled bounding boxes for different objects. The top result for the 2017 challenge went to Megvii's MegDet [19] following Megvii's competitor Sensetime's in collaboration with The Chinese University of Hong Kong [20]. The two use similar implementations related to the framework Mask R-CNN [21] but improving some details. See figure for an example

It is important to note that this framework is used for the computer vision task of segmentation. Its goal is to output a mask for each object instance identified in an image. Also, it is based in a deep learning-based object detector called Faster R-CNN [22] which is considered to be a two stage detector: first it extracts some region of interests (ROI) that contain objects in an image, then after these regions have been identified, a per region operation is done to classify each ROI to a specific category.

Additionally, there are one stage detectors that are inspired by the sliding window paradigm used to detect objects before 2012. Mainly there are two popular methods: SSD [23] and YOLO [24] which provide a fast object detector by the expense of accuracy. The decrease in accuracy is due to the fact that these object detectors have a lot of region of interests extracted, where most of these regions belong to the background, meaning they don't contain objects. This leads to a problem when a model is trained since it represents an unbalanced class between objects present and background noise. The two stage methods are usually more accurate but slower making them a bottleneck for real time applications unless there is a proper infrastructure to support them. To a see a comparison between these methods see [25]

1.4.2 Traffic camera vehicle detection

Besides detecting, it is also necessary to classify correctly the regions extracted by the bounding boxes in relation to this research paper's goal. Even though COCO has classes related to pedestrians, vehicles and bicycles, these are general classes to our problem. Therefore, this paper is going to have as an additional reference the dataset MIO-TCD published for a CVPR 2017 workshop [26]. This dataset contains 137,743 labeled images by 200 different people for the Localization challenge. In total there are 11

categories: Articulated truck, Bicycle, Bus, Car, Motorcycle, Motorized vehicle, Non-motorized vehicle, Pedestrian, Pickup truck, Single unit truck, Work van. See figure 2 for an example the categories. The three top methods [27][23][28] used for detection differs from the top methods in COCO.

Figure 3: Example of the classes of MIO-TCD dataset

1.5 Project Objectives

- Create a mini dataset for our Bogotá traffic cameras detection task.
- Detect different types of classes using such cameras
- Compare performance of computer vision detection models in the dataset
- Compare performance of computer vision detection models in two different contexts

1.6 Outline

Chapter I was dedicated to introduce the problem of this research problem as well as to give a brief introduction of computer vision in relation to the technical problem we are trying to tackle: Object detection.

Chapter II discusses the methodology implemented in the research problem:

- The dataset used
- The methods used
- The experiment details
- The results of the experiments

Chapter III describes the repository created for such project.

Chapter 2

Methodology

2.1 Database

It is important to see how computer vision detection models work in a LATAM context. Currently, we see this type of detection task in a workshop for CVPR 2017, nonetheless, is with data from Canada; an opposite context for the one of Colombia, Bogotá. However, since there is not a dataset for our local task, one will be created and proposed to quickly iterate over it with the computer vision models.

2.1.1 Dataset proposal: Bogotá

Two sets of data from Bogotá are used for this project:

- Set 1: 360 images from two videos for training and 180 images from a different video for testing.

- Set 2: 900 images from four videos for training and 360 images from a different video for testing.

Dataset´s classes

Car	Bus	Pedestrian	Bicycle	Motorcycle	Articulated truck
SUV	Taxi	Pickup truck	Single unit truck	Work van	

2.1.1.1 Video selection

To gather a good dataset, it is important to have a good distribution of data enough to capture variances in terms of brightness, place, and viewpoints to name a few. These type of situations are common problems in computer vision algorithms that need to be covered by the model proposed for any task. For these reasons these videos are proposed from two cameras of the city:

- Video 1: 360 frames

- Video 2: 180 frames

- Video 3: 180 frames

- Video 4: 180 frames

- Video 5: 360 frames

Most of the data available was from AUTONORTE, which is why the frames from video 5 will be used as the test set and the rest of the frames of the videos will be used for training. It is important to state that this distribution was thought for the data Set 2 domain. For the data Set 1 domain, video 2 is the test set and video 3 and 4 are the training set.To capture variance in intensities, the videos are from different timeframes and weather conditions. Also, video 1 is from another location which is used to capture variance in viewpoint and place. As it was stated, currently there isn't a way to evaluate methods in this dataset since there are no labeled images thus labeling is required. Oxford´s VGG Image Annotator Tool is used for labeling [30].

2.1.2 Dataset MIO-TCD: Canada

To achieve the objective of comparing the performance of models in different contexts, this dataset will be compared with ours. The categories from the proposed dataset are: car, suv, pedestrian, bus, work van, pickup truck, taxi, single unit truck and articulated truck. These categories are designed to be as similar as the one in MIO-TCD with the exception that this set doesn't have the classes suv and taxi, instead, they use motorized vehicle and non-motorized vehicle in their place. Note: the training set will be used as testing set since the test set doesn't have public ground truths.

2.2 Methods

Before moving on, it is important to clarify that the reason for the division between Set 1 and Set 2 is to divide the problem and iterate quickly due to the time consuming nature of training deep learning models and labeling. On the other hand, both detection models will be fine tuned from COCO pre-trained weights and evaluated accordingly.

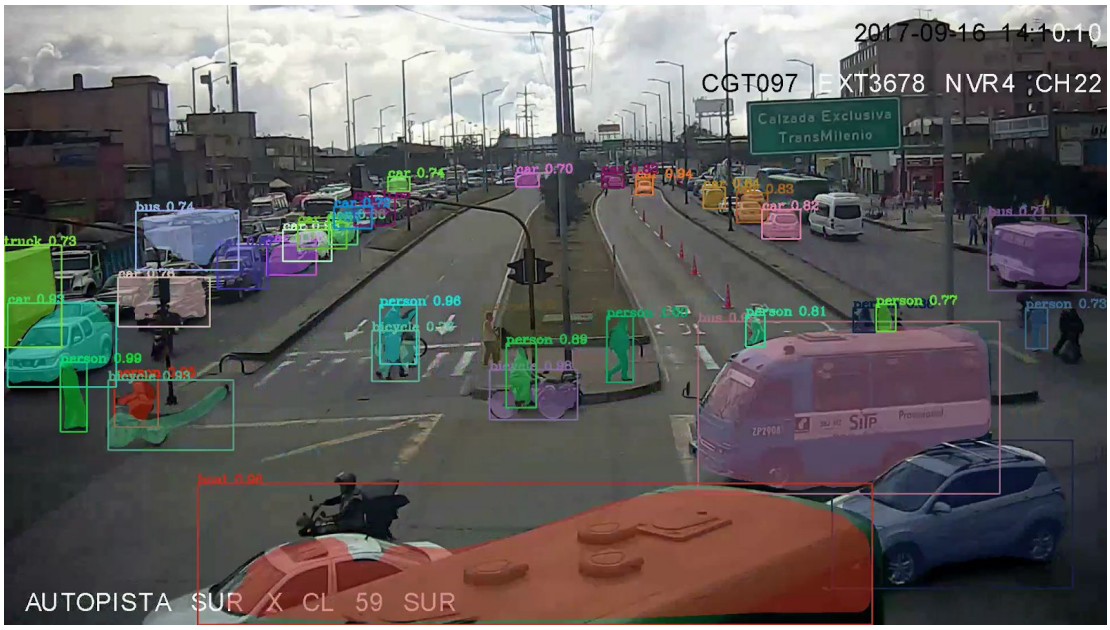

Figure 4: Example of Mask R-CNN [31] with COCO pre-trained weights from a camera of Bogotá

According to the state of the art of the COCO benchmark, the backbone object detector of Mask R-CNN, Faster R-CNN-will be used to compare it to current methods for localization of the MIO-TCD and our dataset. Also, a new one stage detector method RetinaNet [32] will be used to compare it with the two stage detector that Faster R-CNN is. The main advantage of this one stage detector is that it can detect densely while maintaining high accuracy as well as inference speed. The way it achieves this is by proposing a novel loss function call focal loss which attacks the unbalanced class problem mentioned before in the Related

Work section -- allowing it to detect 200k regions of interest while maintaining good accuracy and beating current state of the art two stage detectors.

2.3 Experiments

The implementation from [33] was used to train RetinaNet and the code from [34] was used for Faster R-CNN. For both Faster R-CNN and RetinaNet, a ResNet [35] architecture of 50 layers was used.

The experiments were run with the first set of data since it was the set that didn't have mistakes in the experimentation. While evaluating the second set, the results obtained were always with cero accuracy. The reason why set two didn't work was because of the labeling process, this process has to be very rigorously done and even though the labeling tool is robust, is still lacks user experience.

Taking into account that set one was used to train and evaluate, both models were initialized with COCO pre-trained weights before training to fine-tune the models. The reason of fine-tuning is that usually a neural network needs a lot of data to learn the representation of what you are tackling, with this in mind, our dataset is very small compared to the hundreds of thousands images usually used for training.

Training Faster-RCNN was a bottleneck for this research project since the original project used 8 gpus in its implementation. Unfortunately, do to infrastructure restrictions, this research project only counted with 1 gpu. Thus, training was done with a a batch size of 1 image per gpu for both Bogota's set 1 and MIO-TCD. The total iterations of both were of 1,440,000

steps and to fully train them took an estimate of 20 days. Due to infrastructure and time limitations, the whole training process couldn't be done. Taking into account the above, the training process was done for 8%-16% of the real training estimate. On the other hand, training RetinaNet was much faster and didn't have any problems regarding infrastructure. This is because this method is a one stage detector which makes it faster compared to the region based two stage detector Faster R-CNN.

2.4 Results

RetinaNet Res-50 MIO-TCD		
Class	Instances	AP
articulated_truck	9301	0.8904
bicycle	2260	0.8687
bus	10598	0.9716
car	233497	0.9476
motorcycle	1837	0.8847
motorized_vehicl	25845	0.4513
non-motorized_vehicle	2350	0.3593
pedestrian	7128	0.6733
pickup_truck	44283	0.9281
single_unit_truck	5741	0.6617
work_van	8709	0.7661
mAP 0.7639		

Faster R-CNN Res-50 MIO-TCD; 8.3% training		
Class	Instances	AP
articulated_truck	9301	0.082
bicycle	2260	0.082
bus	10598	0.143
car	233497	0.132
motorcycle	1837	0.097
motorized_vehicl	25845	0.016
non-motorized_vehicle	2350	0.009
pedestrian	7128	0.03
pickup_truck	44283	0.125
single_unit_truck	5741	0.033
work_van	8709	0.073
mAP 0.075		

RetinaNet Res-50 Sub-set Bogota; Epocs:1;		
Class	Instances	AP
articulated_truck	0	0
bicycle	0	0
bus	559	0.0125
car	277	0.0082
motorcycle	116	0.0017
SUV	183	0.0149
taxi	0	0
pedestrian	1	0
pickup_truck	50	0.0002
single_unit_truck	48	0.0247
work_van	28	0.192
mAP 0.0102		

Faster R-CNN Res-50 Bogota subset; 16% training		
Class	Instances	AP
articulated_truck	0	nan
bicycle	0	nan
bus	559	0.045
car	277	0.008
motorcycle	116	0.001
motorized_vehicl	183	0.01
non-motorized_vehicle	0	nan
pedestrian	1	0
pickup_truck	50	0.012
single_unit_truck	48	0.016
work_van	28	0.015
mAP 0.013		

If the results of Faster R-CNN in both MIO-TCD are projected to 100% they show promising results similar to RetinaNet. However, since the infrastructure needed for Faster is very restrictive, it is better to use

RetinaNet for this task.

RetinaNet was only train for one epoch due to the fact that it wasn't much data. This means that it will overfit to the small training data if train for longer. The main reason for this is because this subset doesn't capture all the variance present in the frames due to the limited amount of labeled data. In an ideal setting like MIO-TCD, we see that RetinaNet performed as well as the benchmark of the dataset which is 0.79 for the test set, yet, it is important to note that MIO's result was on the training set which means that the test set mAP should be lower. MIO's labeled test set wasn't available due to the fact that this was a competition in a CVPR 2017 workshop.

2.5 Conclusions

Taking into account that not everyone has 8 gpus, specially in a context like LATAM where it is expensive to have them and there is restricted funding, the recommended method to pursue the object detection task is RetinaNet. Also, a comparison between both Bogotá's and Canada's dataset can't be rigorously done due to the lack of labeled data for Bogotá. An important lesson for why it didn't work so well was the labeling process. This process has to been done close to perfect. For instance, if a bounding box is out of the frame, this will not default to a 0 in x or y but have a negative value which complicates the training and evaluation process. Nonetheless, this work is important for future work since it lays out the foundations for counting vehicles using traffic cameras.

Chapter 3

3.1 Experiments Repository

The following link is where the repository for this research project is stored

https://github.com/tebandesade/Deep_on_Bogotas_Streets

Faster R-CNN:
The new files in this work are:

- Detectron.pytorch/tools/
  - Detectron.pytorch/tools/infer_evaluation.py
  - Detectron.pytorch/tools/parse_location_dataset.py
  - Modified file Detectron.pytorch/tools/train_net_step.py
  - Modified file Detectron.pytorch/tools/test_net.py
- Detectron.pytorch/lib/datasets/
  - Detectron.pytorch/lib/datasets/mio_json_dataset_evaluator.py
  - Modified file Detectron.pytorch/lib/datasets/dummy_datasets.py
  - Modified file Detectron.pytorch/lib/datasets/task_evaluation.py
  - Modified file Detectron.pytorch/lib/datasets/dataset_catalog.py
- Detectron.pytorch/Data/*

RetinaNet

The new files in this work are:

- keras-retinanet/
  - keras-retinanet/test_mio_images.py
  - keras-retinanet/convert_vgglbl_csv.py
- Modified files from keras-retinanet/keras_retinanet/
  - keras-retinanet/keras_retinanet/bin/train.py
  - keras-retinanet/keras_retinanet/bin/evaluate.py

- keras-retinanet/data/*

References

[1] S. (2015, December 23). Sistema inteligente de transporte, aporte de movilidad a la transformación de Bogotá en una Smart City. Retrieved from http://www.movilidadbogota.gov.co/web/sistema_inteligente_de_transporte_aporte_de_movilidad_a_la_transformacin_de_bogot_en_una_smart_city

[2] U. (n.d.). Busquemos caminos más inteligentes para avanzar. Retrieved from https://movement.uber.com/?lang=es-CO

[3] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[4] Stuart J. Russell and Peter Norvig. Artificial intelligence: A Modern Approach. Prentice Hall, 2009.

[5] T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in ECCV, 2014

[6] Romero, L. D. (2017, April 06). Alianza entre Distrito y Waze para mejorar la movilidad en Bogotá. Retrieved from https://www.elespectador.com/noticias/bogota/alianza-entre-distrito-y-waze-para-mejorar-la-movilidad-en-bogota-articulo-688042

[9] Urrego, Germán Enrique, Francisco Carlos, Calderón, Forero, Alejandro, & Quiroga, Julián Armando. (2009). Adquisición de variables de tráfico vehicular usando visión por computador. *Revista de Ingeniería*, (30), 07-15.

[10] Gray, Jeff. “U of T professor Geoffrey Hinton hailed as guru of new computing era.” *The Globe and Mail*. 07 Apr. 2017.

[11] Raina, R., Madhavan, A. & Ng, A. Y. Large-scale deep unsupervised learning using graphics processors. In Proc. 26th Annual International Conference on Machine Learning 873–880 (2009).

[12] R. Szeliski. Computer Vision: Algorithms and Applications. 2010.

[13] P. Viola and M. Jones. Rapid object detection using a boosted cascade of

simple features. In CVPR, 2001.

[14]] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In International Conference on Computer Vision, 1998.

[15] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In Computational Learning Theory: Eurocolt ’95, pages 23–37. Springer-Verlag, 1995.

[16] Leung, T. and Malik, J. "Recognizing surface using three-dimensional textons", Proc. of 7th Int'l Conf. on Computer Vision, Corfu, Greece, September 1999.

[17] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

[18] T. Joachims. Making large-scale svm learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods - Support Vector Learning. The MIT Press, Cambridge, MA, USA, 1999.

[19] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. MegDet: A large mini-batch object detector. In CVPR, 2018.

[20] Liu, Shu, et al. "Path aggregation network for instance segmentation." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[21] He, Kaiming, et al. "Mask r-cnn." Computer Vision (ICCV), 2017 IEEE International Conference on. IEEE, 2017.

[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015

[23] Liu, Wei, et al. "Ssd: Single shot multibox detector." *European conference on computer vision*. Springer, Cham, 2016.

[24] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[25] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In CVPR, 2017.

[26] Z. Luo, F.B.Charron, C.Lemaire, J.Konrad, S.Li, A.Mishra, A. Achkar, J. Eichel, P-M Jodoin MIO-TCD: A new benchmark dataset for vehicle classification and localization in press at IEEE Transactions on Image Processing, 2018.
[27] H.Jung, MK Choi, J.Jung, JH Lee, S.Kwon, WY Jung "ResNet-based Vehicle Classification and Localization in Traffic Surveillance Systems", Traffic Surveillance Workshop and Challenge, CVPR 2017
[28] Tao Wang, Xuming He, Songzhi Su, Yin Guan., " Efficient Scene Layout Aware Object Detection for Traffic Surveillance", Traffic Surveillance Workshop and Challenge, CVPR 2017.
[29] P. Felzenszwalb, R. Girshick, D. McAllester, "Cascade Object Detection with Deformable Part Models", Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR, 2010.
[30] Dutta, A., Gupta, A., & Zisserman, A. (2016). VGG Image Annotator (VIA) URL: http://www. robots. ox. ac. uk/~ vgg/software/via.
[31] Matterport, “Mask RCNN.” https://github.com/matterport/MaskRCNN, 2017.
[32] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." IEEE transactions on pattern analysis and machine intelligence (2018)
[33] Fizyr, Keras implementation of RetinaNet object detection, (2018), GitHub repository, https://github.com/fizyr/keras-retinanet
[34] Roytseng-tw, A pytorch implementation of Detectron, (2018), GitHub repository, https://github.com/roytseng-tw/Detectron.pytorch
[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceeding of IEEE Conference of Computer Vision and Pattern Recognition (CVPR), 2016.