DLVSP Report Assignment 2

Santiago Rattenbach Paliza Bartolomé and Carlos Martínez García

I. INTRODUCTION

This report presents an experimental study on Video Object Detection using the official implementation of the MEGA model. The primary objectives are to become familiar with the use of Git repositories and learning to deal with third-party libraries and its code. The main difficulty of the task lies in the fact that the project on which we will work has been abandoned for the last five years, which introduces many incompatibility problems with the libraries used. Because of that, we will have to adapt certain aspects of the project in order to make it work.

After setting up the environment, we will ensure it is working by running the demo files, then, we will test the implementation using two different checkpoints of the model, BASE and MEGA. Finally, we will analyze the results of each checkpoint and compare them, so we can identify the improvements or errors that appear when using the MEGA approach.

II. METHOD

First, we create a repository from scratch and clone said repository into a local folder. We then download the contents of the original "mega.pytorch" repository and follow the installation instructions in the INSTALL file. The original installation instructions are outdated, so we had to modify some of the command instructions featured in the INSTALL file, we have decided to create an updated version of the said file so the setup could be easily replicable.

When running the setup command instructions, we get some errors related to the "Apex" library. We realized that the project could work without making use of said library, so we modified some lines of code in order to make the setup work without creating problems when running the model. The modified files can be accessed from our GitHub repository[1].

After finishing the setup, we run the demo to check everything works properly, then we run the model on the different videos stated in the instructions (three videos from the UCF-101 dataset and one of our choice), once with the BASE checkpoint and once with the MEGA checkpoint, then we compare the results.

The difference between the baseline version of the model and the MEGA approach lies mainly in the reliance of local aggregation (using only a few adjacent frames) versus global aggregation (using sparse sampling) for information processing. The baseline model relies on either of the two approaches, while the MEGA approach introduces the concept of global-local aggregation, combining both types of aggregations while introducing a long range memory module to overcome limitations related to the lack of information on feature aggregation.

[1]Github repository

III. IMPLEMENTATION

We followed the setup instructions in the project's README file for running the demo, applying the changes specified in the moodle instructions in order to make it work within the environment present in the lab's computers (which used Python 3.7). When executing the command lines, we stopped whenever we encountered any errors (not warnings), specifically in the installation of the "Apex" library, as its setup file had a syntax error: one of the lines was incorrect in Python versions prior to 3.10, so we modified it to make it work. After fixing it, we ran the setup again, and, while there was an error, it did not prevent the installation from finishing.

Once the libraries were installed, we proceeded to run the demo and encountered our first exception, which was expected. The first problem we had was actually related to the version of the Torchvision library used, as the original README file pretended to install the latest version instead of one suited for the version of PyTorch used (1.2.0), so we had to reinstall Torchvision with the correct version (0.4.0), but we made sure to update it in the installation script to avoid this issue in further implementations of the project.

Once we solved that problem, we tried running again and started getting exceptions related to the Apex library. We then proceeded to remove any instance of the Apex library that was used in the files required to run the model, as we realized that it was not necessary. The specific changes have been logged in our updated version of the README file, but in summary, we found the lines that needed to be modified by trying to run the demo and reading the error messages. The main change is present in the nms.py file, where we just removed the use of the amp module from the Apex library, we then removed imports of said module in other files. Apart from that, we only had to remove some function descriptors related to the amp module as they could no longer be used, but it did not matter as their removal did not affect the execution.

After having removed every trace of the amp module from the files used in the demo, we were able to run it without any issues and see the results.

IV. DATA

For running the demo in the "inference on image folder" mode, we used the example folder given in the subject's moodle.

For the experiments using both the BASE and MEGA checkpoints, we used three videos from the UCF-101 dataset (specifically "v_WalkingWithDog_g10_c03.avi", "v_WalkingWithDog_g01_c01.avi" and "v_HorseRiding_g10_c01.avi") as well as a fourth one taken from the internet that we could choose freely. We opted to use a video of a flying car, as "car" is one of the classes recognized by the model. We chose this video mainly because we thought it would be funny, but also because we thought it was interesting to see the output of the model when given a scene of a car with an unusual background, we predict that it may be mistaken for an aircraft.

Fig. 1: Frame of the video of the flying car.

V. RESULTS AND ANALYSIS

We will show and compare the results of the two tested models in all the different chosen videos.

A. Dog

When using the BASE model, the dog is mistaken multiple times with different incorrect classes, especially in the first few frames. At the start of the video, the man and the dog are relatively far away from the camera, and getting progressively closer to it; this aspect mixed with the low resolution of the video makes the first detections quite messy. The dog is also being occluded by its owner in the beginning, so the model gets confused and sometimes detects a bear and a bicycle sometimes it thinks that both the man and the dog make up a dog, sometimes it thinks they both make a bear... It takes a few more frames for it to completely realize that the dog and the man are two separate objects and starts more correctly labeling the dog most of the time. Due to the color of the dog, it sometimes blends with the background, so the model does not always detect it. Also, the model sometimes confuses the dog with a monkey, but looking at the image it becomes clear that it is not such an outlandish conclusion, as it does bear some resemblance with a certain type of monkey.

When using the MEGA model, the dog is always correctly labeled and almost always correctly detected. It is never mistaken for another class, but it sometimes is not detected, and some other times the man is detected as the dog instead of the actual dog, while some other times both are detected as dogs, but it is a rare occurrence and this type of anomalies do not last for more than a few frames.

B. Dog 2

Both the BASE and MEGA model do a terrible job detecting the dog in the image, although this video is a tricky one. The cat that appears at the beginning is mistaken for a dog during the whole video, while the actual dog is not detected most of the time, except for a few frames near the end, when the cat is off focus (even then, the model quickly stops detecting the dog, probably because its fur blends with the background). There is a large white car in the background which is not properly detected by either of the models, as they only bound a partial region of it. The main difference between both models is that, for a few frames, a bench in the background gets mistaken for a bus in the BASE model.

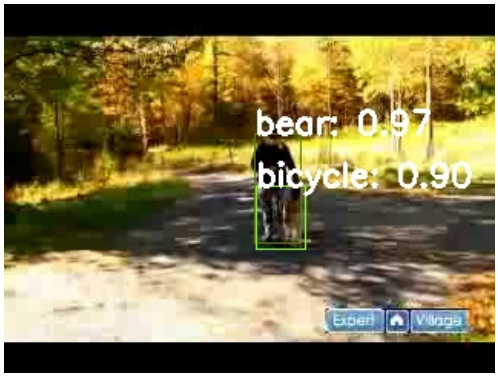


Fig. 2: BASE Frame 2, man and dog mistaken for bear and bicycle.

Fig. 3: BASE Frame 80, dog correctly detected and labeled

Fig. 4: BASE Frame 202, dog not detected probably due to background blend.

Fig. 5: BASE Frame 231, dog incorrectly labeled as monkey.

Fig. 6: BASE Frame 85, cat mistaken for dog, dog undetected, car partially detected and bench mistaken for bus.

C. Horse

In the horse riding video, we can see that both models are guessing correctly the target and labels, the accuracies are really similar, the confidence might change in some specific frames, but in general they are evenly-matched so there is not much to say.

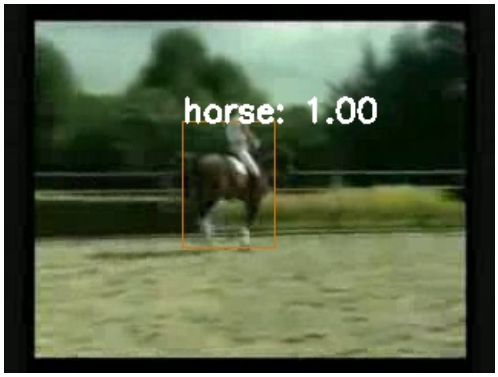
D. Car

Checking the second frame, we can see that the BASE model is detecting the flying car as an airplane, meanwhile, the MEGA model is detecting it as a car. This is just a coincidence as it just happens for a few frames, for the rest of the video, the BASE model may not detect the car at times or it will correctly identify the object as a car and not an airplane.

In some later frames, such as frame 65, we can see that both models are confused by the car and some background walls, thinking that there is a bigger car in the background partially occluded by a smaller one. Some smaller background cars are well detected in both examples.

In frame 103, the largest bounding box (the one that mistakenly mixed the flying car with the background) ceases to be displayed.

In this test, the mega model outperforms the baseline model, being more accurate in most of the predictions and choosing the correct labels in all frames, even though both of them had an hallucination.



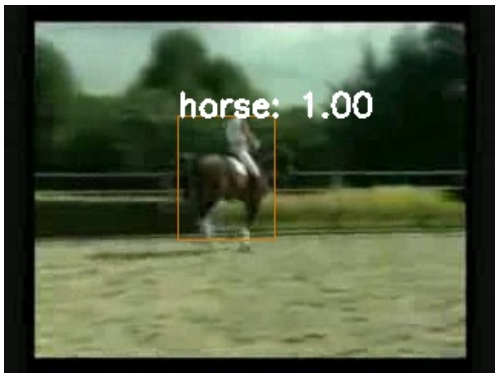


Fig. 7: Frame 53 comparison(confidence 1.0)

Fig. 8: Frame 144 comparison (discrepancy in confidence)

VI. CONCLUSIONS

In this project, we successfully managed to make use of an abandoned repository by resolving version mismatches and

Fig. 9: Second frame comparison

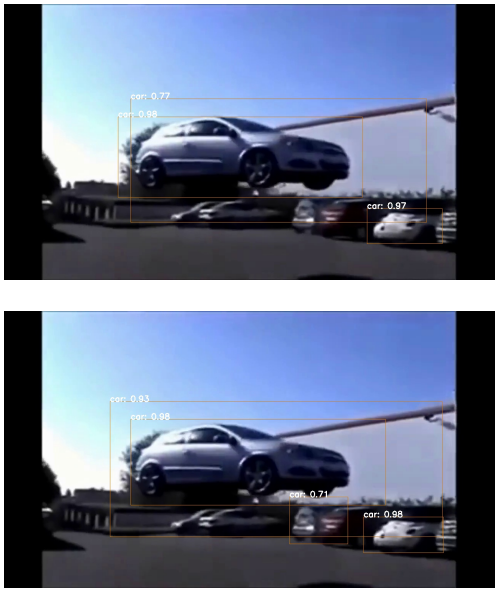


Fig. 10: 65th frame comparison

removing deprecated dependencies (specifically the Apex library) by tinkering with the code. This process of adapting an unknown technology stack provided valuable insight into the life-cycle of deep learning software and the importance of maintainable code, and it gave us insight into the surprising lack of compatibility present in relatively recent projects related to such a contemporary subject.

Experimental results show that the MEGA model offers substantial improvements over the BASE checkpoint. MEGA demonstrated superior accuracy in label prediction and bounding box stability, particularly in complex scenes with occlusions. While both models struggled with the unusual context of the ”flying car” video and the second video of the dog, MEGA consistently provided higher confidence scores

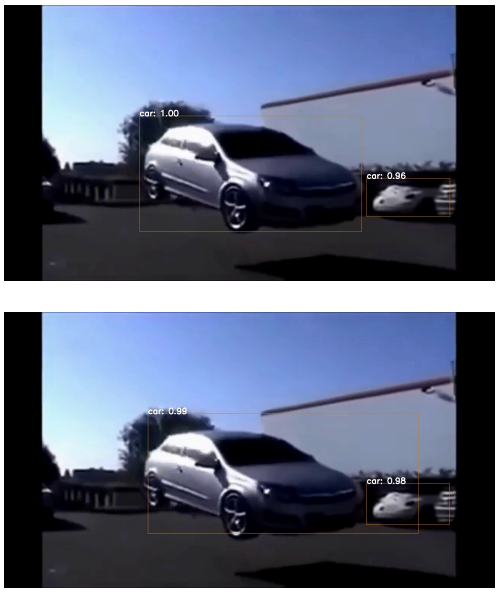


Fig. 11: 103rd frame comparison

and fewer classification errors across the standard UCF-101 dataset samples when compared to the baseline approach, although in some cases the distinction was minimal. That being said, the most important factor still relies on the data fed to the model, as both models had a hard time labeling and detecting objects in the harder samples, even though the baseline approach had a few more inconsistencies.

VII. TIME LOG

- Git setup: 3h Santiago, 1h Carlos
- Coding, testing and troubleshooting: 6h Santiago, 8h Carlos
- Writing the report: 3h Santiago, 3h Carlos