Master Thesis

Object Tracking in Video with TensorFlow

Developing a Model for VID Imagenet Challenge

Andrea Ferri

Supervised by
Jordi Torres
Xavier Giro "I" Nieto

Final Project Report for the
Master in Innovation and Research in Informatics

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

BSC Computer Science Department
Universidad Politecnica de Catalunia
Catalunya, Spain
October 2016

Object Tracking in Video with TensorFlow

Developing a Model for VID Imagenet Challenge **Andrea Ferri**

Supervised by:

Jordi Torres
BSC Department of Computer Science
Xavi Giro "I" Nieto
UPC Department of Image Processing

Abstract

This Thesis[13] was born as collaboration between the BSC Computer Science Department [5] and the UPC Image Processing Group [23], with the purpose to develop an hybrid thesis on Deep Learning.

Nowadays, the interest around Machine Learning, is one of the fastest growing. So far from the side of the BSC Computer Science Department [5], that mainly uses his computational power for data mining and modelling analysis, the main purpose was to verify the difficulty to adapt his infrastructure "asterix", from the GPU Center of Excellence at BSC/UPC [4], the to Deep Learning. Instead, from the side of UPC IPG, there was the interest to test the environment developing a model for Object Tracking in Video that was suitable for the ILSVRC VID challenge [43]. To achieve the first goal and analyze the workload on the machine, I started to become an active user of the TensorFlow [21] community, learning from posts and blogs and I decided to implement a Virtual Environment that, led us to use different dependencies and different versions of the library software, depending on the model and purpose to reach. Till now, from the computer science point of view, this environment was the best choice and the most useful experience to work with, showing the easiness of use and implementation. I had some problems only with third part libraries, specifics for the Visual Recognition area, like OpenCV.

To develop the model for VID challenge, I began learning the basic knowledge for Deep Learning concepts, through on-line courses as the one of Stanford. Then I passed to the deepest and complex knowledge regarding the Visual Recognition topic, reading papers and understanding the main strategies and models that would be useful to me later, during the development. The discovery of many new concepts gave me enthusiasm and scared me at the same time, theory and practice were complementary, but it wasn't easy to pass from the first to the second one.

These latter were the most difficult part of the project, because it wasn't enough adapting my previous knowledge and programming skills to the new ones and mainly to the TensorFlow Environment. The Python library due to its recent birth hasn't developed many models or components yet, as for others environments like Caffe [3] or Theano[50], but the community interest is growing so fast that, luckily, I didn't had to start from scratch. I used some available models directly from Google [1] and some GitHub Project, like TensorBox[44] from a Stanford Phd Student[45] and tested others, like YOLO TF version[16].

The main components were some of the GitHub project I found, but none of them left me withouth problems.

Due to the time constraints, I started trying to extend OverFeat (TensorBox Project) [44] from single to multi class, spending efforts and many time trying to make the difference, increasing the quality of the model, on which I made also some important contribution, solving some of the main code pitfalls. At the end, the big reverse engineering work I realized with the help of the author, it didn't give the expected results. So I had to change the main architecture composition, using other strategies and introducing other redundant components to achieve a theoretically still image detection model. I had to introduce a time and space analysis to correlate results between frames and be more consistent in the detection and tracking of the objects themselves. Starting form the modular architecture proposed by K. Kang et al.[32], I decided to use the single class OverFeat implementation as General Object detector, training it on the whole class set and followed it with other components.

After the General Detector, I implemented a Tracker & Smoother to be more consistent in shape and motion during time and space on the whole frames set, using the knowledge of Slow and Steady features analysis explained by Dinesh Jayaraman and Kristen Grauman [31].

Inception component, the final one, is the most redundant module, because it's at the base of OverFeat architecture; but its use was the faster and only solution to label easily each objects. Thanks to the available model [20] implemented by Google [1], that it was trained on the ILSVRC Classification task, I had only to retrain it on a really smaller class set, thirty instead of one hundred, a workload sustainable by any personal computer available on the market. The complete architecture was composed by three main components in the following order of connection: General Detector, Tracker and Smoother, Inception.

Finally, I reached a working Environment and Model, that led me to submit results for the challenge, evaluate the workload for the "asterix" infrastructure, from GPU Center of Excellence at BSC/UPC, and prove how a Department can adapt and develop a working Deep Learning Research and Development area in few months.

The methodology I used could be defined Fast Learning and Fast Developing. Since I had to start everything from scratch, first of all, the necessary basic theory knowledge, after, the most complex and specific one and finally implement them in the short time interval possible, wasn't easy at all. These reasons according with the time constraints, pushed me to learn and develop in the fastest possible way, using tricks and tips and available components, saving time for the error solving. This latter is a consistent part of my work, solving run and project problems, which took me sometimes hours, sometimes entire days and once caused me a system-crash of "asterix" infrastructure, suffering ten days of black out during August, due to the summer period.

The model reached the last position in the VID ILSVRC competition, because of its low mAP results. As I will explain, the source of these results is the first component, that afterwards hasn't modules able to boost its accuracy; at the same time I will highlight how a different order of components and better implementation of them, for example a trainable Tracker plus Smoother, can be the starting improvements for this first draft work. Moreover, some other precautions can be taken on the Dataset for the train of single components, boosting their accuracy; I only trained the provided Train database, without using tricks and tips on it.

The thesis goals were fully reached in the best way I could, solving and walking through a path full of pitfalls and problems, which made my project harder to end.

Dedication

I dedicate this Thesis to all the people I have met in my life.

The good ones, that always believe in me and push me to never giving up in the worst moment of my life; thanks to all of you I've reached a lot.

Most important, I wanna thank, the ones I loose and left, you taught me a lot. Now I know who I want to be and what I want in my life.
Dedicate to my roots, I will never forget you. Thanks to all the people who joined me in this amazing experience of my life.

Acknowledgements

I want to thank, Professor Jordi Torres and Professor Xavier Giró-"I"-Nieto, for believing in me and give me the opportunity to develop this thesis, I learned a lot from it, but more from both of you; to BSC Computer Science department for the server infrastructure and more to "Marc", Marc Jordà, of the BSC Technical Support, I owe you a lot, you save me many times.

I want to thank all the Students and Researchers I met at BSC Computer Science department and UPC Image Processing Group, you let me feel at home and this was unique.

Finally thanks to all the People I have met this year in Barcelona, you are amazing and It's also because of you if I reached this goal.

To all the people I met in this two years that live around the world, I wanna thank you so much to be part of my life and story, I'm sure that life will always keep us in contact.

To all the people I met in my year at Politecnico: Grazie di cuore a tutti voi, compagni di Lezione, di Studio, in questo momento elencarvi tutti e ringraziarvi uno ad uno risulta emotivamente complicato, ma siete stati importanti, desidero che lo sappiate.

Vorrei ringraziare i miei coinquilini, che mi hanno sopportato nel bene e nel male. Alla fine del percorso insieme, siamo cresciuti, divenendo adulti come non avremmo potuto credere. Siete stati importanti, non ve l'ho mai dimostrato abbastanza.

Infine non meno importante, vorrei ringraziare la famiglia di J.E.To.P, per avermi accolto, cresciuto e spinto nel movimento delle Junior Enterprise non solo a livello italiano ma europeo; abbiamo condiviso momenti belli e brutti, tensioni difficili, ma importanti, "Always one step forward" rimarremo sempre connessi, nessuno può insegnare tanto professionalmente, se non una famiglia come voi.

To my Italian Friends: Grazie! L'amicizia per me è sempre stata un importante tassello di vita; vi ringrazio di non aver mai dubitato di me, di essere cresciuti assieme, di avermi accettato per quello che sono, di avermi reso consapevole della vita, dei suoi valori, di come non vada mai sprecato ogni suo singolo istante. Imparerò ad essere più connesso e presente nelle vostre vite, grazie di tutto quello che avete fatto per me, di aver compreso i miei momenti e di avermi sempre, anche a distanza, fatto sentire il vostro calore umano.

To my Family: Non c'è nessuno che vi superi, in ogni circostanza; questi due anni sono stati difficili per tutti, siamo maturati tanto e nella distanza abbiamo trovato la voglia di stare più vicini e di essere più sinceri l'un l'altro.

Papà, Mamma, grazie di tutti i sacrifici che avete fatto, per farmi rincorrere i miei sogni, grazie di avermi dato la fiducia per cadere e rialzarmi; è solo grazie a voi se ci riesco tutte le volte. Avete fatto tutto questo col sorriso più naturale possibile. Non sarò mai abbastanza bravo come figlio per ripagarvi di tutto quello

che continuate a darmi e trasmettermi.

Sorellina, ti ho sempre detto che avrei voluto un fratello e non una sorella, ma in cuor mio non avrei mai potuto sperare di meglio; sei una ragazza speciale e talentuosa, farai sicuramente più strada di me. Sono orgoglioso di te; so quanto hai sofferto e quanto sei diventata forte in questi anni; ormai sono io che devo imparare da te. Continua così e ricordati che per quanto possa essere distante e distratto, sei e sarai sempre al centro del mio cuore.

Zii, Cugini, parenti, elencarvi diventerebbe lungo, ma non posso non ringraziarvi. Ognuno di voi mi insegna giorno dopo giorno; è un pensiero che mi dà forza, mi fa sentire vivo. Porto parte di ognuno di voi dentro e vi voglio un bene profondo. Scusate la mia imperfezione, cercherò di essere migliore e di continuare su questa strada, non solo per me stesso, ma anche per voi.

Ultimi non per importanza, Nonni, se ho potuto scrivere tutto questo devo ringraziare voi che siete stata un'onda in mezzo al mare, che superando scogli e maree, ha portato alla luce una famiglia stupenda, eterogenea, stravagante, ma che si ama davvero, anche se a volte se lo scorda; siamo belli così. Siete molto importanti per me, a dir poco fondamentali, quello che avete significato e quello che siete stati, nei vostri pregi e difetti, sarà sempre parte di me; vi dedicherò tempo per dirvelo e dimostrarvelo.

Nonno, a te vanno un saluto e un ringraziamento speciale, perchè sei lì a guardarmi sbagliare e rialzarmi e, nonostante non possa leggere queste parole fisicamente, quel giorno sarò così felice e griderò così forte, che non solo vedrai la mia luce da lassù, ma sentirai il mio saluto; allora saprai che "anche a stò giro te l'ho fatta". Grazie di avermi insegnato che la perfezione si trova solo nell'imperfezione stessa.

Contents

List of Figures

List of Tables

List of Contributions

- Fixed a threading problem into Overfeat TensorBox Repository [8] [9];
- Added the model Graph of Overfeat to Tensorboard [11];
- Fixed some errors in the visualization of images in the TensorBoard summary [12];
- Fixed some errors into Google Inception TensorFlow Repository [10].

Chapter 1

Introduction, Motivations and Goals

In this chapter I will introduce firstly the Thesis architecture and with which methods I developed the Project, than I will introduce the two main Requirements of the Thesis.

1.1 Thesis and Project

The thesis [13] focus is to implement a Deep Learning Environment in a computational and modelling infrastructure, that must be dynamic and error prone. From this point,I implemented a model for one of the ImageNet Large Scale Visual Recognition Competition challenges[43].

A Dynamic Environment means that it must be easy to adapt and update to the needs, because the Deep Learning topic is a continuous Research and Develop, where it comes really hard to let working everything and be always up to date.

Due to the time constraints, less than 4 month, I decided not to start from the scratch but from some yet implemented community models. My goal was not to achieve the best results, but a working model, whose output was making sense. Starting from implemented models was not an error prone; through reverse engineering, I had to understand the code, adapt it and when needed, make contributions. This takes me more effort then expected, but at the end I achieved the goal to submit results to the challenge.

As we will see, one of the strong hidden dependencies in Deep Learning are Datasets, which are almost always provided and checked by the challenge hosts but, downloading and managing them, checking their effective accuracy was not so easy as everybody can think.

I will initially explain Visual Recognition State of the Arts, then the methodology I used for achieving this goal in a really high time constrain, how I reached a working model in TensorFlow [21]and finally I'll show the Results Evaluations and Conclusion. It's possible to find some basic knowledge on ILSVRC [43]and Deep Learning in the Appendix, in which the last chapter is a deeper explanation on the method and decisions I made to build up the Environment.

1.2 Requirements: Tensorflow and ImageNet

The requirements of this Thesis were chosen from each Department I'm working with. TensorFlow was chosen from BSC [5] Computer Architecture Department and ImageNET Large Scale Visual Recognition Challenge VID challenge [43] from UPC Image Processing Group[23]. I will highlight the main motivations on these two decisions.

1.2.1 Tensorflow

Figure 1.1: TensorFlow Logo.

Released on the 15th of November 2015 by Google[1], TensorFlow [21] is the newest open source library written in Python for numerical computation.

It has immediately a great success in the Machine Learning community and in less than one year it also had a lot of support and development by Google [1] itself, more over by many community projects, developed in any area of Deep Learning.

The peculiarity of TensorFlow is its work flux, made by data flow graphs. Where Nodes represent mathematical operations, edges represent the multidimensional data arrays communicated between them; the latter can be considered, as in electronics, a Tensor, from here its name. In May 2016, Google [53] has revealed that it has used TensorFlow in AlphaGo project, with a special hardware dedicated to boost library's performances.

To reach rapidly this goals, Google dedicated a special attention to the user experience of TensorFlow[21], which arrives with a great basic support and a well-grown GitHub community [22], the key of this quick improvement.

All these are the peculiarities that pushed us to choose TensorFlow [21], over more developed and bigger communities. The irruption made in the Deep Learning environment, shows up its great future potentialities, that are rolling out day by day.

1.2.2 ImageNet

Figure 1.2: ImageNET Logo.

In its 7th edition, The ImageNet Large Scale Visual Recognition Competition (ILSVRC)[43] was a worldwide competition , where one of the 5 main challenges was VID. This has the aim of Tracking Object in Video, identifying eventually multiple-objects, classifying them and more over giving a track number for each of them, if it's possible.

All the datasets (Training, Validation and Tests) are retrieved on the ImageNET Database, a collection of annotated pictures of different classes of objects. In the further chapter, I will explain in detail.

Developing a deep learning model in any area means to learn from the community and using, adapting, merging, writing code from it. So it was very difficult to reach the goal. TensorFlow community [22] has a fast growth, but the complexity of the models for the Object Tracking in Video are pretty far from the ones present in the nowadays community. Starting from the great idea of a model, composed by different parts proposed by K. Kang et al [32], which achieves the State of the Art in one of the categories of the last year competition, I used, composed and adapted some available models to reach the one I was looking for.

1.3 The infrastructure: Asterix

The Asterix server is part of the BSC-UPC NVIDIA GPU Center of Excellence. To better understand the infrastructure I used, I will explain a little its architecture and show its schema:

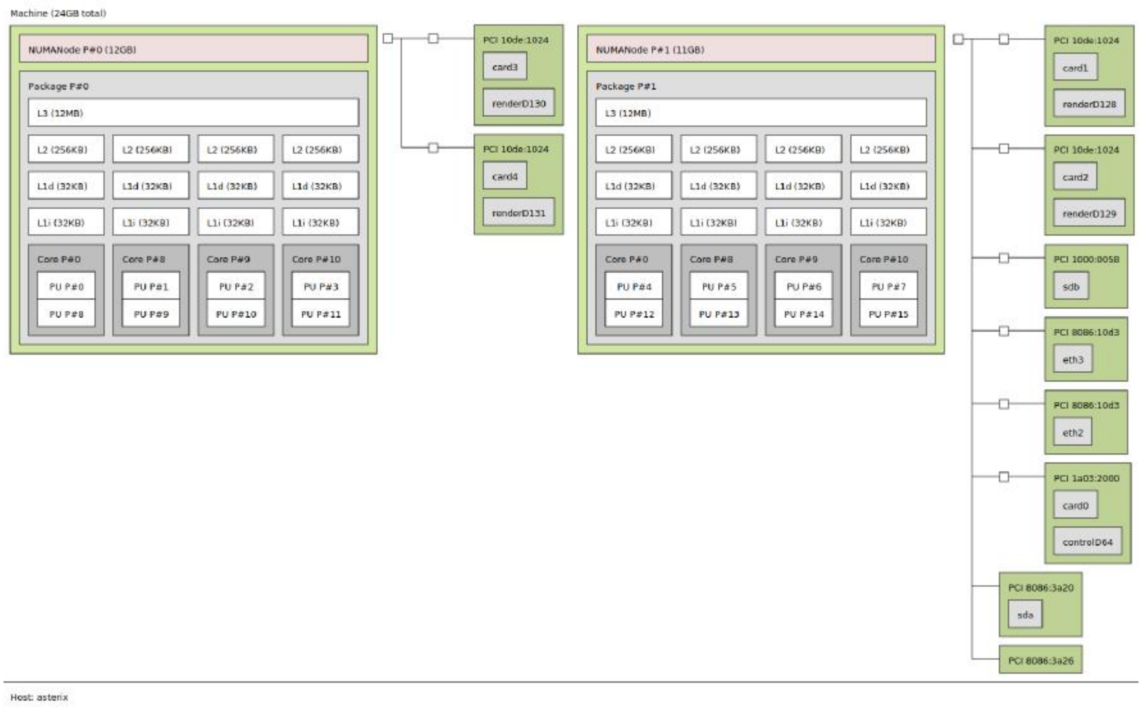
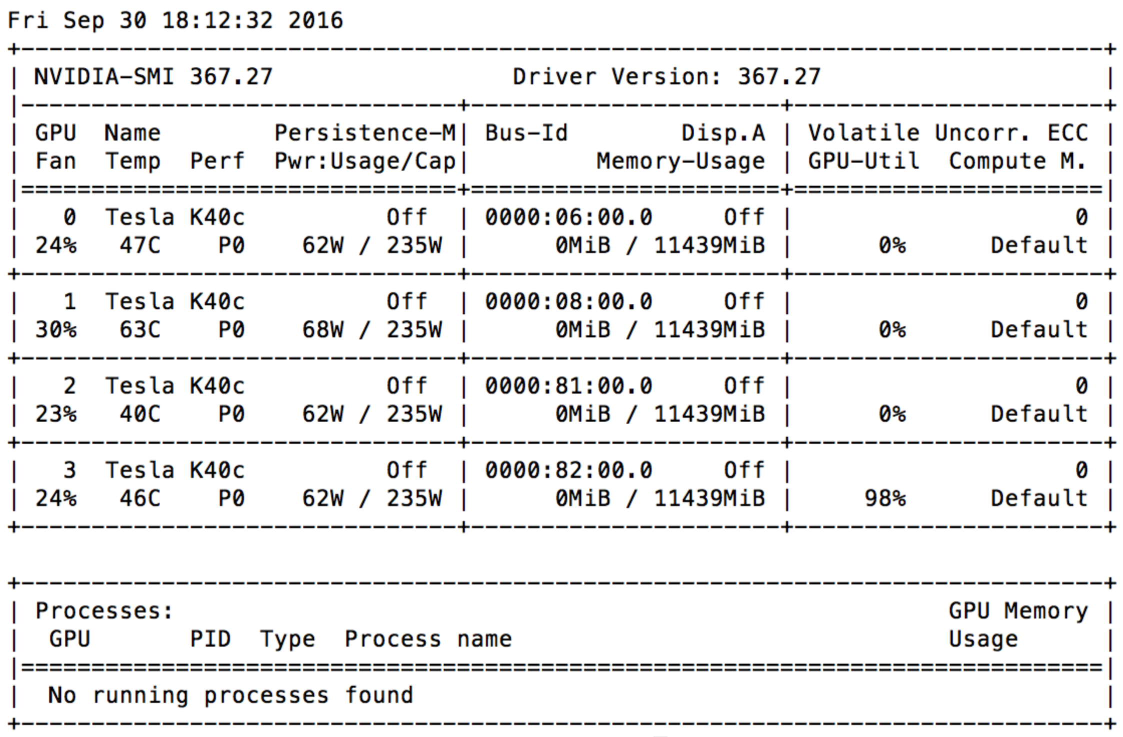


Figure 1.3: Asterix Topology schema.

The machine has a total of 24Gb of local computational power inter-node, divided into to two NUMA (Non-uniform memory access) Nodes. This design architecture is specific for multiprocessing work, where the inter-node memory is used to boost performance. In the two nodes there are 16 CPUs, divided in four core for node, each of them is a Intel(R) Xeon(R) E5620 @2.4 GHz; Each CPUs has three different level of Cache Memory, L3 of 12Mb, L2 of 256Kb and the first level is split into L1i and L1d, both 32Kb. Each NUMA node is supported by four graphic cards, in the figure card from 1 to 4. All of them as visible into the below picture are NVidia Tesla K40c of 12Gb of memory. The ”c” version of the K40 version is the one with an active fan sink that the card uses to cold down itself. This lets the card be suitable for many more purposes than other versions.



```
Fri Sep 30 18:12:32 2016
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.27                 Driver Version: 367.27                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K40c          Off  | 0000:06:00.0     Off |                    0 |
| 24%   47C    P0    62W / 235W |      0MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K40c          Off  | 0000:08:00.0     Off |                    0 |
| 30%   63C    P0    68W / 235W |      0MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K40c          Off  | 0000:81:00.0     Off |                    0 |
| 23%   40C    P0    62W / 235W |      0MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K40c          Off  | 0000:82:00.0     Off |                    0 |
| 24%   46C    P0    62W / 235W |      0MiB / 11439MiB |     98%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figure 1.4: Terminal Output of the nvidia-smi command on the server.

In total to the 24Gb CPU memory are accompanied 48Gb of graphics card. These latter are important and fundamental for the Deep Learning topic. Because they are the ones that architecturally support naturally the mathematical computation.

The time constraints and the many problems and errors I encountered obliged me to learn, use and implement all the knowledge I needed for the topic in the fastest possible way. At the end of the Thesis[13], I will be able to respond to the question: can TensorFlow [21] be the fastest adaptable environment for Machine Learning, for implementing Research and Development, into a different infrastructure architecture with a great improvement perspective?

Chapter 2

State of the Art

In this section, I will talk about the basics State of the Art for Visual Recognition. The concepts will be divided in the so called "Still Image Processing", because Time and Space properties of the analyzed photos are not considered and "In Space and Time Image Processing", where those parameters are considered and learned from the model.

2.1 Still Image Processing

The Still Image Processing is the area that focuses the learning in three different specific goals: Classification of objects in images, their Localization and finally their Detection.

As anticipated, all these tasks are learned by the model without considering Space and Time properties, because the starting hypothesis is that sequential photos are not correlated in time.

2.1.1 Classification

The Classification task is intended as the Visual Recognition of all the possible subjects inside the picture in the descending order of confidence, see Figure 2.1 for examples.

This task is at the base of all the others, because it recognizes if there are objects or not and which classes belong to. This is the starting point of the human visual focus on scene. Respect to the state of the art and the ILSVRC [43] challenges, in 2011 Classification was coupled with localization in a separate challenge and then from 2013 replaced by it, because the improvement rate was flat.

In the classification topic, we can cite many different models in a quality ascending order, that corresponds also to the incremental evolution of them:

- AlexNet (ImageNet Classification with Deep Convolutional Neural Networks, 2012) [33].
- VGG-Net (Very Deep Convolutional Networks for Large-Scale Visual Recognition, 2015) [42];
- GoogLeNet (2015) [47];

- Batch Normalization, Google (Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift) [30];
- Deep Residual Learning, Microsoft (Deep Residual Learning for Image Recognition)[25];
- PReLu/Weight Initialization, Microsoft (Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification)[26];

As we can extrapolate from the paper of Kaiming He et al.[26], a comparison of the five top methods, evaluated on the test set, is of the following order of magnitude:

Method	**Top 5-err (test)**
ResNet (ILSVRC '15) [25]	**3.57**
BN-Inception [30]	4.82
PReLU-ne [26]	4.94
GoogleNet (ILSVRC '14) [47]	6.66
VGG (ILSVRC '14) [42]	7.32

Table 2.1: Comparison of the five top classification models on test set of ILSVRC.

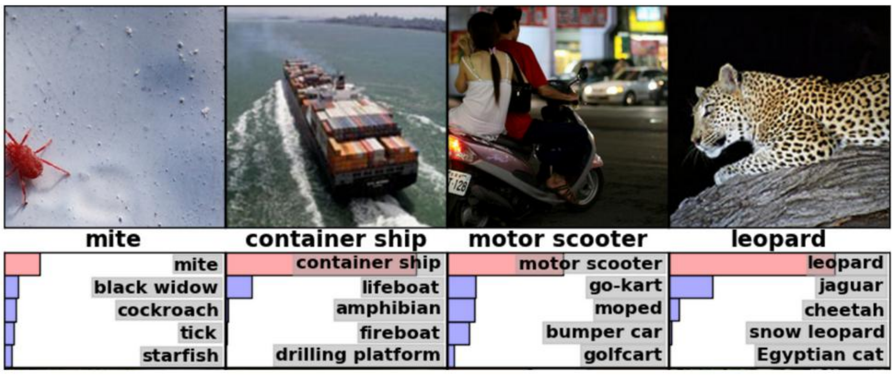


Figure 2.1: Inception model v3 examples of Image Classification (Recognition).

Due to the selected environment and its more available support, I chose the Inception model [20], that is now at the third version and it is fully implemented and supported by TensorFlow authors themselves [21]. This was one of the first decisions I made during the development of the work, sacrificing accuracy in order to respect time constraints and saving time to solve all the possible problems.

This model as we will see, is an evolution of the BN-Inception, that tries to maximize the accuracy minimizing the workload effort. Due to convolutional networks that are at the base of most of the art models' states, there is a lose of computational efficiency which could still be gained with factorized convolutions and aggressive regularization to work on mobile vision scenarios. The structure of the Inception Model and the concepts behind it are based on four simple design principles, which improve the model computation efficiency and accuracy, as the paper from Christian Szegedy et al. [48], shows:

- Avoid early bottlenecks in the model, let the acyclic graph be easily represented with a feed forward model from input to the regressor or classifier. More important, avoid dropping big dimensions, and do it gradually from input to output, ensure not to loose useful correlations values for the task.

- We can obtain a faster train by the use of high dimensions representations and by the activations per tile's increase in the convolutional network.
- To boost accuracy, it's better to reduce dimensions before parameters aggregations; this due to the high correlation between adjacent units.
- More the model is balanced and more the learning is consistent. So it's important to raise in parallel width and depth of the network adding filters where needed.

This concepts applied to the original BN-Inception from Sergey Ioffe and Christian Szegedy [30], that was itself an evolution of GoogLeNet by Christian Szegedy et al. [47], results in the following model structure:

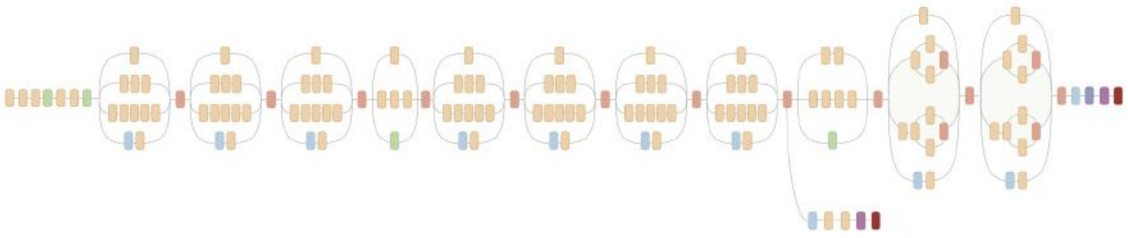


Figure 2.2: Architecture Representation of the Inception model v3 [48] developed by Google .

Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax

Figure 2.3: Legend of the different layers color, corresponding to the ones in the architecture representation of the Inception model v3.

As we can see from the figure 2.2, the graph is balanced in width and depth. It's mostly composed by Convolutional Layers (the Figure 2.3, shows the layer colours legend), that make computation and extract features arrays of the images, then at the end of each architectural tier, they are averaged and concatenated with the following tier. It's only at the two outputs of the model where Dropout and Fully-Connected tiers are used, linking layers, minimizing bottlenecks and sharing information between them and finally fading in Softmax layer to wonder the class. In term of evaluation of the models, in order to better understand the improvement rate we can look at the table below:

At the end, the starting decision I took brought me not to suffer in accuracy and save a lot of time instead of develop a basic component useful for detection and localization. I had only a nominal worst accuracy of 0.01 for Inception v3 respect

Method	**Top 5-err (test)**
Inception v3 ('15) [48]	3.58
BN-Inception [30]	4.9
GoogleNet (ILSVRC '14) [47]	6.67

Table 2.2: Accuracy evolution of the inception model of Google from the first version, GoogleNet[47], to the third one [20] on test datset of ILSVRC 2012.

to the 3.57 of the ResNet. But the highest quality version of Inception, reaches 21,2% , top-1 and 5,6% top-5 error for single crop evaluation on the ILSVRC 2012 classification [28], setting the new state of the art. I think this is more than an acceptable and a great partial result of the project.

2.1.2 Localization

Localization and Detection sound really similar and most of the community gets confused sometimes from the slightly difference. Both aim to detect from the input image to an output made of the presence/or not of objects. If they are present,they have to output an equivalent number of bounding boxes that highlight the objects in the picture.

But, the first one works with 1000 classes, still and moving objects, outputting the top 5-error labels it detects. Instead the second one works with only 200 moving objects typology and it outputs only the best one.

In few words, the difference is the typology of the objects, moving and still objects for Localization and only moving ones for the Detection.

Now that the difference is clear, I think it's easier to understand why classification task was the principal starting challenge, from which they added a parallel Localization one, which they were merged in 2013 and in the same year a different task with less classes, Detection, and more other complex scenes and video challenges were added.

It is interesting to understand why I won't go through models in this section but only with some other considerations, that will explain better the evolution of the challenges and of the topic itself.

In 2011 results [27] of the Classification plus Localization challenge there were only two brave participants, University of Amsterdam [2] & University of Trento and ISI lab. [51] of the University of Tokyo [51]. The winners of that year, University of Amsterdam and Trento, were the first which implemented a selective search with some constraints, combined with an hierarchical grouping of adjacent regions of colour/text achieve a flat error of 0.425mAP score. This score is only 0.08mAP higher than the Classification result of the same participants for the only Classification category. 8% of increase is good but not so surprising. Above the best results from the winners model.

What is really clear from 2011 ILSVRC competition [43] is that Localization task could improve the accuracy of the classification, but it depends completely from the latter accuracy. This means that, the better is the Visual Classification of the image, understanding each ground, color and light level, with the scope to merge and combine those representations in the best way to achieve the highest "human-level" knowledge, the better and easier will be the localization of the objects themselves.

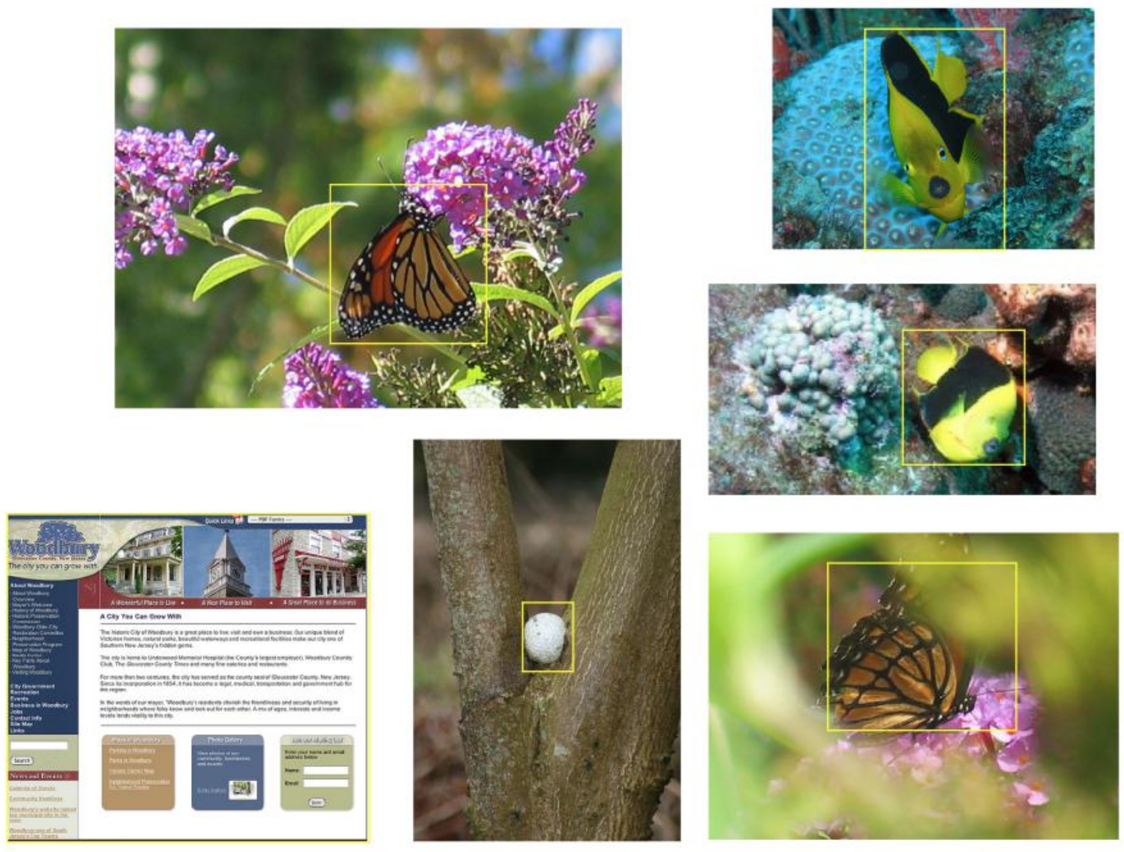


Figure 2.4: Best Localization results, from the winners University of Amsterdam [2] & Trento [51], of ILSRVC LOC 2011[27].

The winners score gives a 50% overlapping error and a really higher accuracy depending on the classes. Comparing per-classes results of the classification and localization tasks, the paring was natural and a confirmation of the above concept. In Figure 2.5 some example of the overlapping error, related to images, where the subject is not thus evident.

So, it results clear that to boost accuracy is important to classify the image with different manipulations, size, crops etc. Obviously, this must be done in parallel with the growth of the length of the network, helping the algorithm to understand, also when it's difficult for the picture, if which and where subjects are present in the picture.

This is not only a matter of algorithm but also, of the network experience, meaning a good train session, avoiding over-fitting, using the best datasets possible. These latter are the ones that define the quality source of the learning experience and it's a concept never to underestimate.

From here forward, I will not differentiate between localization and detection again, because from my thesis point of view, results to be the same. To observe which dataset I was provided with, please read the Development of the Work Chapter. Then for a deeper knowledge about ImageNET datasets [40] and ILSVRC [43]competition, please, go to the Appendix 1.

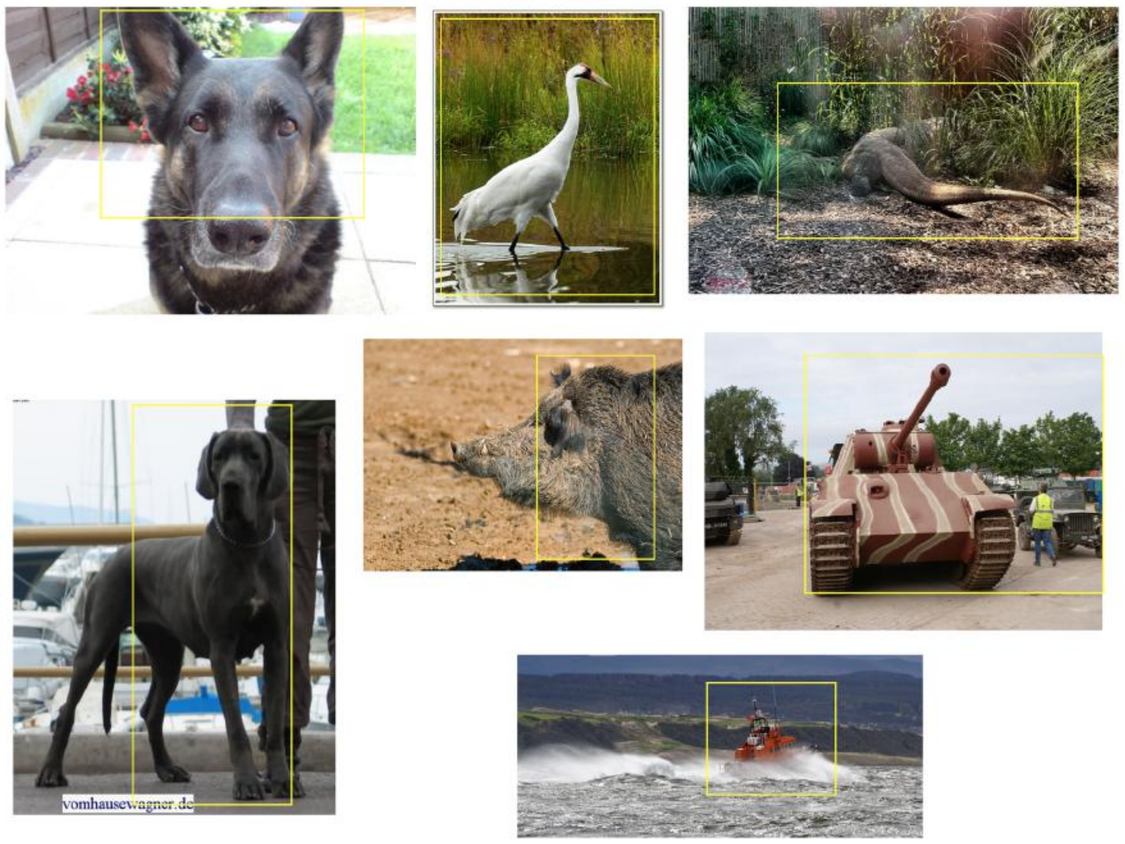


Figure 2.5: Localization overlapping error examples, from the winners University of Amsterdam [2] & Trento [51], of ILSRVC LOC 2011[27].

2.1.3 Detection

Detection, as discussed above, is the task to verify the presence of objects and if positive, highlight them with a bounding box. Previously, I talked about a selective search with hierarchical grouping of adjacent regions of colours. This was one of the first approach to solve that task, but now, I will, like for Classification, give a brief overview on the topic and than explain a little more in deep the one I choose.

Here are the main models for the detection topic:

- Fast R-CNN, Microsoft Research (Ross Girshick)[14];
- Faster R-CNN, Microsoft Research (Faster R-CNN: Towards Real-Time Object Detection with Region Proposal, 2015)[39];
- YOLO (You Only Look Once: Unified, Real-Time Object Detection, 2015)[38];
- OverFeat, NYU (OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, 2014)[41].

This list of models touches all the types of basics strategies for the object detection. One of them the Faster R-CNN [39]could be also considered in the next paragraph, due to its naturally adaptation to the video task. But I will explain others models, more focused on achieving an In Time and Space learning model.

Here below a comparison of the models, to better understand the magnitude order relation: Fast [14] and Faster [39] R-CNN, are based on the R-CNN model

Method	**Test Dataset**	**mAP%**
Fast R-CNN, [14] 2014	VOC 07+12	68.4
Faster R-CNN (RPN + VGG, shared), [39] 2014	VOC 07+12	70.4
You Only Look Once, [38] 2015	VOC 07+12	64.3
YOLO and Faster R-CNN, [38]2015	VOC 07+12	75.0
Overfeat, [41] 2013	ILSVRC2013	24.3

Table 2.3: Comparison of the main detection models, please pay attention to the different test database used and the implementation year.

from Ross B. Girshick et al. [15]. The Region Convolutional Neural Network, is a base architecture that analyzes the image at different level of convolutional and max pooling layers to produce a feature maps. Each output proposal goes through a Region Of Interest pooling layer that extracts fixed-length feature vectors. All this are faded into a Fully connected layer that outputs into two different softmax tier. The first one outputs the K object probabilities classes plus a catch-all "background" class. The latter returns four real values for each of the K item types, corresponding to the bounding boxes coordinates.

This model is no longer maintained, due to its successors, but in PASCAL VOC 2012 it was a state of the art, in which the authors used a variation with a bounding box regressor to boost performance to 53,3%mAP reaching a new limit. Here below a brief table resuming its historical results: In the following reference links, we can

Method	**VOC 2007**	**VOC 2010**	**VOC 2012**	**ILSVRC2013**
R-CNN	54.2% mAP	50.2% mAP	49.6% mAP	-
R-CNN bbox reg	58.5% mAP	53.7% mAP	53.3%mAP	31.4% mAP

Table 2.4: R-CNN median results, from the 2007 to the 2012 VOC dataset; the last column shows the mAP of the model on the ILSVRC 2013 dataset, this last result is more significant for my purposes.

see some per-class results, VOC 2007[15], VOC 2010[54] and VOC 2012[55]. The base R-CNN model is pretty slow to execute but also the Fast version that has some improvements. It's the Faster version, introducing a smart tip, that can reach almost real time executions. This strategy is the Region Proposal Network, which used in parallel with the Fast R-CNN [39], can output cost-free region proposals, sharing the full-image convolutional features detection layers between both systems. Faster R-CNN [39] runs between 5-17fps and 58-59,9% mAP that are near the last year, state of the art value 62,4% mAP by MSRA[25].

Here some results of the Faster R-CNN on the PASCAL VOC dataset, that can help to understand why it's considered as a milestone for models in the Visual Recognition history and why it's at the base of the state of the art detector, written by the same authors Kaiming He et al. The one that I chose, due to a yet semi-working project I found, is Overfeat[41]. Which is An integrated system for Recognition, Localization and Detection based on Convolutional Layers. The authors, Pierre Sermanet et al, starting from the winners model of 2012 [33], integrating sliding windows, multiscale classification, a Regressor and combining their predictions achieve a fifth place with a 13,6% top 5-error rate in 2013 post competition. Where they replaced part of the model with a deeper model respect the one

Model	Training Data	Test Data	mAP	ms/fig
Faster RCNN	2007 trainval	2007 test	69.9%	198ms
Faster RCNN	2007 trainval + 2012 trainval	2007 test	73.2%	198ms
Faster RCNN	2012 trainval	2012 test	67.0%	198ms
Faster RCNN	2007 trainval&test + 2012 trainval	2012 test	70.4%	198ms

Table 2.5: The Faster R-CNN model considered is the version with the VGG-16 as Visual Recognition base model. The training and test data belong to PASCAL VOC competition.

of the competition.

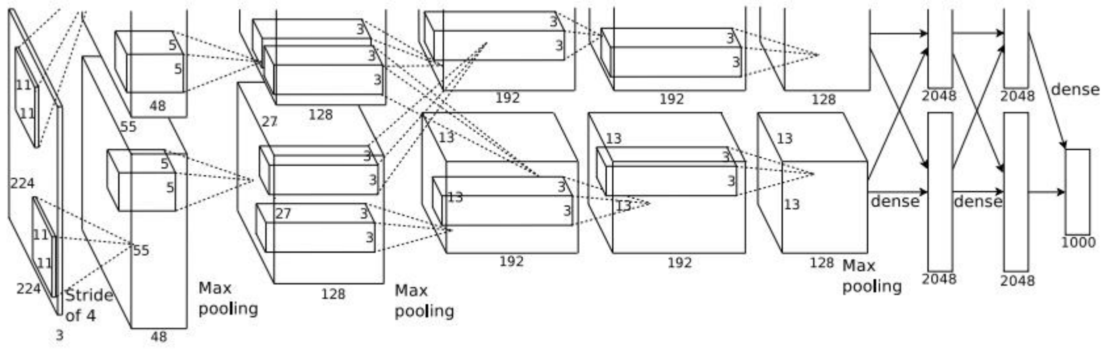


Figure 2.6: Schema of the best Classification model, from the winners of ILSRVC CLC 2012 [33].

Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	231x231	24x24	12x12	12x12	12x12	6x6	1x1	1x1

Table 2.6: Architecture specification of the fast model used by the team during the ILSVRC CLC, LOC and DET 2013 competition

Layer	1	2	3	4	5	6	7	8	Output 9
Stage	conv + max	conv + max	conv	conv	conv	conv + max	full	full	full
# channels	96	256	512	512	1024	1024	4096	4096	1000
Filter size	7x7	7x7	3x3	3x3	3x3	3x3	-	-	-
Conv. stride	2x2	1x1	1x1	1x1	1x1	1x1	-	-	-
Pooling size	3x3	2x2	-	-	-	3x3	-	-	-
Pooling stride	3x3	2x2	-	-	-	3x3	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	221x221	36x36	15x15	15x15	15x15	15x15	5x5	1x1	1x1

Table 2.7: Architecture specification of the deeper model used by the team during the post ILSVRC CLC, LOC and DET 2013 competitions to reach the upper-cited result.

As we can see from the tables 2.6 and 2.7, both versions are well balanced in depth and height and follow mainly the rules described in the previous section 2.1.1 for Inception v3.

The ConvNet when applied with a sliding fashion approach are inherit efficient as the author [41] underlines; because at test time, the higher workload, caused by bigger images, is confined to a small region and it becomes more computational useful when resources are limited.

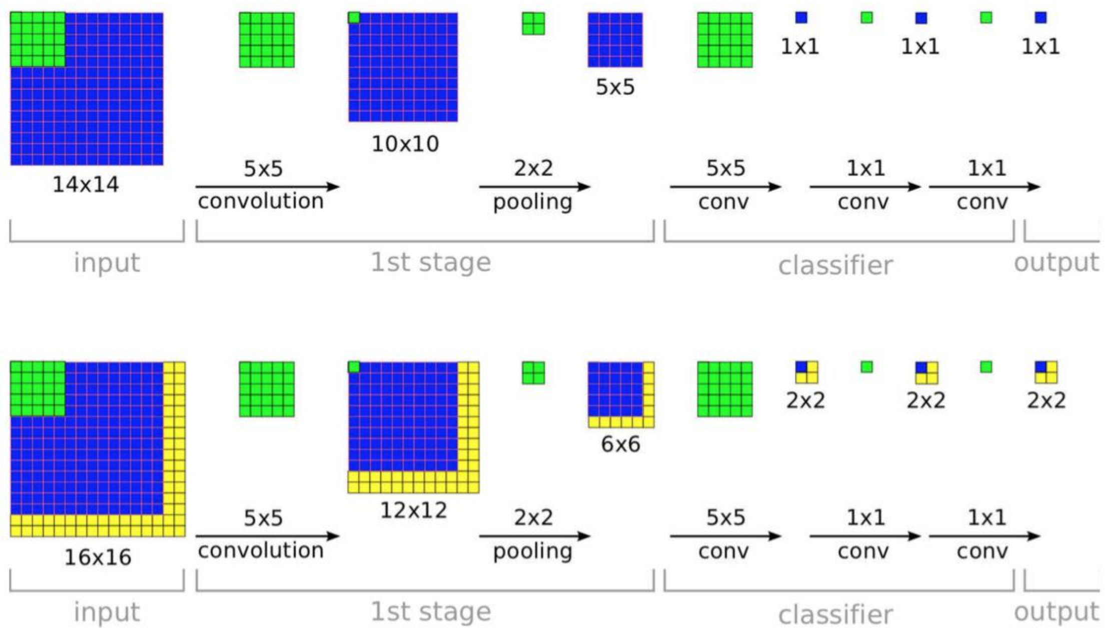


Figure 2.7: Schema of the ConvNet inherit advantages at run time.

The ConvNet use gives an hidden advantage to Overfeat[41], which is more computational aware of the resources to use; this is not to underestimate at all. In the Appendix 1 I will cite the YOLO[38] model I listed above, underling how its approach is one of the more innovative on the field; re-designing the main approach with a single neural network, instead of a deeper architecture.

2.2 In Space and Time Image Processing

All the previous described task for Still Image Processing, can be combined with Time and Space constraints for video analysis. Single frames get correlated each others and the will to share information between them becomes immediate.

We are going to see the principals concepts behind these strategies and than some models that implement them, with more or less complex graph.

2.2.1 Principals Concepts

The principal concept behind the In time and space analysis is the Slow and Steady Feature Analysis reported last year by the paper of Dinesh Jayaraman and Kristen Grauman [31]. This two analysis are based on the concept that, features, which are strictly correlated in time, have smooth changes. For a better comprehension look at the Figure 2.8 that shows the schema of an optimizer, which has learned, how to plain changes in time and works as a smoother for functions in input.

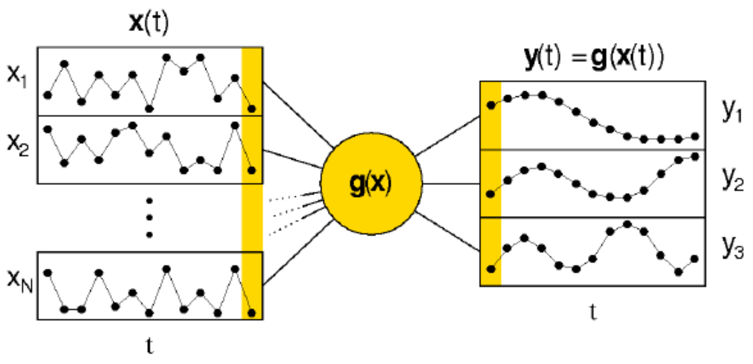


Figure 2.8: Concept Schema of a smoother unit for functions.

The use of this type of analysis at different levels of image features, lets the algorithm operate in time, smoothing changes between strictly correlated in time frames, but it also understands what is happening in the space dimension. This means that we have to work, not only with the first order of temporal coherence but also, with the higher ones. The best way to do it, is from the first frame to the last frame, following the entire scene, understanding all objects time and space's properties. Here a schema showing the concept.

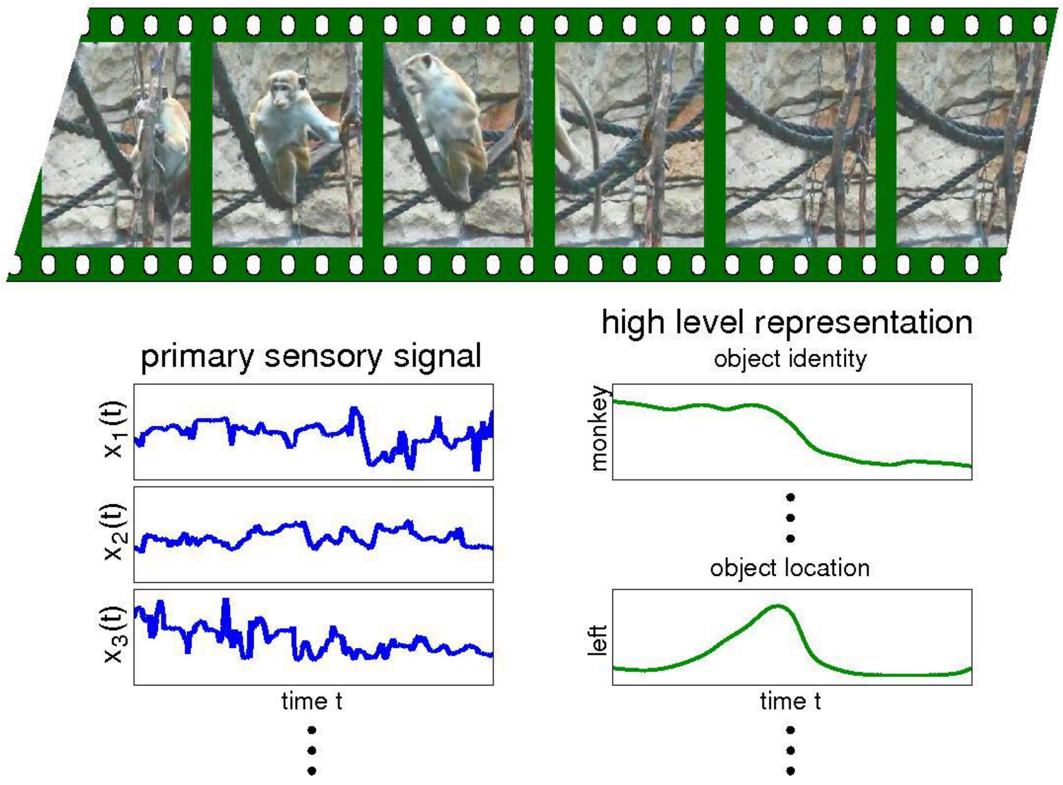


Figure 2.9: Concept Schema of a smoother applied to a Video.

The primary signals are represented under the picture on the left; instead on its Right, the high level representations of the smoothed extracted features show what the network is learning.

Using these concepts to develop an optimizer algorithm for the smoothing task an extracting from lower to higher levels features, it gives us the knowledge that: ”In the video there is a monkey that in the time and space dimension disappears from the scene”.

2.2.2 Models

At this point, it’s time to analyze some of the main models for object tracking in video, that are defining nowadays, the State of the Art for this task:

- T-CNN (Tubelets with Convolutional Neural Networks for Object Detection from Videos, 2015)[32];
- Seq-NMS (Sequential - Non Maximum Suppression, 2015)[24];

The first one is the winner of the ILSVRC 2015 VID challenge, the latter finished third. The second model overcomes the single frame detection, sharing information at the Non Maximum Suppression algorithm level. The NMS part of many common models like Fast R-CNN, doesn’t always choose the best Intersection Over Union bounding box. The idea behind Seq-NMS is to re-rank boxes sharing the temporal information between frames, ensuring the higher possible consistency of the detection.

Method	**mAP%**
T-CNN	67,8
Seq-NMS	48,7

Table 2.8: Accuracy of the listed models.

The core of the model is a loop composed by three subroutines: Sequence Selection, Sequence Re-scoring and Sequence Suppression. Firstly, a Sequence of bounding boxes between frames is selected by linking results of two adjacent frames, then it’s re-ranked respect to a Function F (average or maximum), finally it’s suppressed and the steps, repeated until no sequences, are selectable.

Here the diagram of the process:

The most complex model which inspired me is the Tublenet Convolutional Neural Network developed by K. Kang et al [32], its the last year State of the Art model for ILSVRC VID competition. Starting from the below general architecture schema, I will explain the principal concepts behind this model and gives some visual results from it:

The model is composed by four main components: Still-Image Detection, Multi-context Suppression and Motion-guided Propagation, Tublenet Re-scoring and Model Combination. The still image detector returns detections for all the frames of the video, that are separately threatened by the two following components. The Tublenet re-scoring uses a tracker algorithm to obtain sequence of bounding boxes, that are than clustered in positive and negative according to the statistics of their detections.

The other parallel component uses, in first instance, the Multi-context Suppression that orders the classes score confidence detections in a descending order and maps them in high confidence and low confidence, reducing the false negative

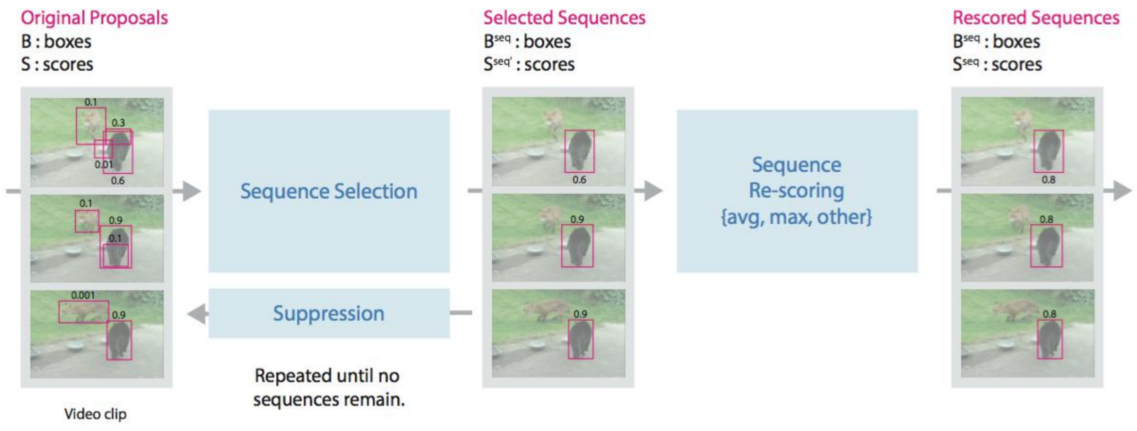


Figure 2.10: Concept Schema of the Seq-NMS, bounding boxes sequence are firstly selected and re-scored (from Left to Right), then suppressed (from Center to Left) and this loop is repeated until any sequence remains selectable.

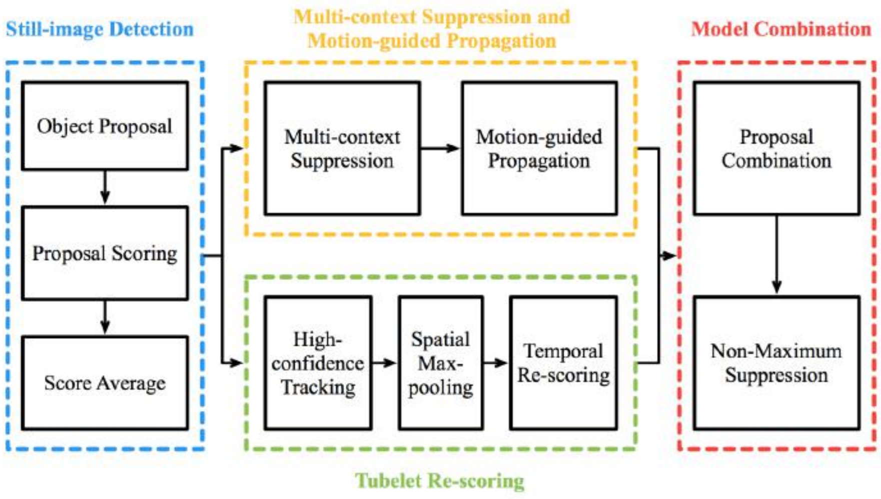


Figure 2.11: Concept Schema of the T-CNN, composed by four main parts with relative internal sub-modules.

possibilities by suppressing the ones mapped as low. The second sub-component, Motion-guided Propagation, shares the missed boxes between adjacent frames, using motion information such as optical flow.
The final component combines the previous two outputs, mapping to [0, 1] each min-max, combined by an Non Maximum Suppression process with an Intersection Over Union threshold of 0.5 obtaining the final results.

The Model structure lets the possibility to switch sub-components of the main parts and also of the principal modules themselves. In this way it's possible to test performances and accuracy of the different combinations. At the same time we can add extra components with agility, due to their independence relationship.

This underlines the dynamicity and the powerful of this approach, but more over why I choose to use it to achieve a model for the VID challenge. Here below some example images of the results from the T-CNN model: As we can see, the bounding boxes in the Figure are really tight to objects and no false negative are represented. It's really interesting to look carefully the last scene where, also when objects are

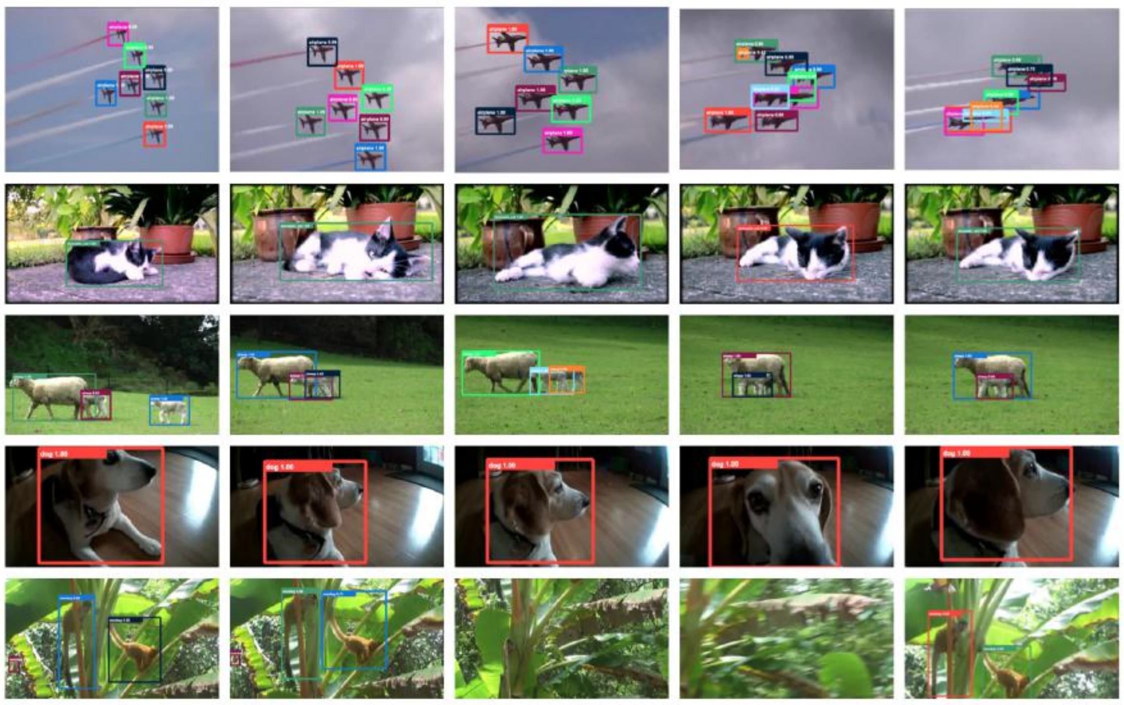


Figure 2.12: Qualitative Results of the T-CNN model.

hidden or bad caught, the algorithm works perfectly.

Chapter 3

Methodology

This chapter goes through the methodology I used, to achieve the goal of the thesis in a short time constraint. I will highlight how I used the instrument of the Digital Era and my previous programming and mathematical knowledge to learn fast and apply faster, to respect the deadline of the results submission.

3.1 Fast Learning and Fast Developing

In order to understand my methodology, it is important to know the starting point and my previous knowledge of the Deep Learning topic.

My experience is pretty various and heterogeneous. In Italy, at Università di Ferrara, I studied the bachelor of Information Engineering, with the deep basics of Programming, Math, Electronics; then I began my master degree at Politecnico di Torino, in the topic of Software Engineering and Digital Systems. It is only in this later year, here in Barcelona at the UPC FIB, where I approached more Research topics as: Complex Social Network, Supercomputing Architectures and Cloud Computing.

I've always loved challenges, but more in deep the ones that ask me to adapt, as fast as I can, my knowledge to achieve a goal. So I took all the nearest knowledge to the Deep Learning topic I had and adapted them. The most useful were Data Mining, Complex Social Network, Math, Signals, and Software Engineering.

All the topics were helpful to understand the papers and the knowledge I had to learn in the faster possible way. However, the Deep Learning is a new hot topic, with many various different areas and cross applications; even with the hints of the Image Processing Group, entering in the topic was hard. The reason was that I had to find my own way to go inside the topic and understand it, before making any further decision.

To do this, I used the best instrument of the Digital Era the online communities, in which any user can learn from other errors and questions avoiding problems and facing only the minimum ones. The power of the communities as I will explain better in the next paragraph, is really not to underestimate.

So I searched people that had my same problems: learn fast a basic knowledge of Deep Learning to understand the topic. Here with some IPG hints I found the Stanford course of Convolutional Layers [Citation]and before I learned basics concepts around the web, as the one I illustrated in the Appendix 1, that are at the base of the Machine Learning and Artificial Intelligence topic; with these, follow the

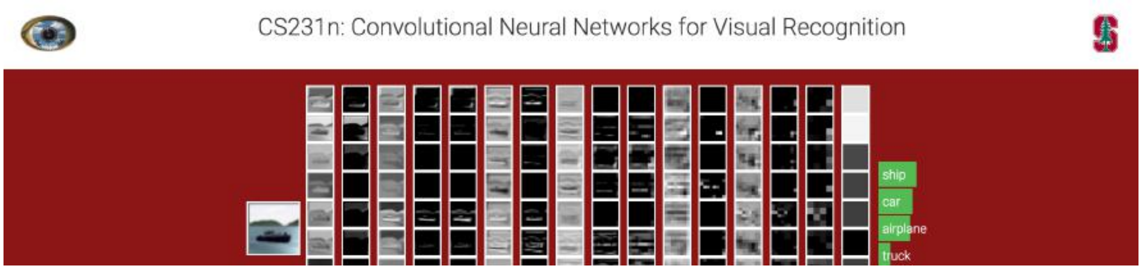

CS231n: Convolutional Neural Networks for Visual Recognition
ship
car
airplane
truck

Figure 3.1: Stanford, CS231n: Convolutional Neural Networks for Visual Recognition website screen.

course was easy and really clear.

At that point my knowledge was sufficient to let me search and understand efficiently any information I could find not only in the web, but also from external hints.

Not to underestimate the ILSVRC competition, on which I hadn't any clue; I started to search and learn information about its history, its evolution, its participants and hosts. This information is really useful when you can't loose your time and you want to be effective in the searching useful papers and knowledge. Some of the basics information I retrieved and I used to understand several concepts are in the Appendix 1.

At this point knowning the principal concepts, models, restriction and state of the art of the competition, I had to find projects to use, because it was clear that it was impossible to implement a model from scratch with my experience in the topic, but more over with TensorFlow environment and the time constraints.

The GitHub community was the answer to my problems. Here I found a large supported group of followers for many application areas, for example the Visual Recognition. One of the main problem was understand the nowadays situation and available resources, that thanks to some user projects [citation link to projects], I had the answer ready to learn; and so I've done.

One of the main issue was the Requirement TensorFlow, that was so new that finding rights projects and examples to use was really a mission. But fortunately, the rest of the community had the answer ready for me, listing tutorials, examples, most rated user projects.
Through users summaries projects on GitHub, I finally completed my overview of the Requirements and I was ready to make good sensible decision, without loosing precious time.

The knowledge needed was really wide and asked me to connect most of it with mine, but this wasn't enough to complete and achieve the goal. It was the time to develop not only the environment but mainly the working model in the fastest possible way.
Many of the participants to the ILSVRC competition, work on their model and results from and for many years; and mainly they are not alone during the elaboration of all the parts of the architecture. Building a working model could takes months as training it and improving it, but I had only four month to learn and develop it.

So I started to achieve a working Virtual Environment in TensorFlow, solving problems and installing dependencies with the help of the Technical Support due to the sensible operations. When this was reached, it was the time of the model.

Initially, I tested the environment with a YOLO Tf version project [16], that it wasn't working at all on the TensorFlow library (training wasn't yet supported and still not). Then I found the TensorBox project that was the one more suitable for my purpose. Here the community was fundamental when problems were raising. I've also made some contribution to the project solving some errors and warnings.

The difficulties and problems were still present, because due to some bugs in the GitHub source, I had to re-adapt my project to a different model architecture and introduce some redundancy to achieve a suitable structure to use in the VID ILSVRC competition.

A special note goes to the TensorFlow community that even if it is very young, it has a great banch of interested supporter and lots of yet projects.
Finally everything started working and it was ready for the results submission, achieving the basic requirements of 0.1% mAP to participate to the competition.

3.2 Community and Environment Magnitude

This section is personally dedicated to the importance of the Community and Environment, that played a key role in my work and experience; but mostly in the Digital Era evolution.

As the reader can understand from the previous section, the community was my best co-worker, it helped me, it solved my problems and it signed me the right direction in order to reach my goal. This is not to underestimate, this is why nowadays we hear about genius boys that born in their own room learning from the community and without previous knowledge change our future.

The community,as I will explain in Appendix 3, is the key factor of the fast growth of TensorFlow, because nowadays open-source and big companies support are the secret ingredients with which the Digital Environment is changing directions, choosing new topics.

This is a matter of User Experience, where some technological companies fail, without giving the right support and experience to the technical users. These need real-time support, an agora where they can share problems and questions, find other users to collaborate with and from whose experience everybody can learn.

These necessities come from a fast evolving environment as the Digital Era one, in which the time is too short to have deep knowledge in any area and it becomes more important the capability of adaptation of experience and base knowledge. For choosing a technology instead of others, it is significant to take care of these aspects, because they will affect not only the present or the near future, but above all the long term future, where solidity and tested support are fundamental for a long drive.

Moreover, it is important to not forget the goal and the scope of the research we are working on, because these aspects will really affect our success. The high level products of our Era are sensible not only to the ending user experience but also to the support and developers, who had to be happy to work on them and must be a stress-less environment, to engage an higher number of people with the minimum effort.

Chapter 4

Development of the Work

This section goes through the development of the model I implemented to participate to the VID ILSVRC challenge[29].

In the first part I will show the structure of the whole project, describing the single components and the time and space strategies I used. Then in the second section I will describe the Dataset I had to use, which were the main problems I encountered and how I overcame them.

4.1 The Project Architecture

The Thesis project is available on the GitHub repository [13] I've made.

In this way I gave back to the community what they taught me, as well as this project could be a starting point to develop and grow a collaboration team between my two main University supporters (BSC Computer Science department[5] and UCP Image Processing Group[23]), to participate to ILSVRC[43] in the further years.

Starting from the modular architecture proposed by K. Kang et al in the paper [32], I decided to compose the model with three simple main modules: a General Detector, a Post Processing Tracker and a Classifier. In this way I had the possibility to modify separately all of them.

This is the general overview of the model: In the following paragraphs, I will

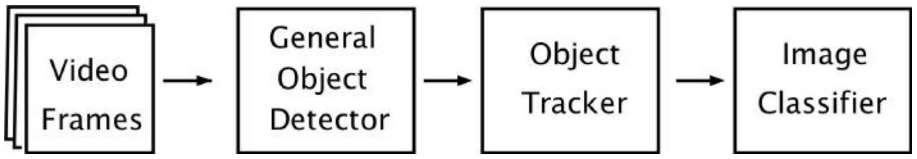


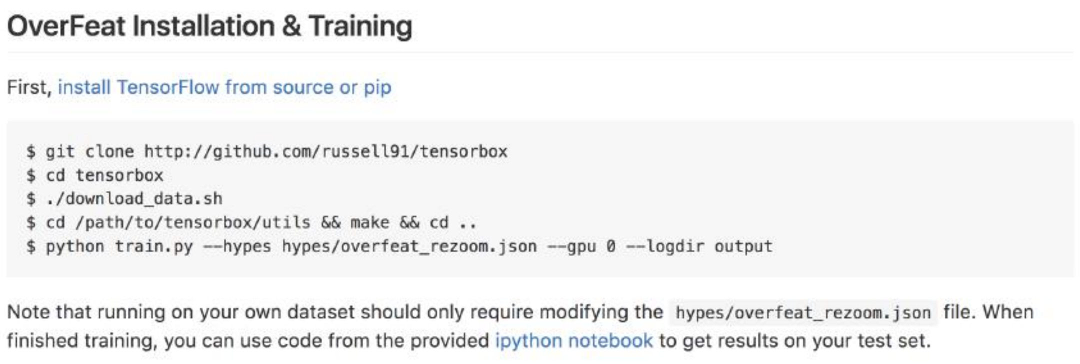
Figure 4.1: General Architecture of the model I developed. As we can see, frames are processed firstly by the general detector, then detections are improved with a post processor tracker and finally each bounding boxes sequence is labeled from the classifier.

explain more deeply each component, respectively TensorBox[44] for the General Object Detector, Inception[20] as Classifier and In time and space analysis; for the tracker. Now, I would like to spend some more words on this general structure I selected. The modular architecture wasn't my first choice, but due to some problems I encountered in the Overfeat detector project (TensorBox)[44], I had to fast adapt, (end of July) the whole structure to be more flexible, so I could also make some small changes and experiments without encounter big adaptation problems in the fastest possible way.

Eventually, this forced choice shows to be the best one, not only for anyone whose starts to work on the topic, but also for the ones that want to be as flexible as they can and embed code and components to boost performances and accuracy. For these reasons, with my modest knowledge, I suggest to consider to adapt your own work code and model to a modular general structure, gaining in flexibility and possibilities of further development.

4.1.1 TensorBox

TensorBox is a GitHub project [44], which aims to reproduce the Overfeat Model of the paper from Pierre Sermanet et al [41]. The repository was created and is still maintained, also thanks to the community, from a Stanford PhD Student [45]. In the same project, the author, shares his personal developed model ReInspect, see its related paper [46], that is an evolution of the Overfeat model, cited and implemented before in the same repository, for Object Detection with high overlapping instances.



OverFeat Installation & Training

First, install TensorFlow from source or pip

```
$ git clone http://github.com/russell91/tensorbox
$ cd tensorbox
$ ./download_data.sh
$ cd /path/to/tensorbox/utils && make && cd ..
$ python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
```

Note that running on your own dataset should only require modifying the `hypes/overfeat_rezoom.json` file. When finished training, you can use code from the provided ipython notebook to get results on your test set.

Figure 4.2: Screen of the GitHub Repository of the TensorBox [44]project I'm describing in this paragraph.

The project was a single class detector, but due to the fact that it was based on Inception model v3 [20] distributed by Google [1], there was the possibility to adapt the code of the model to be multi-class, easily. Then I opened an GitHub Issue and talked with the PhD Student, we adapted the code from our own point of view. After I Ended one million iterations of training on the VID dataset, which showed us that the network wasn't learning, detecting and wondering all the different classes but only one, our changes didn't work.

My main issue was that the hacked code of the inception model hadn't been wrote by Russell [45] himself, but by a friend of him. So any of us was really aware about the problem source. To understand its origin, there was not only the need of a strong reverse engineering work, but also of tries and tries on a field I wasn't comfortable with. My knowledge only had broken the outer shell of the Deep Learning topic, to be agile at all in the development, it's necessary to have years of experience.

To better understand the situation look the figures 4.3, 4.4, 4.5, 4.6 representing the model structure of the project, that with the principles I described into the previous chapter 2.1.1, I can say that it's not balanced at all and this is showed by the run-time test. Instead if we look the Figures 4.7 depicting captions of results

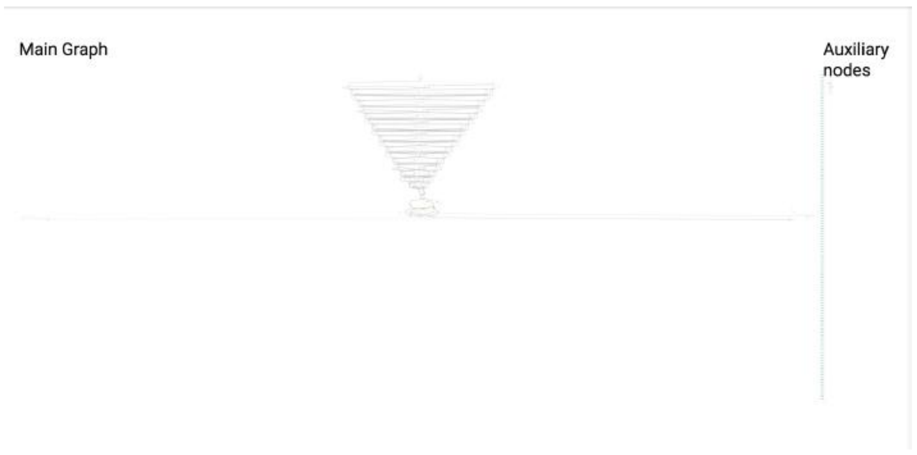


Figure 4.3: Full TensorBox graph, taken from TensorBoard.

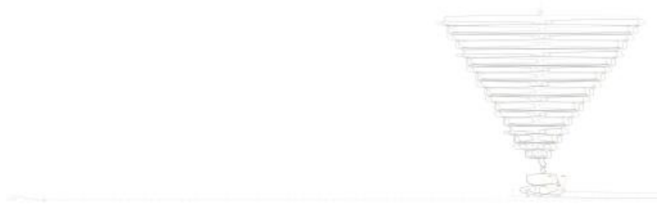


Figure 4.4: Right Zoom of the TensorBox graph, taken from TensorBoard.

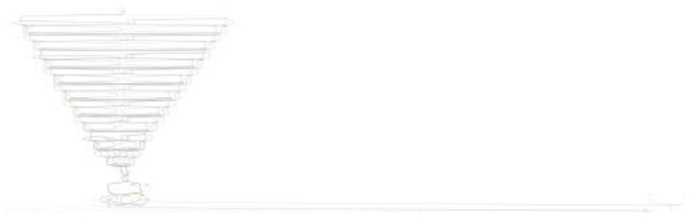


Figure 4.5: Center Zoom of the TensorBox graph, taken from TensorBoard; In the picture the reverse pyramidal graph is the main core of the computation for Overfeat.

given by the original project model, that in single-class gives great optical results on human faces detection. The high accuracy and impossibility to modify the project to be multiclass at all, pushed me to choose a modular configuration of the whole architecture and sacrifice accuracy training of the TensorBox model to be a general object detector.

The number of classes to detect is thirty for the VID challenge and not too high to drop down to 0 the accuracy of the detection. At the same time, the accuracy itself depends on the dataset it's learning from but also on the test set. In fact for clear snapped subjects the detection has always high confidence.

So I took the TensorBox project[44] and I trained it on the VID dataset[40]. Then I used the trained weights to fill them into the test model only to extrapolate

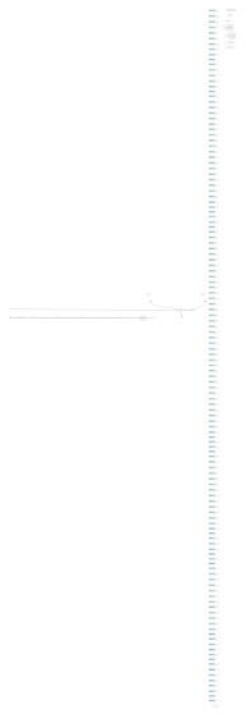


Figure 4.6: Left Zoom of the TensorBox graph, taken from TensorBoard; In the picture is presented the Auxiliary part, that are the nodes defined and then used in the main graph.

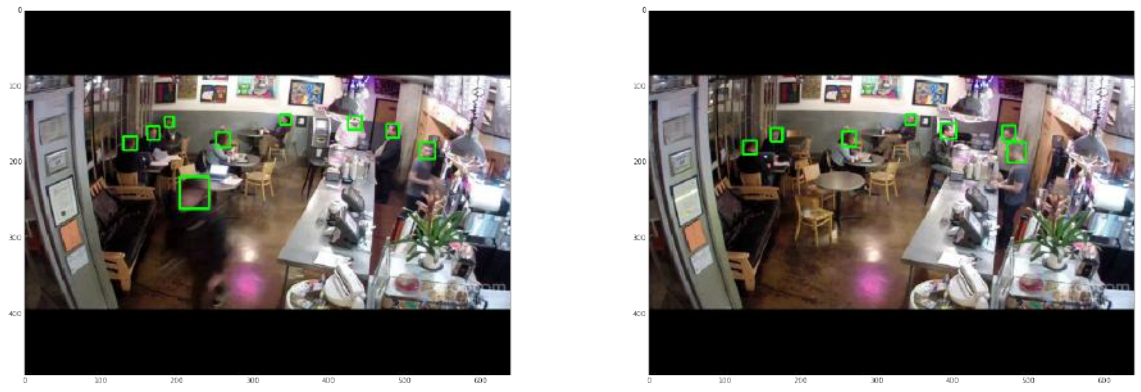


Figure 4.7: Results taken from the output of the GitHub project, TensorBox [44].

bounding boxes proposal for each frame, independently. I selected the bounding boxes with a dynamic threshold depending on the highest detection in the frame itself. Here I supposed that there was always a subject in any frame of the video. This is a strong hypothesis, but time constraint was ending and I had to simplify some concepts, scarifying accuracy.

I will describe deeply the choices I made on the dataset and on the train steps in the dedicated further sections of this chapter.

For my purpose, to develop a working model whose output was making sense, this was acceptable. The importance was not to use only the still image detections to achieve the goal but start making some computations considering the time and space dimensions, to reach better results.

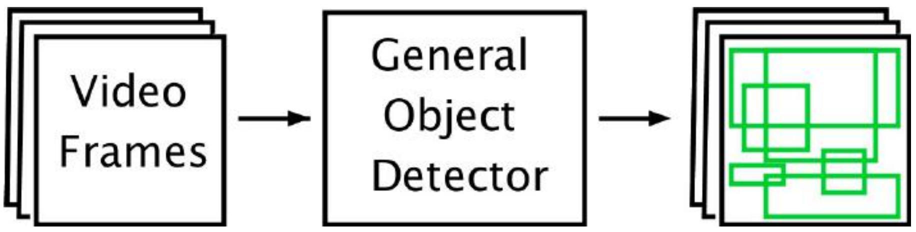


Figure 4.8: Illustration of the model process respect to the output of the first component. On the right side, we can see the Frames marked with Bounding Boxes Detections, which are a batch of rectangle that in space and in shape don't have too much sense right now.

The green boxes in the Figure 4.8 in the picture are the ones returned by the dynamic threshold selection. TensorBox [44] returns 300 bounding boxes for each frame, most of them with a really low detection threshold, rising up the ones more confident. The dynamic threshold takes all the bounding boxes that are in the range highest_value-0.15. This selection ensured me to have always at least one BBox detection (fixed thresholds sometimes return zero detection, also when there are visible objects), but more important to have a possibility to relate them between frames and choose the most appropriate ones.

In the next section, I'm going to expose my choices on the time and space analysis I made to perform a selection on the bounding boxes retrieved from the General Object Detector unit.

4.1.2 In time and Space Strategies

The hardest part began at this point, in which I had to understand how to implement and adapt to my project the slow and steady feature analysis, described by Dinesh Jayaraman and Kristen Grauman [31], in the fastest possible way or at least adapt some main concepts of the paper with a post-processing strategy.

In less than one month and half, I had to develop some sort of temporal and space strategies, without the time to build and train a network to learn it. This because building a network to learn "slow and steady features analysis" is not the only time consuming problem. Once built the model, the other difficulty was the training time needed, that for sure it was the longest I was encountering in the development. The time and space analysis wasn't the only task that the models had to apprehend, but also how to compensate the General Object Detector Errors, which is far harder than learning only the slow and steady analysis.

So, I decided to develop many greedy algorithms to post process the detections sequence of the frames, to smooth the object movements in time and space; then I chose the most theoretically correct one to use for the results submission.

The base of all the algorithm accuracy was the complete trust of the detections of the first two frames from which I started defining some different possible starting points for the detection:

- One object in the video;
- Multiple objects in the video.

From which I define some different starting selection strategies, respectively:

- Select the bounding box with the highest confidence and follow it ;
- Use a Non Maximum Suppression algorithm to retrieve the most confident rectangles and follow them.

The differences between the first two frames detections, were also defining:

- Shape changes deltas, due to objects rotation or movement on the backgrounds line;
- Position changes deltas, due to object movement on its background space.

I followed both the starting points for the detection, but taking only one object for video, was a too big restriction, with the lost many positive detection, useful for an output that have to make sense.

I ended up choosing the Second starting point, considering the Non Maximus Suppression output. It was also giving me the challenge and the chance, to give a track number to each object in the first frame and use it to build sequence of Rectangle in time; recognizing the different objects in the image that, at this point, was always at least one.

After some tests, I found the best Non Maximum Suppression algorithm value configuration, selecting the Overlapping Threshold in 0.3; in this way what I was loosing where only the really overlapping objects (like the monkey in the Figure 4.9).

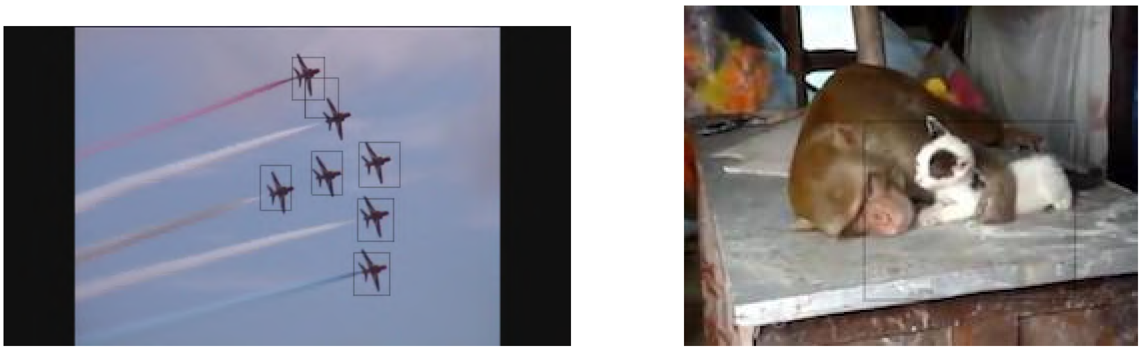


Figure 4.9: The first picture (a) show that the algorithm finds good results for multiple objects, which aren't so much overlapped; Instead the second (b) confirm that we are loosing the ones that are really overlapped, in this case is useful to note that the algorithm choose the most visible for the color, so the cat and not the monkey.

This was an acceptable starting point, applying this process for the First frame was giving me the possibility, to select in the second frame the most suitable rectangles between the ones proposed by the General Detector.

The second key point of the process was calculating and sharing deltas between the first two frames among all the other in the sequence. Here, I defined some different deltas, respect to the space I used the difference of the centers coordinates and respect to the shape the areas ratio. In this way comparing from the first frames detections to the last one I followed the trusted starting rectangles doing:

- Continue the object movement if not present, using center deltas to move the one of the considered box in the right direction;

- Insert rectangle objects if not present, starting from the previous detection and using the deltas difference to follow, as in the previous point, the right direction;

- Re-size the rectangle according to the shape deltas, following the founded ratio.

All this computations were made iff the highest IoU rectangle extrapolated from the detections doesn't respect a 0.2 error attenuation. If not, It's considered trustable and all the deltas are updated according to it, following the natural detection of the algorithm.

I tried to implement the basics for a smoother tracking object; the schema behind the principal idea shows the relations between inputs (upper part) and outputs (lower part):

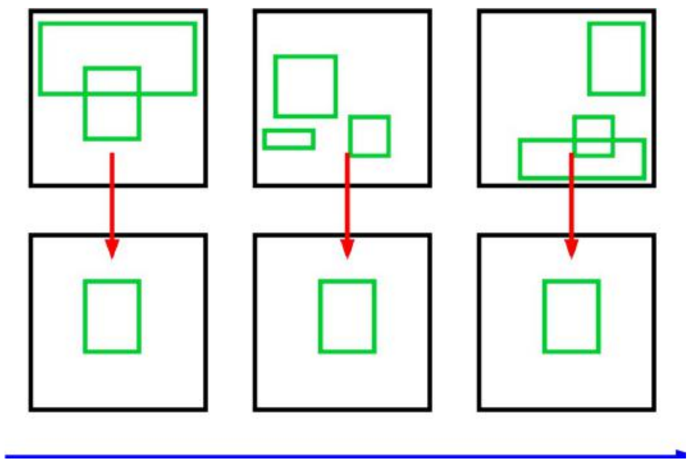


Figure 4.10: Inside process overview of the Object Tracker unit, we can see that in the first frame, the NMS output box is selected and the motion flux is followed into the further ones. The blue line shows the time line direction.

The outputs of the second unit are finally suitable to low-down the computation of useless information on the picture. So we are ready to label each sequence of rectangles with the next model unit.

4.1.3 Inception

The last architectural unit is the Image Classification. It is the least component, so it has only to compute the necessary runs to save time and computational power.

My first thought was to use a brute force approach, but due to the hypothesis I made in the earlier part of the model, I decided to continue with them and save as much time as possible.

The model I used for Inception v3 model [49] is directly provided by Google [1] in the GitHub repository [20]. Some examples of basic[18] and advanced[19] use of the model, given by Google, are also available, but is this developers tutorial[6], that was fundamental to follow to implement the needed model.

This latter didn't use the ImageNET dedicated training code available on the official repository [20], which results to be more accurate when multiple objects images are used to train the model.

To ensure that the lost accuracy in the training was compensated by something, I thought to crop the tracker detections for each frame needing the objects labels and use them as input for the classification. In this way I have almost always, a single subject image to ensure to identify the right object class also in multi-class images. I Classify the object only in the First two frames and select the highest

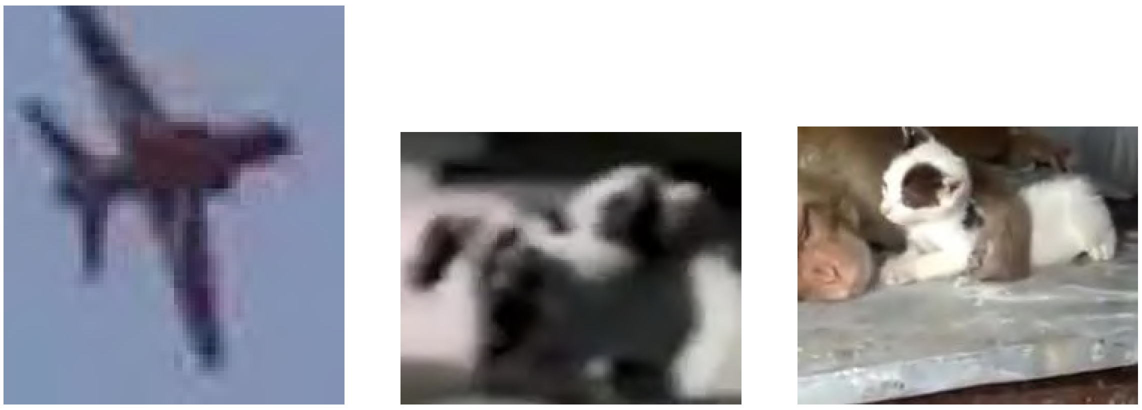


Figure 4.11: Example of cropped frames taken from a run of the thesis model, from Left to Right in order: airplane, dog, cat.

class of the combined results and then I share it with all the others rectangles of the same Track ID.

Differently from the TensorBox graph the one of Inception is consistent with the one presented in the state of the art and respects the design laws described in the Google paper; here below the representation taken from TensorBoard: The last remaining pieces of the puzzle are the description of the code structure, the Dataset I used and finally the training process I made.

In the Figure 4.13 we can see the schema that describes the whole process that takes place in the model, from video to labeled frames.

4.2 Code Specifications

To develop the code in the most incremental way, I used my software engineering knowledge to structure the project in the best way to save time and coding.

I folded in basic, utilities and interface classes to make the final script more readable and slim. In the Figure 4.14 we can see the basic classes I implemented. Starting from the base unit for the detection, that is the Rectangle, I took the one implemented into TensorBox project [44] that was singleclass and extended it to multiclass, making it suitable to store and be used with the ImageNet challenge. I did this adding: all the label properties (label_confidence, label, label_chall, label_code), extended the setting and the getting functions to work with the added properties and finally some useful functions to retrieve the string formats for the submission of the results. The two outside class-scope functions, are the ones used to pop the best

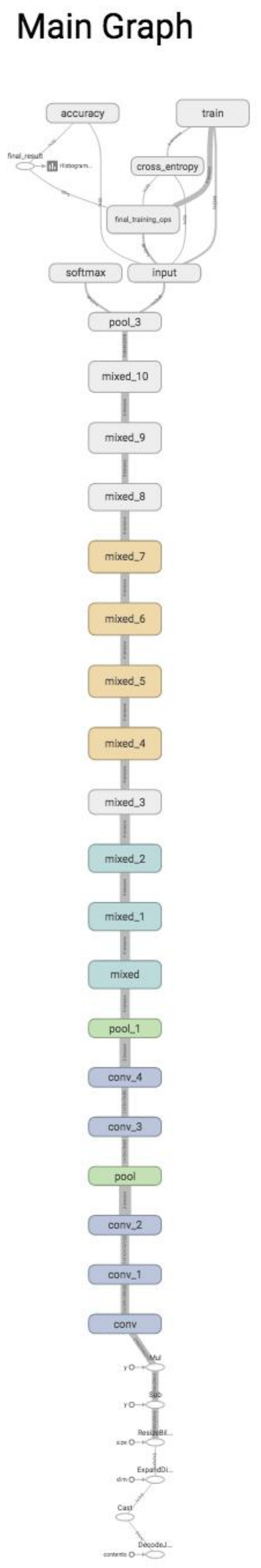


Figure 4.12: Inception v3 graph, taken from TensorBoard, the singles node are inside composed as showed in the state of the art chapter.

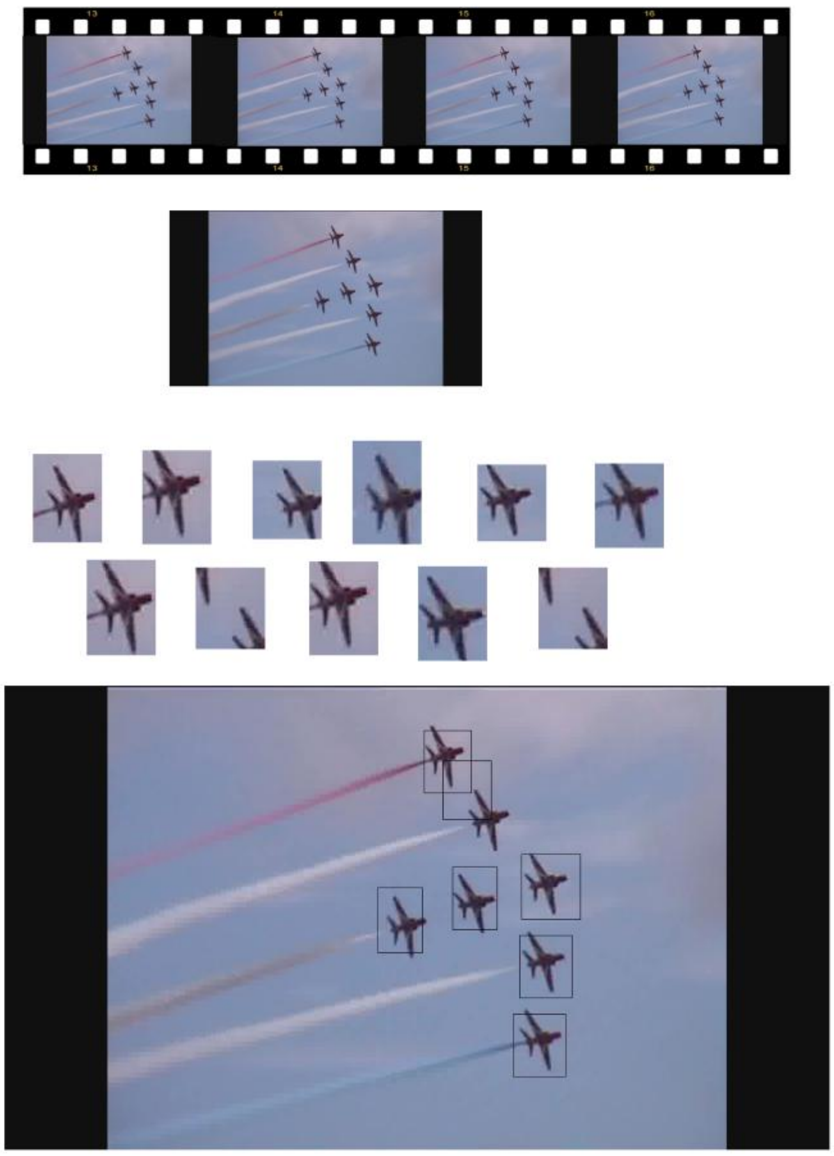


Figure 4.13: Whole model process that starting from the video, took one frame, computed object proposal, cropped the rectangles and finally chose the best ones and tracked them on the frame.

rectangle during the tracking (I used the IoU but I implemented also the Overlap one).

The other really important developed class is the Class List stored into the VID_classes.py file. This one contains only four properties that are arrays storing the main related information on: Class String Name, Class Code CLC, Class Code VID and finally the colors to use for bounding boxes related to each Class. The order of the arrays follows the VID Class Code (1 to 30) minus one, so from 0 to 29. In parallel, in the same file I defined some outside class-scope functions, that are only switcher to retrieve easily values related on a single class.

The last basic class is the Frame_Info, that I made to keep track all the informa-

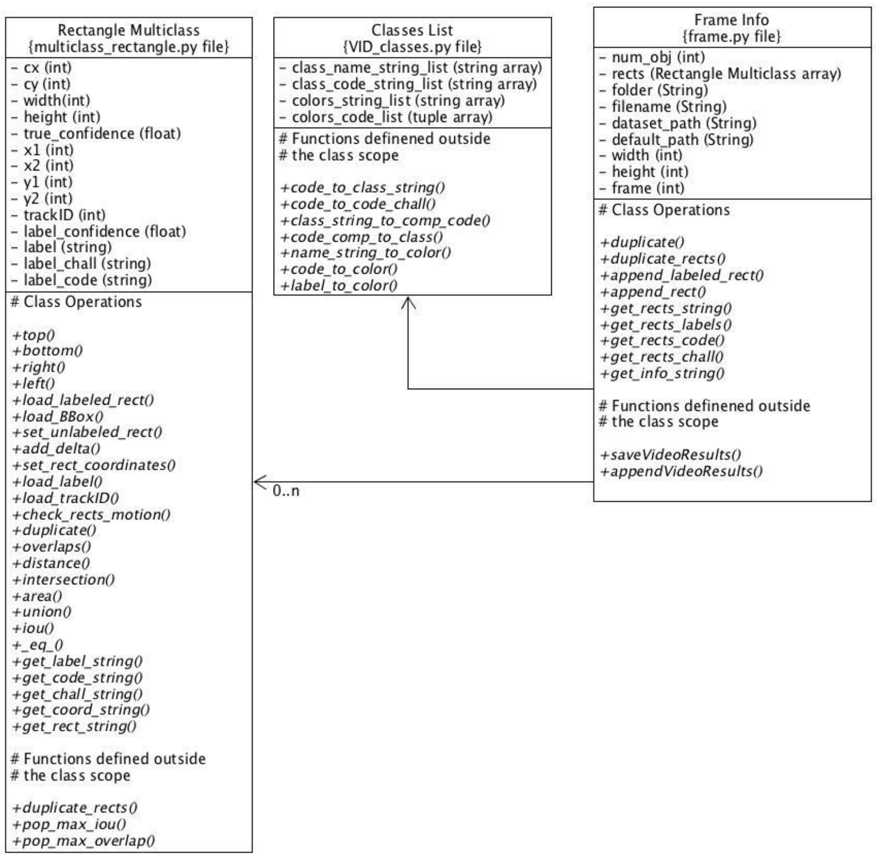


Figure 4.14: UML diagram of the basic classes implemented into the project.

tion related to a frame. This could be considered the basic info class for a video, that is composed by many frames. This class have a single Class_List instance and it can have many of the Rectangle_Multiclass. This relations are easy to understand analyzing the basic properties of the considered class: num_obj, rects[], folder (where the frame is stored), filename (frame filename), dataset_path (path of the dataset where the folder is stored), default_path (extra variable to store special path if needed), width height (sizes of the frame), frame (frame number). The rects[] is the variable containing all the Rectangle_Multiclass instances. The Class_List instance instead is only used into the implemented functions, to relate useful information.

These functions are high level operations some of which are wrapping the single rectangle functions written into the Rectangle_Multiclass. Also here outside class-scope operations are presented and mainly work to save and append the video results to a file.

Then there are the utils classes, implemented to slim the code in the interface classes. Here below it's possible to see their UML class diagram.

I developed utils classes related to each unit I was working with, so:

- system (utils)
- images (image_utils)
- videos (video_utils)

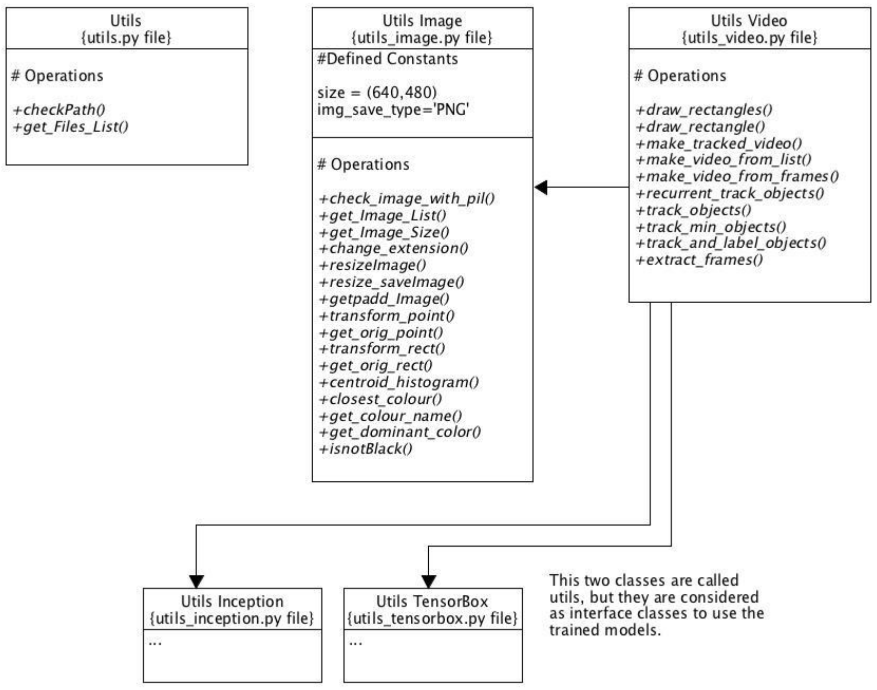


Figure 4.15: UML diagram of the utils classes implemented into the project.

The first one is really small and contains only two useful functions, to check file existence and get files list from a folder path.

The image_utils file defines two constant, a size tupla and a save type format both useful to work with TensorBox that takes only (640,480) images and prefers ".PNG" type format. This class contains also fundamental functions to relate results of TensorBox to original size image like:

- getpadd_Image
- transform_point
- get_orig_point
- transform_rect
- get_orig_rect

Finally there is the video_utils that is the only one that use instances of others utils files. I also add the two "Utils" files (Inception and TensorBox) to the schema , see Figure4.15, to highlight that they are used from the video_utils class, mainly called for some useful operations defined in them. This latter utility file is the one that contains the functions to track object inside the video:

- recurrent_track_objects
- track_objects
- track_min_objects
- track_and_label_objects

The first function is the one I used for the final submission results.

Finally, the Interface "Utils" classes I implemented to work with the trained models and their results, can be considered as interface to use the projects code and at the same time utils to use inside my project.

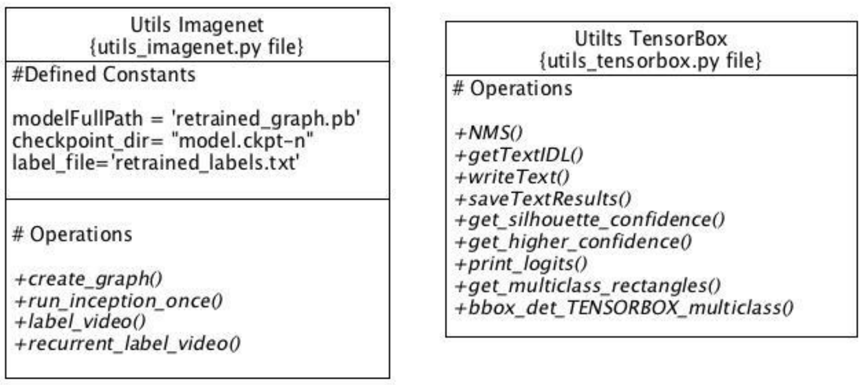


Figure 4.16: UML diagram of the interface classes implemented into the project.

The utils_tensorbox defines functions not only to load and write the files to train the model, but also the Non-Maximum-Suppression algorithm and the most important function bbox_det_TENSORBOX_multiclass, that computes object detection for all the frames of the video.

On the contrary, the utils_inception, defines some constants referring to the trained model in the original code project. Then, with the create_graph operation I loaded the graph and the weights, and the run_inception_once is the basic function to label images, which I used into the more complex label_video and recurrent_label_video, that really label the detected object in the video.

So structured, the final code of the VID script is the following:

```
import argparse
import utils_image
import utils_video
import Utils_Tensorbox
import Utils_Imagenet
import frame
import vid_classes
import progressbar
import time
import os
def read_test_textfile(filename , data_folder):
    videos_list = []
    video_frame =[]
    with open(filename) as f:
        video=None
        for line in f:
            file , idx = line.strip().split(' ')
          new_file='ILSVRC2016_'+file.split('_')[1]+'_'+file.split('_')[2]
```

```
            file=new_file
      print 'File name:\%s'\%file
      if video is None:
                video_frame.append((data_folder + file + '.JPEG',idx))
                video=file.split('/')[0]
            else:
                if video==file.split('/')[0]:
                    video_frame.append((data_folder + file + '.JPEG',idx))
                else :
                    videos_list.append(video_frame)
                    video_frame = []
                    video_frame.append((data_folder + file + '.JPEG',idx))
                    video=file.split('/')[0]
    return videos_list
def main():
    tot_start = time.time()
    parser = argparse.ArgumentParser()
    parser.add_argument('--data_folder', default='/VID/test/', type=str)
    parser.add_argument('--file_path', default='VID/test.txt', type=str)
    parser.add_argument('--hypes', default='overfeat_rezoom.json', type=str)
    parser.add_argument('--weights', default='save.ckpt-1500000', type=str)
    parser.add_argument('--perc', default=5, type=int)
    parser.add_argument('--path_file_output', default='output.txt', type=str)
    args = parser.parse_args()
    video_list_test = read_test_textfile(args.file_path , args.data_folder)
    for video in video_list_test:
        start = time.time()
        decomposed_path=video[0][0].split('/')
        path_video_folder = decomposed_path[len(decomposed_path)-2]
        print "Path Video Folder: \%s"\%path_video_folder
    os.makedirs(path_video_folder)
        print"Created Folder: \%s"\%(path_video_folder)
        progress = progressbar.ProgressBar(widgets=[progressbar.Bar('=', '[', ']'), ' ',progressbar.Percentage(), ' ',progressbar.ETA()])
    frame_inception=[video_info[0] for video_info in video]
        frame_tensorbox=[]
        print "Creating Tensorbox Files for video:\%s"\%path_video_folder
        for image_path in progress(frame_inception):
            frame_tensorbox.append(utils_image.resize_saveImage(image_path, path_video_folder))
        video_info=Utils_Tensorbox.bbox_det_TENSORBOX_multiclass(frame_tensorbox, args.hypes, args.weights)
        tracked_video=utils_video.recurrent_track_objects(video_info)
    labeled_video=Utils_Imagenet.recurrent_label_video(tracked_video, frame_inception)
    frame.appendVideoResults(args.path_file_output , video,labeled_video)
        end = time.time()
        print("Elapsed Time:\%d Seconds"\%(end-start))
        print("Running Completed with Success for video: \%s!!!")\%path_video_folder
    tot_end = time.time()
    print("Elapsed Time:\%d Seconds"\%(tot_end-tot_start))
    print("Completed the whole Test with Success!!!")
if __name__ == '__main__':
    main()
```

The script defines a function to read the text test file and in 83 lines achieves a slim and clear implementation.

4.3 Dataset Processing

The dataset, as I said many times, is the primary quality source of the learning.

Understand its quality and how the model learn from it, are key factors, which sometimes are underestimate and that sometimes can be improved or fixed easily.

The databases [40] provided by Imagenet [43] hosts, are different for each challenge, from Classification to Scene Parsing. The order of dimension for the whole database for the VID challenge is around 100 Gb (Train + Validation + Test).

Manage that amount of database, might seem easy, but when its on a server-side, it's not; especially, if you have run errors and you would like to check visually the data. For this reason, the database is provided with a full text of instructions that the user can apply to understand its structure, weakness and strength points, but more important how to use it for the development.

This latter was my main problem during the work. The database is redundant, it has video snippets, extracted frames, which are the ones belonging to the annotations files. Into it are also provided ".txt" files that link the train, the validation and the test database.

Due to my inexperience about the topic and the main tricks to use to speed up the development with the database learning, I didn't use the yet prepared ".txt" files but I generated them by myself, clustering the whole dataset differently from the provided structure, in the way TensorBox[44] was requiring.

The provided structure wasn't divided in classes, but mixed and I clustered them for type to have a best personal view. This decision took me time and effort, instead from the learning point of view is better a mixed database to perform a longer train that is less over-fitted on specific object shapes. Precautions such as this, could be one of the first improvements for the project.

The work I made on the database wasn't a waste of time, because I learned directly from the experience and now I have a better and clear overview on their importance and I know the best practices to use with them to boost accuracy learning.

The competition rules on the database are easy. There are two different submission categories, depending on the used dataset:

- Use only the Train dataset of the competition, with any strategy ratio;
- Use as base the Train dataset, then use other dataset, as for example the Validation from the same challenge of ILSVRC;

I submitted only the first one, due to time problems and few experience. Some of the main used strategies for this submission type, because of the nature of the challenge, are to use not all the frames but for example, one every two or three. So we can reduce the overfitting for the classification unit and train on more "difficult" data the tracker and the object detector. This because, the differences between following frames are too small that can cause overfitting to the object shape and reduce the real possibility to learn a true task applicable to test data.

This strategy was employed by K. Kang et al [32], who used a really smart precaution also for the second type of submission. They took the DET dedicated dataset, to boost recognition, localization and detection accuracy mixing it with the VID one with the following ratio: VID:DET=2:1. They didn't gain the first place in last year competition, but they showed how this boost their results. Tricks like this are unforgettable.

With the dataset is provided the Development Kit with the instructions I mentioned before, and some routines to evaluate the developed model.

For VID competition the basic unit is the snippet, which contains many frames and a Video might contain several snippets. There were three sets of snippets data and objects labels:training, validation and test; with no overlap between the three sets. The validation and test datasets of this year, were partially refreshed by the hosts:

As the instructions explain, most of the videos are queried using Youtube API with manually selected topicIDs from Freebase. These latter are specifically related to a particular synset (Cartoon and unnatural videos are not considered).

All visible objects from all frames in each snippet are exhaustively annotated with bounding box with the use of VATIC. Two types of boxes are provided: the manually

Number of snippets			
Dataset	**TRAIN**	**VALIDATION**	**TEST**
VID	3862	555	937

Table 4.1: Number of snippets in each dataset[40] provided for the ILSVRC VID competition.

annotated ones, that are mostly tight with the visible portion of the object, and the others generated using linear interpolation. One of the most important note to take care of in the instruction is the following: *"Due to the scale of the data, a small subset of frames in each snippets may nevertheless have errors in the annotations."* This highlights how datasets are to be considered carefully.

The object categories in the VID dataset are 30 classes: those which can move, such as 'airplane' and 'dog'; whereas CLS-LOC and DET contain more static classes: 200 in total, e.g. 'chair' and 'backpack'.

Some other useful statistics to relate snippets number, with the numbers of images and boxes are in the table below.

Datasets Statistics			
Dataset	TRAIN	VALIDATION	TEST
Snippets	3862	555	937
Frames	1122397	176126	-
Annotated Objects	866870	135949	-

Table 4.2: The Training dataset has median of 116 positive snippets per synset and 180 frames per snippet; the Validation dataset has a median of 17 snippets per category and 232 frames per snippet; the Test dataset information aren't released.

The structure in which the datasets are organized is different. For the train one there are thirty different ".txt" files, each one corresponding to a label:

ImageSets/VID/train_1.txt

...

ImageSets/VID/train_30.txt

In these files, every line contains just a snippet_id and a label. The snippet frames are stored in a directory as JPEG images with naming like snippet_id/%06d.JPEG, and the corresponding annotations are snippet_id/%06d.xml.

The train snippets are also divided in four batch: ILSVRC2015_VID_train_0000

ILSVRC2015_VID_train_0001

ILSVRC2015_VID_train_0002

ILSVRC2015_VID_train_0003

Each one contains at most 1000 snippets.

Validation and Test dataset are organized in the same way, but with a single-batch structure. The instructions also describe the submission format for the challenge, which is a ".txt" file that must be filled with this line format:

frame_index ILSVRC2015_VID_ID confidence xmin ymin xmax ymax

Frame_index corresponds to the frame index from val.txt or test.txt, depending on which set of images is being annotated. The change in 2016 competition was the possibility to submit tracked or not tracked object detections. I submitted only the one with the tracking id, but I post validated only the not tracked one.

The available routines for evaluating a submission for the detection task were in Matlab and only the one without tracking was provided. The hosts also gave some submission examples, to better understand the whole structure and kit usage.

To clarify the comparisons I made in the state of the Art chapter it is important to understand some differences between the databases I presented above, the ImageNet and the PASCAL VOC, on which many models were tested. As we can evince from the PASCAL VOC website [56], the database provided contains:

- 20 classes;
- Train and Validation data has 11,530 images,containing 27,450 ROI annotated objects and 6,929 segmentations.

This numbers are pretty far from the ones of the VID, CLC and LOC. So the results I showed in the second chapter, must be taken carefully, because with a bigger dataset, results can really change a lot due to different photos and object exposure, which cause a different learning.

4.4 Training Specifications

In this chapter I will highlight the training process of the Inception and TensorBox components of the model.

4.4.1 TensorBox Training

The TensorBox project takes an ".idl" file for training. It must contain a line for each image, with the relative bounding boxes objects kept into the picture. The batch size was related to the grid size of the model, becoming 300 (15x20) with a learning rate of 0.001. Here some extrapolation from TensorBoard of the training phase for the model over 2M iterations.

As we can see from the graphs in the Figure 4.17, all the functions become flat, but still have sparse results and peaks out of the flat value. This is due to the generalization of the model, that is not learning different masks for each object category, but only one for all of them. This lowered the whole accuracy.

As the Figure 4.18 shows, the weights of the model after 1.6M iterations, come immediately flat. This means that the model wasn't learning any more.

It's useful to see now the Test graphs available in TensorBoard which give us a better awareness of the previous graphs and the real accuracy of the model in test phase.

It becomes immediately clear from the Tests graphs in the Figure 4.18, that only the loss functions are the most flatted, instead the accuracy still be sparse and with many peaks.

TensorBoard gives the possibility to observe the test phase also on the images, giving a more clear view on the accuracy of the model, here below an example: The test image above shows how really the model is learning, where it becomes easier

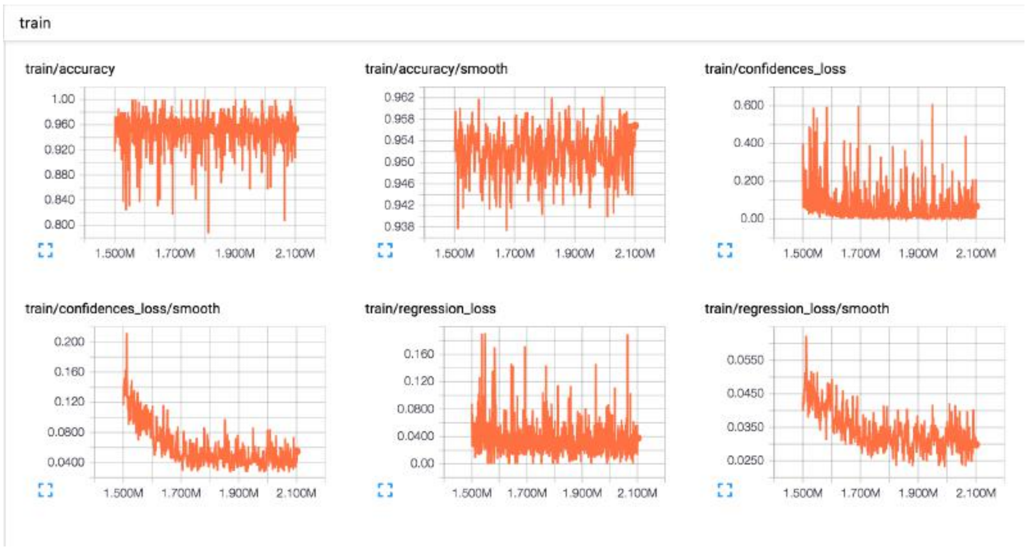


Figure 4.17: Event training graphs, extrapolated from TensorBox

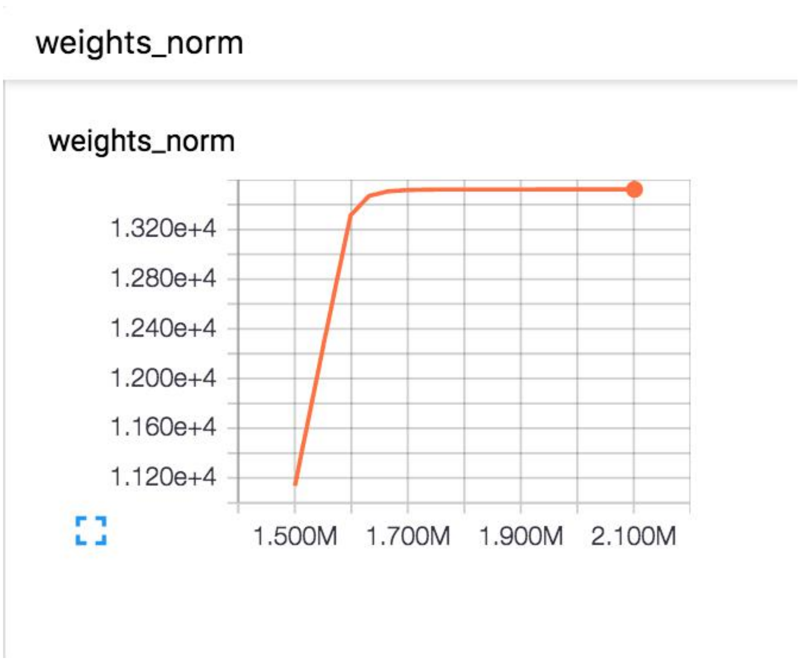


Figure 4.18: Training weights graphs, extrapolated from TensorBox

to detect objects in monochrome backgrounds (like airplanes into the figure of the whole process), but training to detect generalized object, where this condition is not satisfied, gives us a lower accuracy, more if overlapped object are present.

4.4.2 Inception Training

Differently from TensorBox, Inception is more lightweight and could be run also on a Personal Computer without too much effort. More over it needs really less training steps than the previous component, because it arrives yet trained for ImageNet CLC. The train scope becomes adding specific layers for the new recognition task, but using the previous learned knowledge for CLC.

I trained the model for 125K iteration steps with a train batch of 500 images

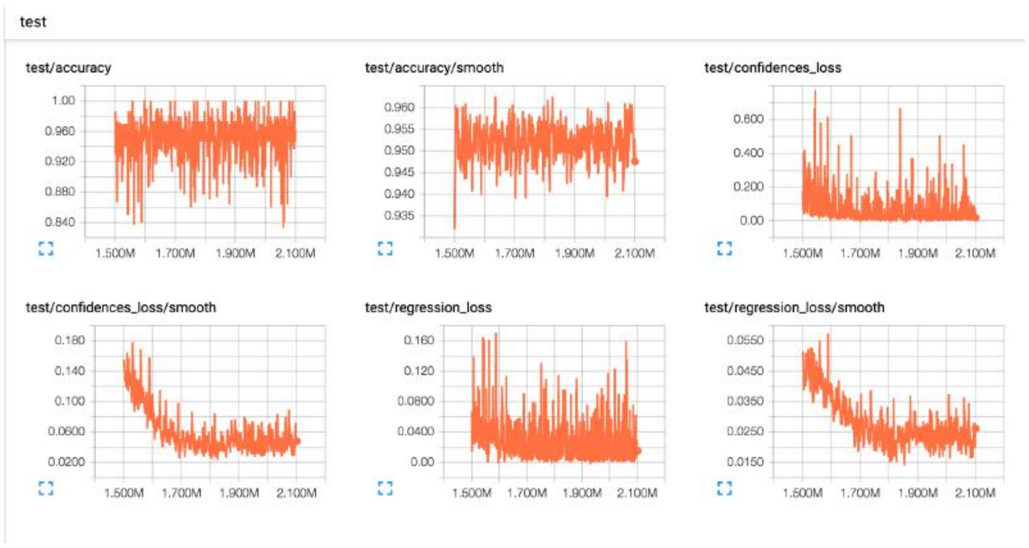


Figure 4.19: Event training graphs, extrapolated from TensorBox

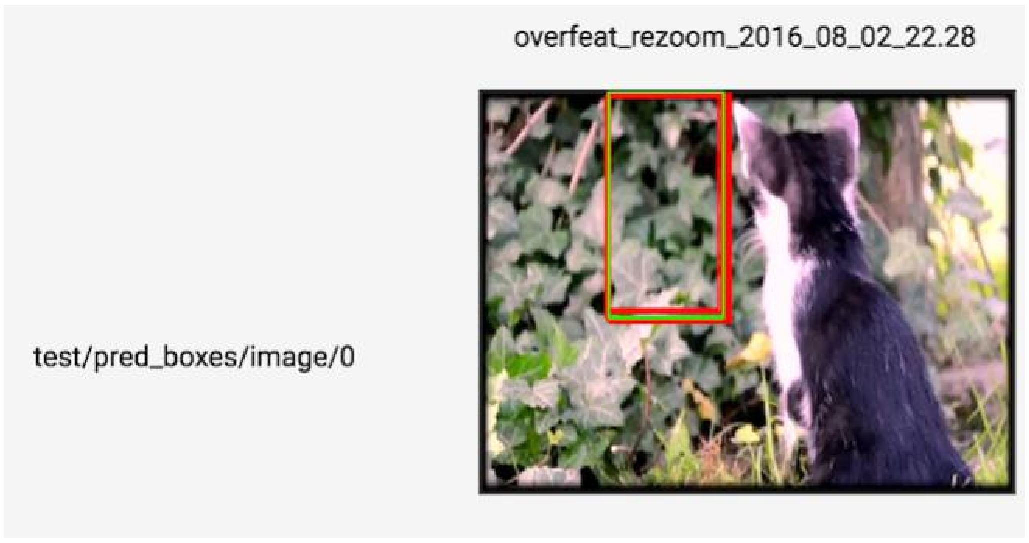


Figure 4.20: Test Image extrapolated from TensorBox

and a learning rate of 0.001.

Here below the graphs of the training accuracy and cross-entropy cost function: The Figures 4.21 highlight that both functions become flat almost immediately, because the model only has to focus on 30 classes of 1000 objects it yet knows. To confirm this aspect we can look at the Figures 4.22 below, where the Standard Deviation of weights and biases highlight that the graph learns well.

As for TensorBox, we can see train/validation graphs to confirm the training ones.

Differently from the TensorBox test graphs, the Inception validation ones shows less out of the range peaks and a more stable trend.

In the next chapter we will make some evaluations on the work and then the conclusions.

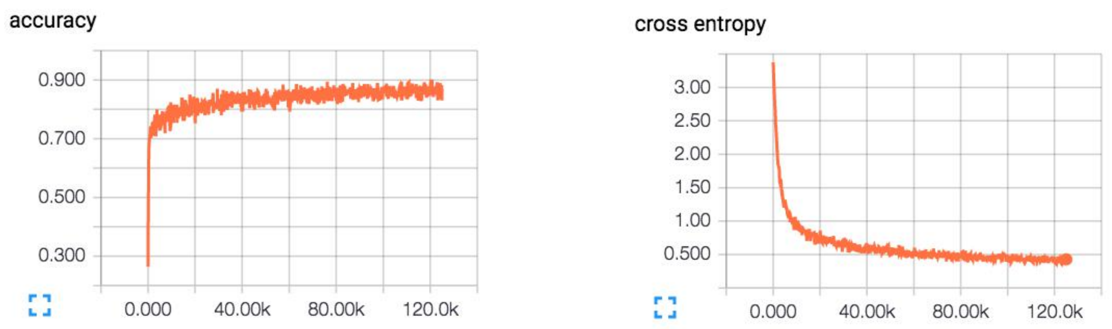


Figure 4.21: Graphs of Accuracy and Cross Entropy cost function, taken from TensorBoard.

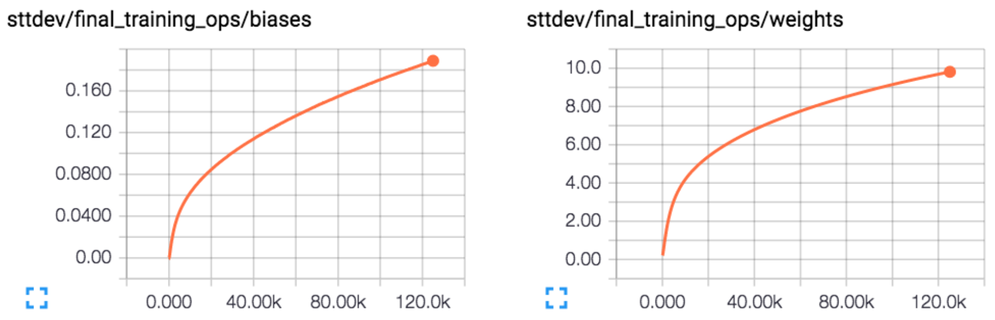


Figure 4.22: Standard Deviation graphs of Biases and Weights values of the model, extrapolated from TensorBox.

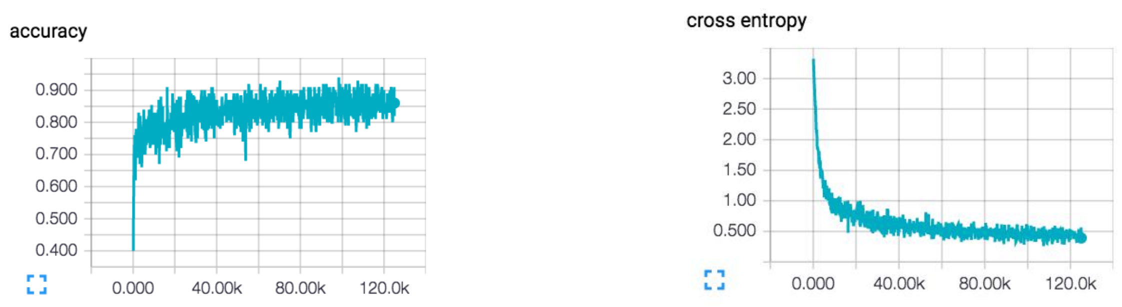


Figure 4.23: Graphs of Accuracy and Cross Entropy cost function, taken from TensorBoard.

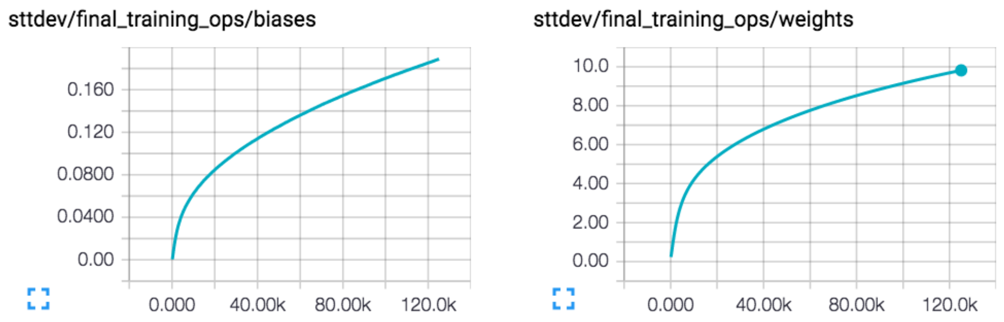


Figure 4.24: Standard Deviation graphs of Biases and Weights values of the model, extrapolated from TensorBox.

Chapter 5

Evaluation of the Work

This section concerns the evaluation of the results I got from the model for the Validation and Test dataset, showing also the results of the VID ILSVRC 2016 . Then I will resume some evaluation on the asterix infrastructure performance and adaptation to the TensorFlow environment.

5.1 ImageNet Challenge Results

The 2016 challenge winners are listed in the following Tables 5.1 5.2 5.4 5.5, according to the training data and to the detection+tracking or not of the results.

Team name	**Entry description**	**Number of object categories won**	**mAP**
NUIST	cascaded region regression + tracking	10	0.808292
NUIST	cascaded region regression + tracking	10	0.803154
CUVideo	4-model ensemble with Multi-Context Suppression and Motion-Guided Propagation	9	0.767981
Trimps-Soushen	Ensemble 2	1	0.709651

Table 5.1: Object detection from video with provided training data, Results from ILSVRC 2016.

As we can see there is a big difference of 0.22 mAP between the Object Detection in Video, and the Object detection with tracking in Video. The latter is the one in which I participate with the model and I was introduced only this year to the competition.

More over the additional training data raise up the score only of the detection plus tracking category of a 0.03 mAP that is not so significant at all; and for the other contrary category to the last year trend there is a lost of the mAP. Here below the results of last year competition allow a better overview of the challenge trend.

This year's results respect to the 2015's have been increased from 0.67 mAP to 0.8 mAP with a boost of 0.13 in the accuracy for the challenge with only provided

Team name	**Entry description**	**Number of object categories won**	**mAP**
NUIST	cascaded region regression + tracking	17	0.79593
NUIST	cascaded region regression + tracking	5	0.781144
Trimps-Soushen	Ensemble 6	5	0.720704
ITLab-Inha	An ensemble for detection, MCMOT for tracking	3	0.731471

Table 5.2: Object detection from video with additional training data, Results from ILSVRC 2016.

Team name	**Description of outside data used**
NUIST	proposal network is finetuned from COCO
NUIST	proposal network is finetuned from COCO
Trimps-Soushen	Extra data from ImageNet dataset (out of the ILSVRC2016)
ITLab-Inha	pre-trained model from COCO detection, extra data collected by ourselves (100 images per class)

Table 5.3: Extra Training Data information refered to the previous table of results from ILSVRC 2016.

Team name	**Entry description**	mAP
CUVideo	4-model ensemble	0.558557

Table 5.4: Winner of the Object detection/tracking from video with provided training data of ILSVRC 2016, the second place was taken from NUIST Team, that was the winner of all the others category.

Team name	**Entry description**	**Description of outside data used**	**mAP**
NUIST	cascaded region regression + tracking	proposal network is finetuned from COCO	0.583898

Table 5.5: Winner of the Object detection/tracking from video with additional training data of ILSVRC 2016.

data, instead for the category with additional data has a lower improvement from 0.73 mAP to 0.79 mAP.

The winner with the highest mAP is the NUIST team, that in the specification entries about their model they wrote:

- ”inception v2 is used in the VID task, which is almost real time with GPU.”
- ”cascaded region regression is used to detect and track different instances.”
- ”context inference between instances within each video.”

Team name	**Entry description**	**Number of object categories won**	**mAP**
CUVideo	Average of models, no outside training data, mAP 73.8 on validation data	28	0.678216
RUC_BDAI	We combine RCNN and video segmentation to get the final result	2	0.359668

Table 5.6: Object detection from video with provided training data, Results from ILSVRC 2015.

Team name	**Entry description**	**Number of object categories won**	**mAP**
Amax	only half of the videos are tracked due to deadline limits, others are, only detected by Faster RCNN (VGG16) without tempor smooth.	18	0.730746
CUVideo	Outside training data (ImageNet 3000-class data) to pre-train the detection model, mAP 77.0 on validation data	11	0.696607
Trimps-Soushen	Models combine with main object constraint	1	0.480542

Table 5.7: Object detection from video with additional training data, Results from ILSVRC 2015.

Team name	**Description of outside data used**
Amax	–
CUVideo	ImageNet 3000-class data to pre-train the model
Trimps-Soushen	COCO

Table 5.8: Extra Training Data information refered to the previous table of results from ILSVRC 2015.

- ”online detector and tracker update to improve recall .”

Their specification highlights which components for a model are the most important as: inception for visual recognition, a detector, a tracker, and the cascade region regression to detect and track. This team is formed by a part of Amax team that won last year competition.

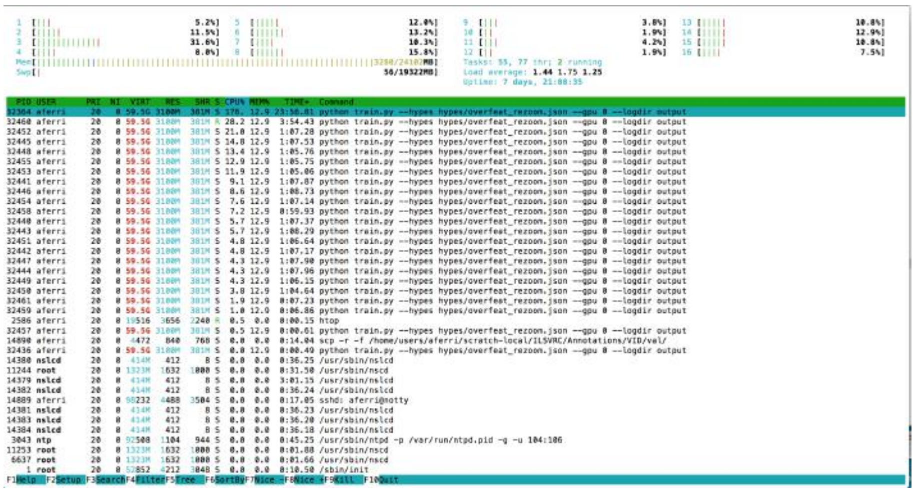
This aspect highlights how many teams are working for years to implement and improve their model, trying to overcome new limits.

5.2 Computational Evaluation

For its modular nature, the model I developed has different computational workloads on the infrastructure. As I anticipated in the previous chapter the Inception v3 workload is so low because of its great implementation, that could be trained and tested for 30 object classes also on a Personal Computer.

It is the TensorBox component that has affected the whole model workload a lot. Due to its unbalanced model implementation and the imperfect use of TensorFlow library, it has a really high usage of CPUs and GPU.It works only with one graphic process unit and not with many in parallel.

The importance of the parallelization for some models is really more important than for others, TensorBox workload needs are enormous and with a parallel implementation, the model could been faster and lighter to train and test,instead it could be easily trained and tested only at server side.



```
  1  [|||                          5.2%]   5  [|||||                        12.0%]   9  [|||                          3.0%]   13 [||||||                       10.8%]
  2  [|||||                       11.5%]   6  [||||||                       13.2%]   10 [||                           1.9%]   14 [||||||                       12.9%]
  3  [|||||||||||||               31.6%]   7  [||||                         10.3%]   11 [|||                          4.2%]   15 [||||||                       10.8%]
  4  [||||                         8.0%]   8  [||||||                       15.8%]   12 [||                           1.9%]   16 [|||||                         7.5%]
  Mem[||||||||||||||||||||||||||||||||||||||||||||||||||||||3280/24102MB]   Tasks: 55, 77 thr; 2 running
  Swp[|                                                  56/19322MB]   Load average: 1.44 1.75 1.25
                                                                        Uptime: 7 days, 21:08:35

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
32364 aferri     20   0 59.5G 3100M  381M S 178. 12.9 23:50.81 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32460 aferri     20   0 59.5G 3100M  381M R 28.2 12.9  3:54.43 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32452 aferri     20   0 59.5G 3100M  381M S 21.0 12.9  1:07.28 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32445 aferri     20   0 59.5G 3100M  381M S 14.0 12.9  1:07.53 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32448 aferri     20   0 59.5G 3100M  381M S 13.4 12.9  1:05.76 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32455 aferri     20   0 59.5G 3100M  381M S 12.9 12.9  1:05.75 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32453 aferri     20   0 59.5G 3100M  381M S 11.9 12.9  1:05.06 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32441 aferri     20   0 59.5G 3100M  381M S  9.1 12.9  1:07.87 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32446 aferri     20   0 59.5G 3100M  381M S  8.6 12.9  1:08.73 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32454 aferri     20   0 59.5G 3100M  381M S  7.6 12.9  1:07.14 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32458 aferri     20   0 59.5G 3100M  381M S  7.2 12.9  0:59.93 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32440 aferri     20   0 59.5G 3100M  381M S  5.7 12.9  1:07.37 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32443 aferri     20   0 59.5G 3100M  381M S  5.7 12.9  1:08.29 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32451 aferri     20   0 59.5G 3100M  381M S  4.8 12.9  1:06.64 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32442 aferri     20   0 59.5G 3100M  381M S  4.8 12.9  1:07.17 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32447 aferri     20   0 59.5G 3100M  381M S  4.3 12.9  1:07.90 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32444 aferri     20   0 59.5G 3100M  381M S  4.3 12.9  1:07.96 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32449 aferri     20   0 59.5G 3100M  381M S  4.3 12.9  1:06.15 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32450 aferri     20   0 59.5G 3100M  381M S  3.8 12.9  1:04.64 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32461 aferri     20   0 59.5G 3100M  381M S  1.9 12.9  0:07.23 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
32459 aferri     20   0 59.5G 3100M  381M S  1.0 12.9  0:06.86 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
 2586 aferri     20   0 19516  3656  2240 R  0.5  0.0  0:00.15 htop
32457 aferri     20   0 59.5G 3100M  381M S  0.5 12.9  0:00.61 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
14890 aferri     20   0  4472   840   768 S  0.0  0.0  0:14.04 scp -r -f /home/users/aferri/scratch-local/ILSVRC/Annotations/VID/val/
32436 aferri     20   0 59.5G 3100M  381M S  0.0 12.9  0:00.49 python train.py --hypes hypes/overfeat_rezoom.json --gpu 0 --logdir output
14380 nslcd      20   0  414M   412     8 S  0.0  0.0  0:36.25 /usr/sbin/nslcd
11244 root       20   0 1323M  1632  1000 S  0.0  0.0  0:31.50 /usr/sbin/nscd
14379 nslcd      20   0  414M   412     8 S  0.0  0.0  3:01.15 /usr/sbin/nslcd
14382 nslcd      20   0  414M   412     8 S  0.0  0.0  0:36.24 /usr/sbin/nslcd
14889 aferri     20   0 98232  4488  3504 S  0.0  0.0  0:17.05 sshd: aferri@notty
14381 nslcd      20   0  414M   412     8 S  0.0  0.0  0:36.23 /usr/sbin/nslcd
14383 nslcd      20   0  414M   412     8 S  0.0  0.0  0:36.20 /usr/sbin/nslcd
14384 nslcd      20   0  414M   412     8 S  0.0  0.0  0:36.18 /usr/sbin/nslcd
 3043 ntp        20   0 92508  1104   944 S  0.0  0.0  0:45.25 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 104:106
11253 root       20   0 1323M  1632  1000 S  0.0  0.0  0:01.88 /usr/sbin/nscd
 6637 root       20   0 1323M  1632  1000 S  0.0  0.0  0:01.66 /usr/sbin/nscd
    1 root       20   0 52852  4212  3048 S  0.0  0.0  0:10.50 /sbin/init
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Figure 5.1: CPU usage of the TensorBox component of the model.

As we can see from the Figure 5.1 representing the CPU usage panel, TensorBox program creates many threads, 22 to be precise and tries to use as much memory as available. It's the infrastructure middle-ware itself that balances the workload on the whole set of CPUs.

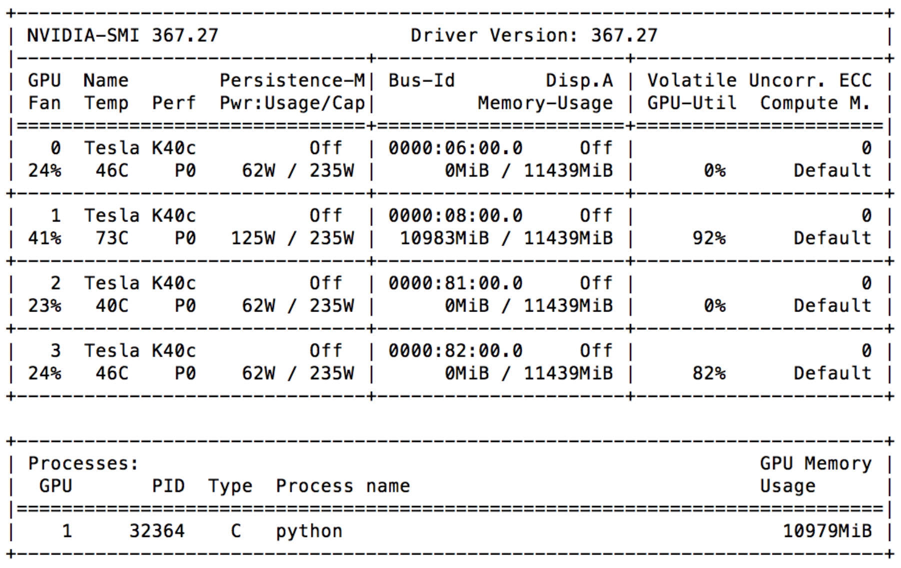
Instead for the GPU, the script can run only on one a time, using 92% of its power and memory, losing parallelization potentialities. It becomes clear how much the TensorBox model implementation is power consumption and how much the performances could be boosted and computation needs lowered, if it has a better implementation .

5.3 Model Evaluation

The model I made reaches the last position of the Object Detection and Tracking competition with the score of 0.002263 mAP.

The evaluation of the validation dataset returns the following values: To understand why these results are so low, we have to go through some considerations.

First of all the model composition, the modularity is one of the key aspects for all the main models into the competition. This because it lets developers decide the



```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 367.27                 Driver Version: 367.27                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K40c          Off  | 0000:06:00.0     Off |                    0 |
| 24%   46C    P0    62W / 235W |      0MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K40c          Off  | 0000:08:00.0     Off |                    0 |
| 41%   73C    P0   125W / 235W |  10983MiB / 11439MiB |     92%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla K40c          Off  | 0000:81:00.0     Off |                    0 |
| 23%   40C    P0    62W / 235W |      0MiB / 11439MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla K40c          Off  | 0000:82:00.0     Off |                    0 |
| 24%   46C    P0    62W / 235W |      0MiB / 11439MiB |     82%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    1     32364    C   python                                     10979MiB |
+-----------------------------------------------------------------------------+
```

Figure 5.2: GPU usage of the TensorBox component of the model.

Class Name	**mAP score**	**Class Name**	**mAP score**
airplane	0	**lion**	0
antelope	0	**lizard**	0
bear	0	**monkey**	0
bicycle	0	**motorcycle**	0.0219
bird	0	**rabbit**	0
bus	0	**red_panda**	0
car	0.0002	**sheep**	0.0329
cattle	0	**snake**	0
dog	0.0006	**squirrel**	0
domestic_cat	0.1492	**tiger**	0
elephant	0	**train**	0
fox	0	**turtle**	0.0615
giant_panda	0	**watercraft**	0.0001
hamster	0	**whale**	0
horse	0	**zebra**	0

Table 5.9: Per-Class validation routine results.

best configuration between modules to get the best results for the task.

So it's important to analyze each of them to understand which one is the main responsible. Starting with the training results I explained in the Development chapter, we can easily understand which between the two main modules (TensorBox, Inception) is the one that is lowering down the mAP.

Inception for its nature and implementation is the most solid, having high accuracy for Visual Recognition of objects in images. This is clear from the validation/test graph extrapolated from TensorBoard (see figure), so for sure this is not the main responsible of the low results.

Instead for TensorBox, if we look the considerations made into the Development chapter, it's clear that the peaks of accuracy into the test phase and the image test taken from the Control Board, explain that the low detection precision comes from this model.

So the modules after the General Detector suffer of its low accuracy of detection, from the tracker to the Inception module. The tracker might increase or decrease the accuracy, but the errors come from the first module and the latter is not learning how to compensate the errors but it only makes a post processing on the results given by TensorBox.

The final consideration could be to invert the order of some modules trying to achieve a better final result. I will go more in deep on this final consideration in the conclusive chapter.

Chapter 6

Conclusion

I divided the following chapter into overall considerations about the thesis and personal ones on the project itself.

6.1 Overall Conclusions

Resuming, the main aims of the thesis are: implementing a Deep Learning environment with TensorFlow technology and trying to implement a model suitable for the VID ILSVRC challenge.

The first one seems easy to realize, but face to the real implementation of the environment, solving problems and have it always up to date was seriously difficult; but at the end it is working efficiently and now anyone can work on it easily.

The second one was the hardest to achieve and at the end, the final results show that there are only the bases and the basic concepts for a real, well-working model for object detection and tracking in video. This consideration comes from the low mAP of the validation set, from the challenge results and mainly on the training results of the models.

The modules configuration could be for sure improved, switching the order of them into the following: Inception, TensorBox, Tracker.

Mainly the tracker must be a trainable component and not a post processing module. This for sure can improve the accuracy, not only for the tracking accuracy, but also to avoid false positive or missing detections.

In the switched architecture, the Inception component will detect the objects into the frames, whose accuracy it's for sure higher than the one of TensorBox and could be also improved by statistical considerations on the whole set of frames in the video. Then we have to train an Overfeat model for each class of the challenge. In this way, using the higher quality results of Inception model object recognition, we can use the class-specifics trained model to look for objects into frames.

Finally, the trainable tracker will make the in space and time analysis on the still image detection made by the previous modules.

Considering the really small time constraints, 5 months and my personal limited knowledge into the Deep Learning Topic at the start, I think I reached the aims of the thesis. The initial question was: "Can TensorFlow [21] be the fastest adaptable environment for Machine Learning, for implementing Research and Development, into a different infrastructure architecture with a great improvement perspective?

” I think the answer is clear: Yes, Tensorflow [21] is a great, adaptable and really well supported Machine Learning library, with an incredible improvement perspective, given not only by the author itself, Google [1], but, more important by a fast-growth Digital Community. Tensorflow [21] is perfect to implement a Research and Development sector in any Academics, Company or house; its adaptation properties are incredible, it can be installed and used from a Raspeberry Pi [] to a Supercomputer infrastructure, giving the possibilities to adapt the development to the research needs and to the available computational power. All these features are incredible and due to the Python environment, it is easy to use other third parts' libraries adapting the environment to the needs. I suggest everyone interested in approaching Machine Learning or implementing a Research and Development area in the topic to start with TesorFlow[21].

6.2 Personal Considerations

The thesis was a great personal challenge, anything easy to achieve. Initially I was pretty worried about a topic as Machine Learning, that for me was personally new, but more for the results expectations, because I was starting everything from scratch.

In 5 month, I started a Deep Learning environment into a Server Infrastructure, solved problems and errors regarded not only the TensorFlow [21] constraints itself but also for some useful third part libraries, like OpenCV. I Learned from the basics, to the more advanced knowledge in the Machine Learning general topic and Image Recognition specific one, adapted to my previous theory experience into it, with a young, powerful, but ”restrictive” Library as Tensorflow [21]. Despite to the fast-growth of its Community and available implemented projects, it gave me many problems. This because, from its early birth, not all the well-experienced Researcher and Users of the Image Recognition topic, started re-implement from scratch models and components yet available in Theano [50] or in the more complex Caffe[3] libraries.

This is why I said ”restrictive”, because it has limited my possibilities to work faster and had a better implementation of single modules of the model to achieve highers mAP results.

The time constraints led me the possibilities to try different projects as YOLO [16], TensorBox[44] or Inception[20] but not to implement complex models structure from scratch. Because to implement the working and error-prone environment took me almost one month, then learning the Deep Learning basics, understanding which were the bests starting points projects, took me another month and half. Finally when everything was ready to work the time was shorter and going from theory to implementation was a great step and for anything error-prone.

I had to solve: run problems, dataset problems, architecture development problems, all by myself in almost half of the total time constraints. The Machine Learning topic it's really complicated:it's necessary to have experience and a depth knowledge on the general and specific topic, so the team work it's the best choice, where is possible a parallel development and problem solver strategies.

I had to face problems one by one, spending time on the development of parts in which there where problems with the dataset or with tests and vice-versa. If I had a more stable project or a longer time constraints the possibilities were incredible; more over If I had experience in the specific and general topics, knowing best-practice for the development and for the dataset managing, etc. Some of the considerations I made in the Overall conclusions were discussed with the Advisors, but were labeled as ”last-chance”, that at the end it was impossible to implement in less than one month, so I decided to continue and try to boost accuracy as much as I could. The development's period, Summer, and the ”asterix” infrastructure, due to the high GPUs workload gave me a lot of problems, crashing many times, culminating in August with a 10 days off, due to the impossibility to restart it outside from the technical office. Summing the main factors as: TensorFlow, the ”short” time constraints, the development of a model for Object Detection in Video, my important lack of knowledge into the topic (general and specific), I've done the maximum possible facing also hard personal moments. For these reasons I'm glad about knowledge I've acquired, not only from this experience, but also for the whole road I've crossed to reach these results, which after the previous considerations are more than acceptable.

Appendix A

Appendix

This chapter resumes some basic knowledge I learned to reach my Thesis' purpose.
Initially, I will briefly expose the ILSVRC history and evolution that,from my point of view,it is really useful to understand how the challenge and the knowledge has evolved in time.
Then, I will give to the reader some basic notions in order to help him to understand the State of the Art and Development of the Work chapters, in which different structures and concepts are clearer.
The last concepts I will write about are the knowledge and steps that I had to get in TensorFlow library, showing the importance of its community and then some information about its general performance.

A.1 Basics History Information about ILSVRC

A.1.1 The competition

Figure A.1: ILSVRC competition Logo.

The ILSVRC was born in 2010 helded as a "taster competition" in conjunction with PASCAL Visual Object Classes Challenge 2010 (VOC2010). Created by Alex Berg, Fei-Fei Li and Jia Deng respectively from Columbia, Princeton and Stanford.
The first edition had only one challenge, focused on image categorization with 1000 object classes. Now at the 7th edition we can find 5 challenges:

- object localization;
- object detection;
- object detection in video;

- scene classification;
- scene parsing;

All with the same number of classes to identify.

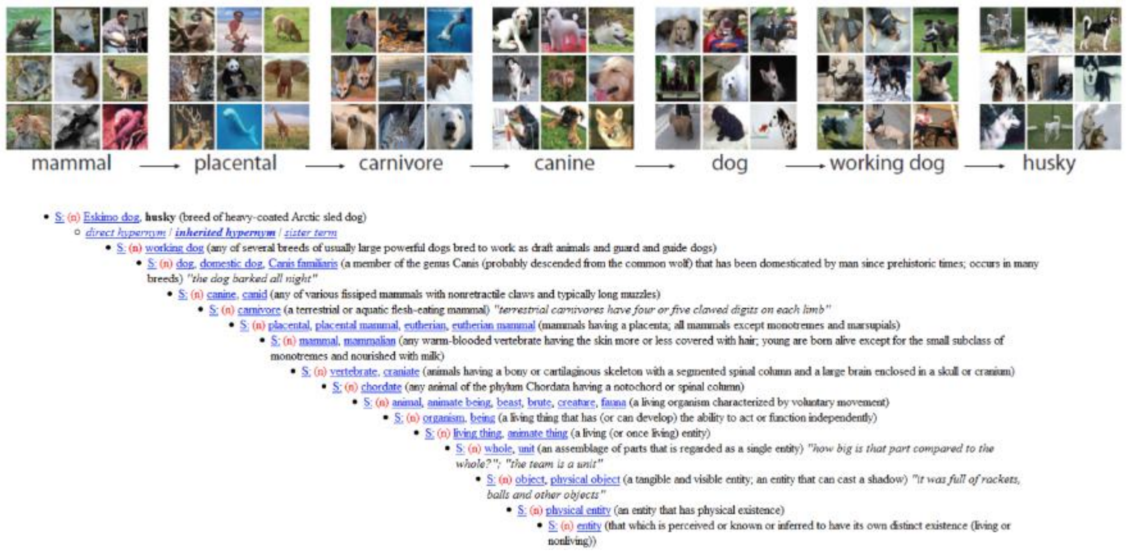
As evident, the increasing interest in the community, through the growing support by the University Authorities, led the challenge to become not more a tester challenge but always hosted in conjunction with PASCAL VOC. But it's important to highlight that the evolutions of the challenge, which started from the basic image classification as evolved, undertaking new challenges and eliminating the ones where the State of the Art limit was fixed, helped to improve the support and the focus on the topic.

In the Digital Era, the Visual Recognition area is an hot topic of interest not only for university researchers but more for service companies as Google [1], Facebook [7], Microsoft [35] and NVIDIA [36], that supporting this type of competitions, can achieve better results for their projects. This because device applications of the algorithms implemented into the competition are Virtual Reality, Augmented Reality, or as few weeks ago Google affirmed Transmogrified Reality [17].

A.1.2 The ImageNET Dataset

The challenge is based on the ImageNET Database, a collection that grows with the challenge in time: adding categories, images, video, snippets, annotations, and more important checking the correctness of the latter, otherwise the quality of the learning could be affected.

As their website describes: "ImageNET is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node" [43].


mammal → placental → carnivore → canine → dog → working dog → husky

- S: (n) Eskimo dog, **husky** (breed of heavy-coated Arctic sled dog)
  - *direct hypernym* | ***inherited hypernym*** | *sister term*
    - S: (n) working dog (any of several breeds of usually large powerful dogs bred to work as draft animals and guard and guide dogs)
      - S: (n) dog, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
        - S: (n) canine, canid (any of various fissiped mammals with nonretractile claws and typically long muzzles)
          - S: (n) carnivore (a terrestrial or aquatic flesh-eating mammal) *"terrestrial carnivores have four or five clawed digits on each limb"*
            - S: (n) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
              - S: (n) mammal, mammalian (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
                - S: (n) vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
                  - S: (n) chordate (any animal of the phylum Chordata having a notochord or spinal column)
                    - S: (n) animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)
                      - S: (n) organism, being (a living thing that has (or can develop) the ability to act or function independently)
                        - S: (n) living thing, animate thing (a living (or once living) entity)
                          - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
                            - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rockets, balls and other objects"*
                              - S: (n) physical entity (an entity that has physical existence)
                                - S: (n) entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Figure A.2: Hierarchical organization of Worldnet Database of the English language.

For who is wondering, Worldnet [52] is a large database of the English language, from syntactical to lexical, hosted by Princeton University. The use of this database for the labels and semantics was fundamental to correlate further complex computations without loosing previous results, so from the localization to the labeling, finally to the scene classification and parsing. This always has to follow the parallel human knowledge of the meanings on the images.

If someone asks which is the aim to invest all this effort to build a so detailed database of images, the answer is easy and direct: Helping the continuous interest on the topic to develop and increase the community. Everything is strongly related with the database because without it, the source of learning is none and so it becomes impossible the development and the research in new directions. As I said many times, Database is the source of the quality and the type of the learning, and both are not to underestimate.

A.2 Basics Knowledge of Deep Learning

I will give some definitions and basic laws, some of the principal history milestone, with their technical advantages and finally how now the environment is evolving.

A.2.1 Definition and Basics Laws

The complete name is Machine Deep Learning; it's a branch of the learning based on a set of algorithms, that attempts to model high level abstractions in data, using multiple processing layer. These models are based on two fundamental concepts, the neural network, see Figure A.3, and the back propagation algorithm.

The first is a system that tries to approximate the human brain structure with programs and data structure combination. The network initially is trained on a large amount of data and rules about their relationship and then tested on the learned function, like object detection. The structure of the network is usually composed by three fixed layer typology: Input Layer, Hidden Layer, Output Layer. The ones that really learn are the hidden ones, that can be 1+n, in any combination and functionality, where their total number defines the length of the model; from here the adjective deep.

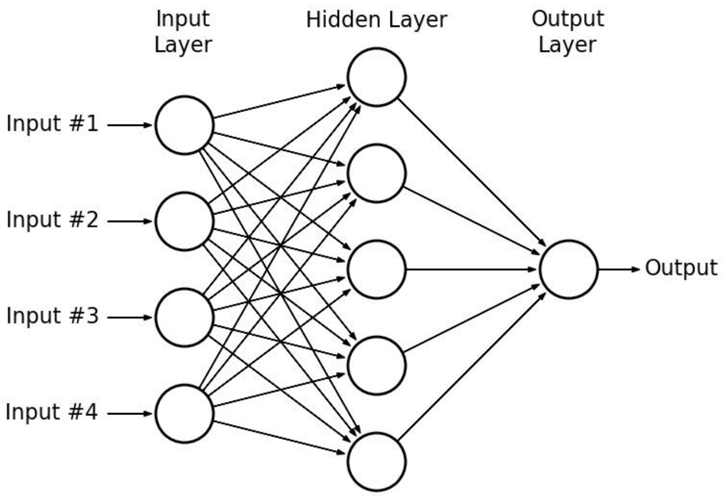


Figure A.3: Neural Network schema, on the edges are stored the weights of the model. This features values are the one updated through the Back Propagation algorithm.

Then it's the Back Propagation Algorithm that makes possible training the model respect to an optimization method. The system calculates the gradient of a loss function, respect to all the weights in the network and then it's passed to the optimization method that updates all the network weights trying to minimize the loss function.

This two laws are at the base of any other more complex concepts in Deep Learning area, in the next sub paragraph I will go through the history milestone of the concepts that are at the base of the nowadays models.

A.2.2 Principal History Milestone

The history of Machine Learning was born in written digit and numbers' areas: the first ones are digitally representable. In the 1965 Alexey Grigorevich Ivakhnenko, a Soviet and Ukrainian mathematician, builds the first working learning algorithm, the Group Method of Data Handling, becoming famous as the father of Deep learning. It's only 6 years later, in 1671, that the GMDH becomes the first 8 layers Neural Network, trained.

But the problem of train multiple hidden layers simultaneously remains open until 1898, when Yann LeCun [34] applies the back propagation algorithm to a Neural Network, from here forward also the hardware limitations was starting to end shortly.

The 90s are the Gold Age for the basics architectures, starting from 1993 with Jurügen Schmidhuber proposing the Recurrent Neural Network, a particular model where thanks to the directed cycle of connections between units, have internal memory, showing dynamic themporal behaviour.

Still Jurügen with Sepp Hochreiter , in 1997, propose the Long Short Term Memory model, an evolution of the RNN, from which differs for his well-suited structure to learn from experience to classify, process, predict also time series, that becomes important for long term run as the needed. Finally in 2000, Yann LeCun, proposes the Convolutional Neural Networks. Developed starting from the organization of animal visual cortex, the network is composed by two fundamental cells types: simple ones, that respond maximally to specific edge-like patterns inside their receptive field, and the complex ones, having larger receptive fields and locally invariant to the exact position of the pattern and acting as local filters over the input space.

From here forward the developed models comes as combination or evolution of those we have been through.

A.2.3 Nowadays Environment

Today, the most of the proposed architectures are basic models evolutions depending on the application area.

One particular example that I encountered during my fast learning, is the You Only Look Once model [37], that differently from the main strategies for visual recognition as the sliding window in the convolutional neural networks proposes a complete different and really light weight model.

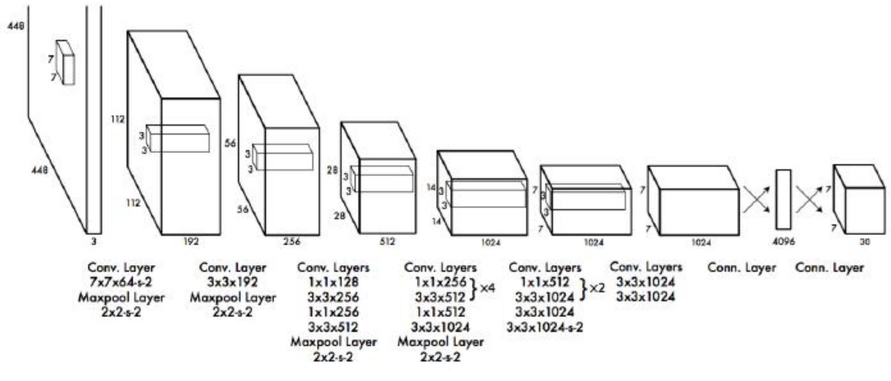

448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }x4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }x2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure A.4: Architecture schema of the YOLO model.

This architecture,as showed in Figure A.4, is composed by a single neural network on the whole image, that divides in fixed regions the picture and proposes different bounding boxes and related confidence for each one.

Strategies like YOLO are the raising ones that try to overcome the high computational workload, with architectures that, also on a portable computer, could be considered "real-time". Unfortunately it has really less support from the community, due to an high effort needed for the training (quality of the pictures) and lower detection rate.

I can affirm that is a matter of trade off between real applications and research topics. Here, some results form a run of YOLO on artwork:

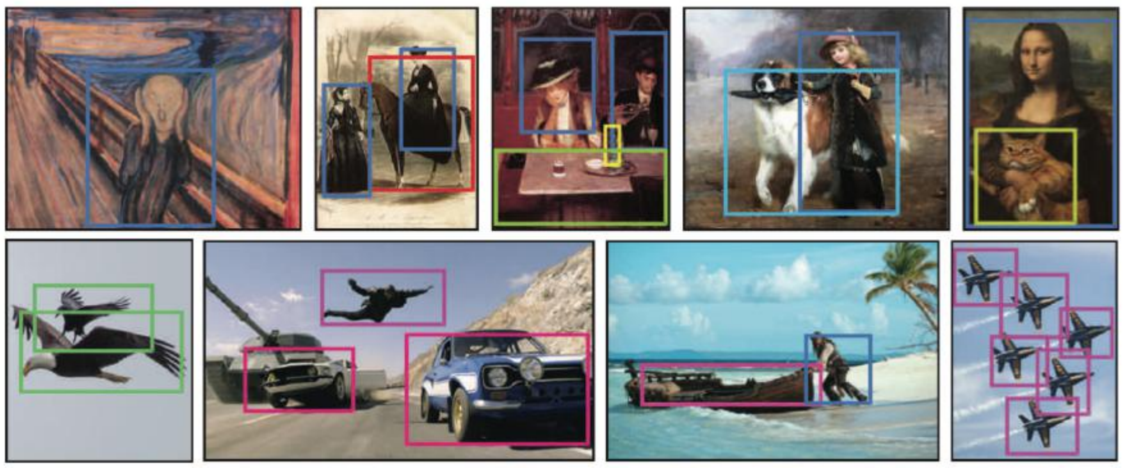


Figure A.5: Results on artwork dataset, from You Only Look Once model, looks pretty accurate if wasn't for some wrong classes detection.

A.3 Basics Knowledge about Tensorflow

This last section, from the computer architectures point of view, is one of the more technical. I will explain how I installed the environment problems I encountered and how I solved them. Then I will go through the community that makes so strong and fast in growth TensorFlow and finally I will make a brief computational evaluation of the library and some possible supported configurations.

A.3.1 The TensorFlow Environments

TensorFlow library is available for Python 2.7 or 3.3+, for all Linux-kernel based systems as Linux and OSX.

It comes with many installation support, but most important methods, we can install it in five different ways: Pip, Anaconda, Docker, Virtual Environment and from source.
The plenty number of possible isolated installation, made easy to choose the more comfortable one for the infrastructure and use it. As asterix as yet installed VirtualEnv and its great usability and small abouse of systema space, I've use the following instruction to install it:

```
mkdir VirtualEnv
cd VirtualEnv
mkvirtualenv tensoflow
cd tensoflow
pip install --source-link-tensorflow
```

From here the basic TensorFlow library was operating, but because of the infrastructure possibilities, 4 GPUs Nvidia Tesla 40c, was required to boost and use all the available hardware to speed up performance. With the help of the technical support for the really delicate installation, we found after a while the right version of the CUDA and cuDNN libraries of Nvidia to maximize the GPU computational performances. Problems came out due to the shared environment, where some versions were compatible each other and others had unsolvable installation problems, that obliged me to skip it. To use this performance computational libraries with the Virtual Envirornment, I had only to export the needed paths:

```
export LD_LIBRARY_PATH="\$LD_LIBRARY_PATH:/usr/local
    /cuda-7.5/lib64:/home/users/aferri/cuda"
export CUDA_HOME="/usr/local/cuda-7.5"
```

Here ends the principal environment configuration for TensorFlow, but it was everything than easy to find the right combination between the library and the ones of Nvidia.

Then, when I started developing in the Visual Recognition topic, I immediately encountered some other dependencies problems with the Open Source Computer Vision library.
This library is one of the most used for the Visual Recognition, with a lot of useful functions yep implemented, but it's pretty tricky to install in a server environment and then use it in the Virtual one. As done with the Nvidia libraries, after installing on the server, we have referenced the paths inside the Environment of work:

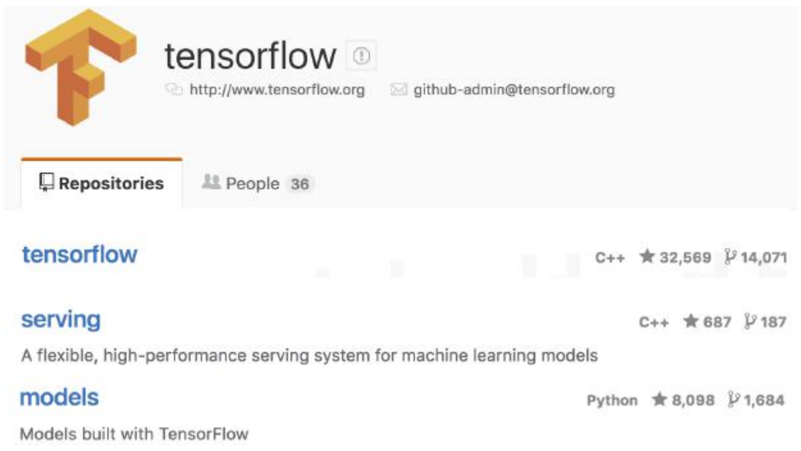
```
export PYTHONPATH="\${PYTHONPATH}:/opt/amd64/opencv
    -3.1.0/lib/python2.7/dist-packages"
export LD_LIBRARY_PATH="\$LD_LIBRARY_PATH:/opt/amd64
    /opencv-3.1.0/lib"
```

Here the main difficulties ended, later on, I only used some other useful python libraries, easily installable through "pip install" as: PIL the python image library, sometimes preferred to OpenCV, progressbar, to highlight the remaining time for the computation and the last but not the least NumPy library part of the SciPy package. This library is the basic of the numerical computation in python, defining types and functions useful for the numerical manipulation of images.

A.3.2 The community

The TensorFlow community's help was really important to achieve the thesis goals and personally, for its really early birth, I think it's one of the best created and well-grown in the Digital Era so far.

Starting from the official website [21], where all the basics information to download & setup and basic usage are available, then there are example tutorials that cover the basic programming structure for almost all the basic models for visual recognition, here I found the one for Inception [18].



tensorflow
http://www.tensorflow.org github-admin@tensorflow.org
Repositories People 36

tensorflow C++ ★ 32,569 14,071

serving C++ ★ 687 187
A flexible, high-performance serving system for machine learning models

models Python ★ 8,098 1,684
Models built with TensorFlow

Figure A.6: Tensorflow GitHub community page.

Another great move from the Google side was to find and activate immediately with the release of the library a GitHub project for TensorFlow [22] that in less than one year has more than 13k fork, 32k stars and more than 3k of watching users. The project page is not only for the mentioned numbers a great source of users, but a great one for code, users custom library, users project, that are daily committed to the project and that increase significantly the potentiality of the library itself. Between the most relevant users contributions there are some of them like the TensorFlow Slim, a framework that gives easier APIs for yet implemented layers, so the user can build with less effort more complex networks from scratch. The success of this commit was so high that also Google starts to simplify some of the interfaces to have a more intuitive usage.

A.3.3 Computational Evaluation

TensorFlow was born for a distributed environment and Google hasn't fear to show it. The library has many tools to use to understand the dynamic state of the model in real time during the training only, but more over gives a banch of APIs to output what it's our interest on this tool. The most useful is TensorBoard, see the Figure A.7, that executed in a different process in the output folder of the running model, shows us what we want:

- Any kind of Event Graph, from loss functions to activations and weights;
- Any kind of Image, from the training to the test one;
- Any kind of Audio, like the previous point;
- The graph model, where is possible analyze the depth and balance of the architecture;
- Any kind of Histogram function.

All the models structures and graphs I reported in the thesis are extrapolated from this amazing Tool. The power of this control board, that is visualizable on the

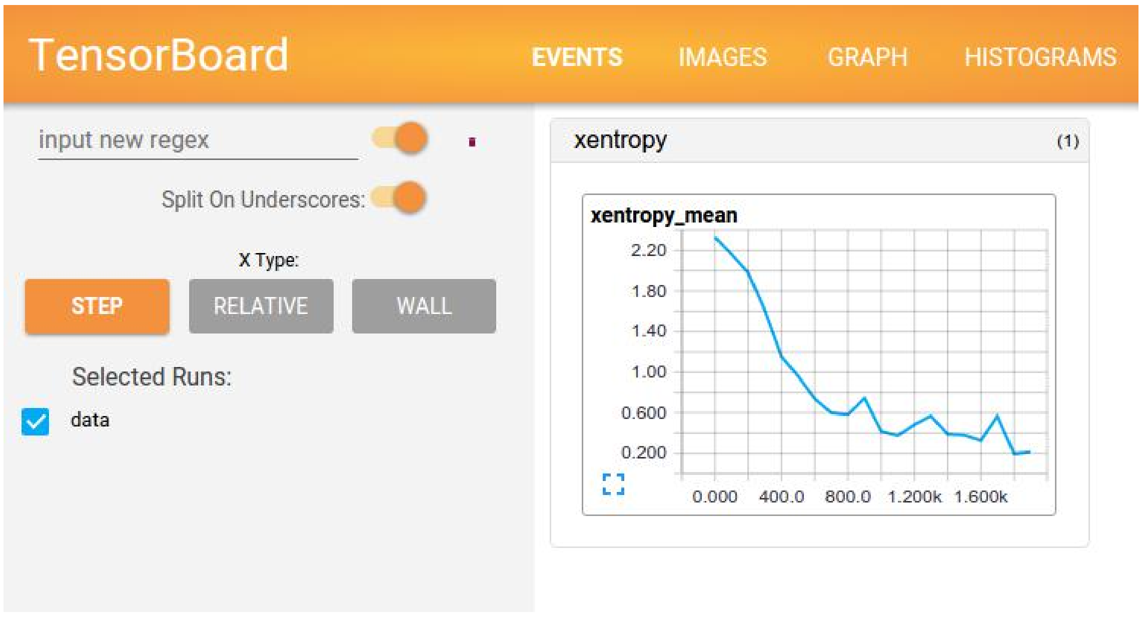


Figure A.7: Example of Tensorboard control pannel.

browser is unlimited, giving a great and better prospective to the developer and researcher during the model development . This tool works not only with a single machine but also with a distributed run, for these reason Google gives a great amount of examples, tutorials and how to, for the distributed use of TensorFlow. We can find direct links to the official website [19], but also for each developed model present on GitHub [20], there are plenty instruction on how to implement the distributed version of it. The aim of this library is to run in environments with multiple CPUs and GPUs giving the user the possibility to choose in numbers and workload, having always everything under control with the help of the control board.

Bibliography

[1] Alphabet. *Google*. 2016. URL: https://abc.xyz/.

[2] University of Amsterdam. *University of Amsterdam*. 2016. URL: http://www.uva.nl/en/home.

[3] Berkley. *Caffe, Deep Learning Framework*. 2008. URL: http://caffe.berkeleyvision.org/.

[4] BSC-UPC. *BSC-UPC, NVIDIA GPU Center of Excellence (GcoE)*. 2015. URL: https://www.bsc.es/marenostrum-support-services/hpc-education-and-training/cuda-center-excellence.

[5] Barcelona Supercomputing Center. *BSC*. 2016. URL: https://www.bsc.es/.

[6] Google Developers. *TensorFlow for Poets*. 2015. URL: https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/index.html#0.

[7] Facebook. *Facebook Website*. 2016. URL: https://www.facebook.com/about/login/.

[8] Andrea Ferri. *Contribution 1 Link (a)*. 2016. URL: https://github.com/Russell91/TensorBox/pull/39#issuecomment-233150694.

[9] Andrea Ferri. *Contribution 1 Link (b)*. 2016. URL: https://github.com/Russell91/TensorBox/commit/aab8e42c12d20c5e9acaa933535ea1696d66c608.

[10] Andrea Ferri. *Contribution 2 Link*. 2016. URL: https://github.com/tensorflow/models/pull/200#issuecomment-225902722.

[11] Andrea Ferri. *Contribution 3 Link*. 2016. URL: https://github.com/Russell91/TensorBox/pull/46#issuecomment-234479801.

[12] Andrea Ferri. *Contribution 4 Link*. 2016. URL: https://github.com/Russell91/TensorBox/pull/49#event-733979605.

[13] Andrea Ferri. *$Tensorflow_Object_Tracking_Video$*. 2016. URL: https://github.com/DrewNF/Tensorflow_Object_Tracking_Video.

[14] Ross B. Girshick. “Fast R-CNN”. In: *CoRR* abs/1504.08083 (2015). URL: http://arxiv.org/abs/1504.08083.

[15] Ross B. Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *CoRR* abs/1311.2524 (2013). URL: http://arxiv.org/abs/1311.2524.

[16] gliese581gg. *YOLO Tensorflow*. 2016. URL: https://github.com/gliese581gg/YOLO_tensorflow.

[17] Google. *GDC Transmogrified Reality Talk*. 2015. URL: http://www.gdcvault.com/play/1022343/Transmogrified.

[18] Google. *Image Recognition in TensorFlow*. 2015. URL: https://www.tensorflow.org/versions/r0.10/tutorials/image_recognition/index.html.

[19] Google. *Imagenet Retraining Last layer for new task*. 2015. URL: https://www.tensorflow.org/versions/r0.9/how_tos/image_retraining/index.html.

[20] Google. *Inception Tensorflow*. 2016. URL: https://github.com/tensorflow/models/tree/master/inception.

[21] Google. *TensorFlow*. 2015. URL: https://www.tensorflow.org/.

[22] Google. *Tensorflow*. 2016. URL: https://github.com/tensorflow.

[23] Image Processing Group. *IPG*. 2016. URL: https://imatge.upc.edu/web/.

[24] W. Han et al. "Seq-NMS for Video Object Detection". In: *ArXiv e-prints* (Feb. 2016). arXiv: 1602.08465 [cs.CV].

[25] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). URL: http://arxiv.org/abs/1512.03385.

[26] Kaiming He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *CoRR* abs/1502.01852 (2015). URL: http://arxiv.org/abs/1502.01852.

[27] ILSVRC. *ILSVRC 2011 Results*. 2011. URL: http://image-net.org/challenges/LSVRC/2011/index.

[28] ILSVRC. *ILSVRC 2012 Results*. 2012. URL: http://image-net.org/challenges/LSVRC/2012/results.html#t2.

[29] ILSVRC. *ILSVRC 2016 Results*. 2016. URL: http://image-net.org/challenges/LSVRC/2016/results.

[30] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167.

[31] Dinesh Jayaraman and Kristen Grauman. "Slow and Steady Feature Analysis: Higher Order Temporal Coherence in Video". In: *CoRR* abs/1506.04714 (2015). URL: http://arxiv.org/abs/1506.04714.

[32] K. Kang et al. "T-CNN: Tubelets with Convolutional Neural Networks for Object Detection from Videos". In: *ArXiv e-prints* (Apr. 2016). arXiv: 1604.02532 [cs.CV].

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: (2012). Ed. by F. Pereira et al., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[34] Yann LeCun. *Yann LeCun Website*. 2015. URL: http://yann.lecun.com/.

[35] Microsoft. *Microsoft Website*. 2016. URL: https://www.microsoft.com.

[36] Nvidia. *NVidia Website*. 2016. URL: http://www.nvidia.it/page/home.html.

[37] Joseph Chet Redmon. *YOLO: You Only Look Once*. 2015. URL: http://pjreddie.com/darknet/yolo/.

[38] J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *ArXiv e-prints* (June 2015). arXiv: 1506.02640 [cs.CV].

[39] S. Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *ArXiv e-prints* (June 2015). arXiv: 1506.01497 [cs.CV].

[40] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[41] Pierre Sermanet et al. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *CoRR* abs/1312.6229 (2013). URL: http://arxiv.org/abs/1312.6229.

[42] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). URL: http://arxiv.org/abs/1409.1556.

[43] Princeton University Stanford University. *ImageNet Large Scale Visual Recognition Competition*. 2010. URL: http://www.image-net.org/challenges/LSVRC/.

[44] Russel Stewart. *TensorBox*. 2016. URL: https://github.com/Russell91/TensorBox.

[45] Russell Stewart. *Russell Stewart website*. 2014. URL: http://russellsstewart.com/.

[46] Russell Stewart and Mykhaylo Andriluka. "End-to-end people detection in crowded scenes". In: *CoRR* abs/1506.04878 (2015). URL: http://arxiv.org/abs/1506.04878.

[47] Christian Szegedy et al. "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842 (2014). URL: http://arxiv.org/abs/1409.4842.

[48] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). URL: http://arxiv.org/abs/1512.00567.

[49] C. Szegedy et al. "Going Deeper with Convolutions". In: *ArXiv e-prints* (Sept. 2014). arXiv: 1409.4842 [cs.CV].

[50] Theano. *Theano*. 2008. URL: http://deeplearning.net/software/theano/.

[51] Università di Trento. *Università degli Studi di Trento*. 2016. URL: http://www.unitn.it/.

[52] Princeton University. *Worldnet, a lexical database for English*. 2010. URL: http://wordnet.princeton.edu/.

[53] The Verge. *Google revelead AlphaGo secret Hardware with ASIC chip-tensor Processor Unit for Machine-Learning*. 2016. URL: http://www.theverge.com/circuitbreaker/2016/5/19/11716818/google-alphago-hardware-asic-chip-tensor-processor-unit-machine-learning.

[54] PASCAL VOC. *PASCAL VOC 2010 Leadboard*. 2015. URL: http://host.robots.ox.ac.uk:8080/leaderboard/displaylb_dt.php?challengeid=6&compid=4.

[55] PASCAL VOC. *PASCAL VOC 2012 Leadboard*. 2015. URL: http://host.robots.ox.ac.uk:8080/leaderboard/displaylb_dt.php?challengeid=11&compid=4.

[56] PASCAL VOC. *PASCAL VOC Website*. 2015. URL: http://host.robots.ox.ac.uk/pascal/VOC/.