# Lab2 ”Video Object Detection”

Sven Leschber and Jade Martin-Lise

## I. INTRODUCTION

This laboratory report examines video object detection classification through two distinct sessions using the BASE and MEGA approaches from the CVPR-2020 paper “Memory Enhanced Global-Local Aggregation for Video Object Detection” [1]. The code implementing these approaches is available on GitHub [2] and it will be used in both sessions. In the first session, the setup and the required settings are installed in order to run the code from the GitHub repository [2]. Slight modifications are made on the 'apex' library, the detailed instructions to follow to run the code are given in the Section 'Method, Task 1.2'. Then, the demo code for both MEGA and BASE approaches is run in ”Inference on image folder” mode. In the second session, the code using MEGA and BASE approaches is run on videos and the result obtained with both approaches are compared. The aim being to identify the differences between the image based object detection and the tracks post-processing methods.

## II. METHOD

This assignment implements the BASE and MEGA approaches as given by the GitHub repository [2]. In order to run this code, the original instructions of the INSTALL.md must be followed as well as the additional instructions given below.

### A. Task 1.2

First step, follow the instructions given in the INSTALL.md file in the GitHub repository [2]. Then, slight modifications are made to these instructions to ensure that the code runs correctly.

• Install PyTorch version 1.2 instead of PyTorch 1.3. Furthermore, ensure that the correct corresponding of Torchvision is installed. The CudaToolkit must also be compatible with your environment. When using Linux environment and CUDA 10.0, the command is as follows : `conda install pytorch==1.2.0 torchvision==0.4.0 cudatoolkit=10.0 -c pytorch`. For any different configurations, PyTorch installation details are given on https://pytorch.org/get-started/previous-versions/.

• For the installation of Apex library, replace the line `'python setup.py install --cuda_ext --cpp_ext'` with `'python setup.py build_ext install'` to avoid CUDA errors.

• Errors related to the Apex Library require to slightly modify 7 python files from the library. Each python script listed below should be replaced by its modified equivalent which can be found in the GitHub repository [3]. It is also necessary to perform a git checkout command when cloning the GitHub since a recent change in the Apex GitHub causes an error when running the demo code. These modifications are detailed below.

- *Git Checkout Command :* A modification has been made in the GitHub of the Apex Library on the 28th of November. This modification causes an error when running the code, running a git checkout command will checkout this modification so the error is resolved. The git checkout command line to run after having cloned the Apex GitHub is : `git checkout a1df804`

- *grad_scaler.py :* The original code relies on `torch.cuda.amp.GradScaler` which causes an `AttributeError` due to its unavailability in PyTorch 1.2.0. To solve this, a GradScaler class is implemented using Apex AMP's scale_loss function for loss scaling. The original python script *grad_scaler.py* located in the `apex/transformer/amp` directory should be replaced by the python script named *grad_scaler.py* given in the GitHub repository [3]. By doing this, the modified *grad_scaler.py* script is used and the corresponding error is solved.

- *utils.py :* An `AttributeError` occurs because torch.distributed._all_gather_base does not exist in PyTorch 1.2.0. To fix this, torch.distributed.all_gather_into_tensor is replaced with a manual implementation using torch.distributed.all_gather. The original python script *utils.py* located in the `apex/transformer` directory should be replaced by the *utils.py* given in the GitHub repository [3]. The modified python script is implemented and will solve the error.

- *mappings.py :* The torch.distributed._all_gather_base is unavailable in PyTorch 1.2.0, the script attempts to map this method to torch.distributed.all_gather_into_tensor which causes an `AttributeError`. To resolve this, a custom implementation of all_gather_into_tensor is created. The original python script *mappings.py* located in the `apex/transformer/tensor\_parallel` directory should be replaced with the *mappings.py* given in the GitHub repository [3].

- *common.py :* The `torch.cuda.amp.GradScaler` is referenced in the code which causes an `AttributeError` due to its unavailability in PyTorch 1.2.0. To fix the error, a try-except block dynamically handles the import of torch.cuda.amp.GradScaler. If it is unavailable, it assigns GradScaler = None. Moreover, a dynamic-type notation is introduced which is set to either torch.cuda.amp.GradScaler or None ensuring type compatibility. The original *common.py* located in the `apex/`

transformer/pipeline_parallel/schedules should be replaced by the *common.py* given in the GitHub repository [3]. By doing this, the modified *common.py* file is used and avoids errors.

- *fwd_bwd_no_pipelining.py :* Attempting to access torch.cuda.amp.GradScaler results in an AttributeError because the AMP functionality is not available in PyTorch 1.2.0. To solve this error, the same solution as earlier is implemented. This consists in adding a try-except block to check for the existence of torch.cuda.amp.GradScaler. If unavailable, assigned GradScalerType = None to ensure compatibility. Moreover, the function parameter torch.cuda.amp.GradScaler is replaced with the dynamic-type notation GradScalerType to adjust based on availability. The original *fwd_bwd_no_pipelining.py* located in the apex/transformer/pipeline_parallel/schedules should be replaced by the *fwd_bwd_no_pipelining.py* given in the GitHub repository [3]. The modified *fwd_bwd_no_pipelining.py* file is used which avoids errors.

- *fwd_bwd_pipelining_with_interleaving.py* : The AttributeError occurs for the same reasons as in the previous python script. The same modifications have been applied to this python script in order to solve the error. The original *fwd_bwd_pipelining_with_interleaving.py* located in the apex/transformer/pipeline_parallel/schedules directory should be replaced by the *fwd_bwd_pipelining_with_interleaving.py* given in the GitHub repository [3]. By doing so, the corrected version of the *fwd_bwd_pipelining_with_interleaving.py* file is used which avoids errors.

- *fwd_bwd_pipelining_without_interleaving.py* : The AttributeError occurs for the same reasons as in the two previous python scripts. To solve the problem, the same modifications are included in this python script; the try-expect block is added as well as the dynamic type notation GradScalerType. The original *fwd_bwd_pipelining_without_interleaving.py* located in the apex/transformer/pipeline_parallel/schedules directory should be replaced by the *fwd_bwd_pipelining_without_interleaving.py* given in the GitHub repository [3] to avoir errors.

Finally, download the model checkpoints ('singleframe baseline' and 'MEGA' with backbone ResNet-101) from the root folder of the GitHub [2] and place these checkpoints in the the 'mega.pytorch' folder.

## III. IMPLEMENTATION

The implementation of the code consists in running the demo code, in both the MEGA and the BASE approaches. A slight modification is necessary in the *predictor.py* script located in the mega.pytorch/demo directory. In line 611, in overlay_class_names, modify the code to ensure (x, y) is cast to integers before passing to cv2.putText. Update the line as follows:

```
cv2.putText(image, s, (int(x), int(y)),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255,
255), 2)
```

### A. Task 1.2

The demo code is run in the "Inference on image folder" mode using the given images. It is run with both the BASE and the MEGA approaches, the model checkpoints are changed accordingly. The output is a set of images featuring the bounding boxes and their labels which can be found in the visualization folder. The following command is used to run the demo code using BASE approach :

```
pythondemo/demo.pybaseconfigs/vid\_R\
_101\_C4\_1x.yamlR\_101.pth--suffix".
JPEG"\textbackslash\\--visualize-path/
image\_folder\textbackslash\
\--output-foldervisualization
```

### B. Task 2.1

The demo code is run in the "Inference on video" mode using four different videos. It is run with both the BASE and the MEGA approaches, the model checkpoints are changes accordingly. Each output is a video featuring the bounding boxes and their labels. For example, the following command is used to run the demo code on the video "v_HorseRiding_g10_c01.avi" using BASE approach :

```
pythondemo/demo.pybaseconfigs/
vid\_R\_101\_C4\_1x.yamlR\
_101.pth--video\textbackslash\
\--visualize-path/v\_HorseRiding\
_g10\_c01.avi\textbackslash\
\--output-foldervisualization--output-video
```

## IV. DATA AND CHALLENGES

### A. Task 2.1

For the inference on video mode, the data used are three videos taken from the UFC-101 dataset as well as a video from the internet.

The **"v_HorseRiding_g10_c01.avi"** video from UCF-101 dataset does not present any challenges as there is only one object to detect in the image and no scale change, occlusion nor variation in appearance of the object.

The **"v_WalkingWithDog_g01_c01.avi"** video from UCF-101 dataset presents challenges. The dog and human being featured in the video walk towards the camera which means there is a change in scale of the object. Then, the dog sometimes is occluded by the human.

The **"v_WalkingWithDog_g10_c03.avi"** video from UCF-101 presents certain challenges as well. The first challenge is that there are multiple objects to detect in the video (car, cat and dog). There are also objects which do not belong to any classes such as a human being, a bench and some vegetation in the background. The second challenge is that multiple objects are moving and the camera has slight movement as well. The third challenge is that the objects are close to each other in some moments in the video.

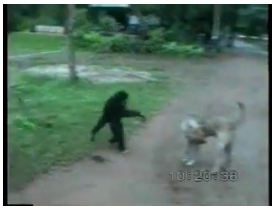
The video taken from the internet features a monkey playing

with a dog. The challenges in the video taken from internet were the low resolution of the video chosen as well as the fast movement of the monkey.

(a) Horse Riding

(b) Walking With Dog 01

(c) Walking With Dog 01



(d) Video from internet, Monkey and Dog

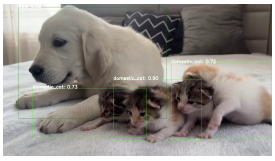
Fig. 1: Samples frames from the data

## V. RESULTS AND ANALYSIS

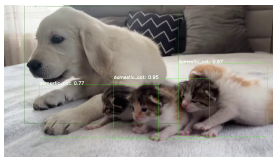
### A. Task 1.2

The aim of this section is to present briefly the results obtained when running the code in inference on image mode with approaches MEGA and BASE.

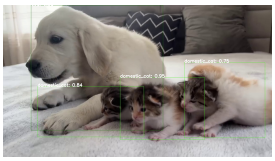
(a) First Image Sample BASE



(b) First Image Sample MEGA



(c) Sample Image 9 BASE



(d) Sample Image 9 MEGA

Fig. 2: Results for Inference on Image Folder

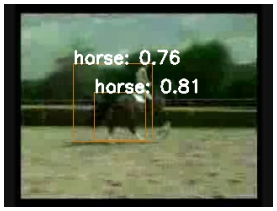
The MEGA and BASE approaches yield similar results. Overall, the MEGA approach tends to detect more objects as illustrated in Figures 2a) and 2b). Conversely, the BASE approach occasionally provides slightly more confident classification predictions as shown in Figures 2c) and 2d). Deciding which approach to use depends on the specific goals of the model's application. If the primary objective is achieving high accuracy in classifying each detected object, the BASE approach is preferable. On the other hand, if the focus is on detecting a greater number of objects regardless of the confidence level in their classification, the MEGA approach is better suited.

### B. Task 2.2

The goal of this section is to analyse the visual results obtained using the MEGA and the BASE approaches on the data. The two approaches are compared in order to establish the main differences between them. Additionally, the aim is to provide an explanation for the underlying reasons behind these results.

First, the horse video **"v_HorseRiding_g10_c01.avi"** from the UCF-101 dataset. Overall, both methods return similar results with the adequate bounding box and high confidence prediction, between 0.70 and 1 of the object being a horse. However, we notice two frames in which the MEGA approach is mistaken. In these cases, the MEGA approach distinguishes two bounding boxes instead of only one, one of these cases is pictured in Figure 3.b). For a simple case, where the video does not present any challenges the two methods perform well, the BASE approach seems to perform slightly better compared to the MEGA approach on this specific data. Both models perform very well on the data since the video presents only one object to track and there are no challenges in this video.


(a) First Frame BASE




(b) First Frame MEGA

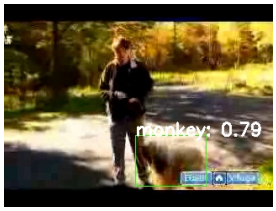

(c) Fourth frame BASE


(d) Fourth frame MEGA

Fig. 3: Results for Horse Riding Video

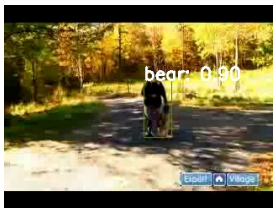
Then, the **"v_WalkingWithDog_g01_c01.avi"** video from the UCF-101 dataset. The results of the base approach feature the following classes : bear, bicycle, dog and monkey. In comparison, the results of the MEGA approach only feature the dog label which is the correct one. The MEGA approach consistently detects the dog in the video correctly. On the other hand the BASE approach misclassifies the dog in multiple frames. This is an improvement of the MEGA approach compared to the BASE approach. In the examples

given in Figure 4 c,d,e,f), the MEGA approach does not return any results when the BASE approach does. It seems that the MEGA approach does not have a result superior to the defined threshold so no detections are made. In contrast, the BASE approach gives a result but it is incorrect. This can be considered as an improvement because in certain cases it can be better to not give an result rather than give a wrong result. Finally, in certain frames as in the Figure 4.g,h) both methods output the wrong class. In this frame, both approaches classify the human being as a dog. This can be explained by the fact that the dog is situated behind the human being. In other words the dog is occluded by the human which causes the model to confuse the human with a dog.



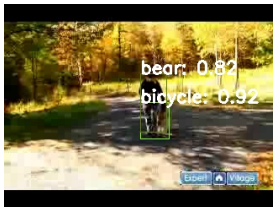

(a) Frame 227 BASE


(b) Frame 227 MEGA



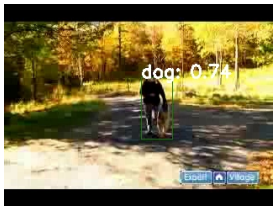

(c) Frame 10 BASE

(d) Frame 10 MEGA




(e) Frame 1 BASE

(f) Frame 1 MEGA




(g) Frame 5 BASE


(h) Frame 5 MEGA

Fig. 4: Samples of Results for Walking With Dog 01 Video

Third, the "**v_WalkingWithDog_g10_c03.avi"** video. This video contains three objects to classify : the car, the cat and the dog. First, the MEGA model often identifies one object more than the BASE model. This is visible in the frames 15 featured Figure 5 a,b). The BASE model does not detect the two animals in the video while the MEGA model identifies that there are two animals. Another difference resides in the misclassification of the bench in the video. The BASE approach classifies the bench as a bus over multiple frames while the MEGA approach only misclassifies the bench as a car in one frame. In Figure 5.c), frame 87 from the BASE approach is shown as an example of one instance where the bench is misclassifed. In figure 4.d), the frame 115 from MEGA approach is shown which is the only instance in which MEGA misclassifies the bench as a car. Then, the two approaches share some similarities. First, both misclassify the cat as a dog as can be seen in Figure 5.a,b). Then, both detect the car in the background consistently with high confidence while other objects are not detected. For instance, in Figure 5.e,f), only the car is detected as an object with high confidence (0.85 in BASE and 0.96 in MEGA) while other objects are not detected. Overall, both approaches do not show high performance. The MEGA approach is slightly better than the BASE one in that it detects more objects and also misclassifies objects less. However, even the MEGA approach misclassifies the cat repeatedly as a dog. These results can be explained by the fact that the video presents multiple challenges as explained in section IV.A.


(a) Frame 15 BASE


(b) Frame 15 MEGA


(c) Frame 87 BASE


(d) Frame 115 MEGA

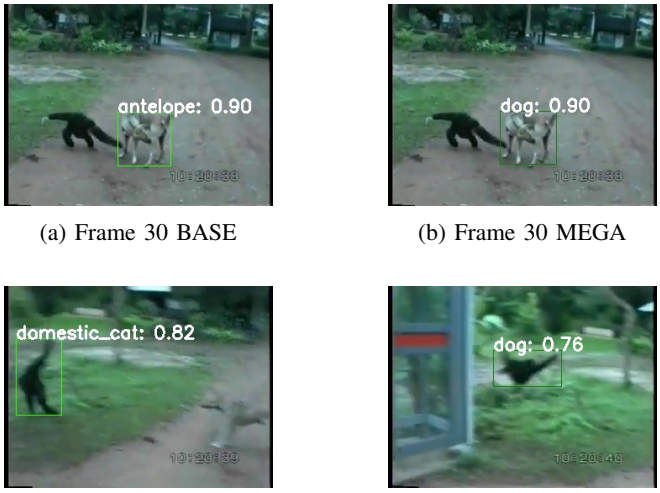

(e) Frame 192 BASE


(f) Frame 192 MEGA

Fig. 5: Results for Horse Riding Video

Finally, the video from the internet. In the BASE approach the following classes are featured in the results : dog, bird, fox, domestic cat, whale, antelope, monkey, sheep, bear. In

other words, the BASE approach classifies the dog and the monkey in the video as a mix of all the animals listed previously. The MEGA approach only uses the classes dog, monkey and cat. The MEGA approach labels the dog in the video as a dog correctly throughout the video which was not the case in the BASE approach. In Figure 6a), Frame 30 of the BASE model classifies the Dog as an Antelope. in contrast, in the same frame the MEGA approach classifies the dof correctly. The MEGA approach performs much better than the BASE approach. The only limit of the MEGA approach is when the monkey moves fast, it is labeled as a domestic cat or as a dog. This result is shown in Figure 5 c,d). This can be explained by the fact that the monkey moves fast which changes its appearance. In both frames of the video, the monkey is blurred and even a for a human being, it can be difficult to identify that it is a monkey.




(a) Frame 30 BASE

(b) Frame 30 MEGA

(c) Frame 60 MEGA

(d) Frame 68 MEGA

Fig. 6: Samples frames from the data

## VI. CONCLUSION

In conclusion, the choice between the BASE and MEGA approaches depends on the specific requirements of the application. The MEGA approach seems more suitable for scenarios with multiple objects in a video as it demonstrates a stronger ability to detect a wider range of objects. On the other hand, the BASE approach seems appropriate for easier data without significant challenges. It is worth noting that the MEGA has a slightly higher inference time compared to the BASE approach. Consequently, if the priority is fast detection and the number of detected objects is not critical, the BASE approach is the better option. However, if detecting a greater variety of objects is more important than speed, the MEGA approach seems preferable.

## VII. TIME LOG

Below the amount of time spent on each aspect of the project is detailed. The times contain both participant's time summed up.

1) Code Implementation: 10 hours, long implementation due to the modification need in the Apex Library.
2) Code Execution : 1 hour
3) Report: 10 hours
4) Miscellaneous: 3 hours, corrected the Apex files multiple times because we didn't save them correctly.

## REFERENCES

[1] Y. Chen, Y. Zhang, and Y. Wang, "Memory Enhanced Global-Local Aggregation for Video Object Detection," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, Link to paper.

[2] Scalsol, "MEGA.pytorch Demo," *GitHub repository*, 2020, Link to GitHub.

[3] "GitHub for Video Lab2" *GitHub repository*, 2024, Link to GitHub.