REPORT

In this project, we worked on the two separate tasks of :

-homogeneous and non-homogeneous image dehazing

-object detection

OBJECT DETECTION

In the first part, we focused on setting up the environment for fine-tuning a pre-existing model specialized in object detection. The model chosen is the state of the art network yolo_v5. As a preliminary stage, we also decided to experiment with fine-tuning yolo_v5 directly on the A2I2 dataset, without applying any preliminary data augmentation or dehazing techniques.

First of all, we created our custom fine-tuning dataset based on the training and dry-run set collected from the A2I2 dataset . In order to do so, it was necessary to perform a pre-processing step of the label files in order to comply with the bounding boxes coordinates format expected by the network. Second, we created a custom configuration file for the dataset, in order to set the correct paths to the folders containing images and labels. This file is located at YOLO_v5/data/A2I2.yaml.

Before proceeding, we also enabled the W&B flow for real time visualization and cloud logging, in order to be able to have a better overview of the QoR of each fine-tuning attempt.

Finally, we proceeded with several fine-tuning attempt playing with some of the hyperparameters leveraged by yolo_v5 and we finally found an optimal combination for the fine-tuning procedure to be:

-50 epochs

-batch_size=32

The results of training are located in the folder yolov5/runs/train/exp6, while the results of testing are at yolov5/runs/detect/exp8. The training results summary and an example of object detection on test images are also showed below

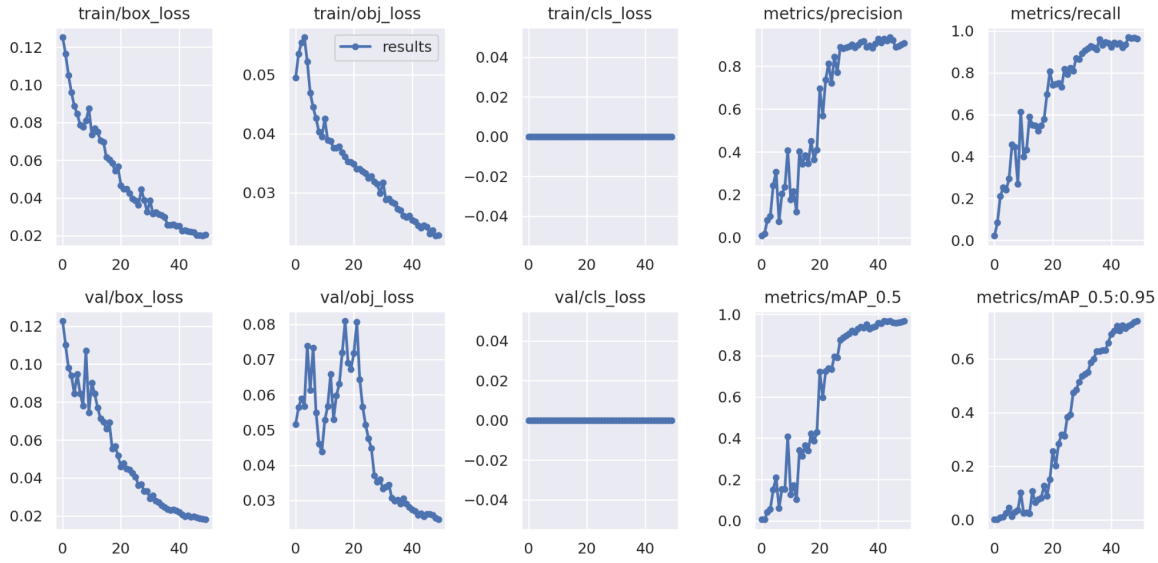


IMAGE DEHAZING

The second part of the work focused on the task of image dehazing.
After reviewing the existing literature, we decided to focus on 3 models:
-FFA.
-DWGAN

-AECR-Net

FFA.

FFA-Net (Feature Fusion Attention Network) combines Channel Attention with a Pixel Attention mechanism that treats different features and pixels unequally to exploit the differences in channel-wise features and the non-uniformity of haze distribution across different image pixels. Its architecture uses Local Residual Learning and Feature Attention in its fundamental building blocks enabling bypassing of relatively insignificant information like thin hazy regions. The different levels of Feature Fusion ordered hierarchically through Attention let the weights to be learned adaptively by giving greater significance to more important features.

The main advantage of FFA is that it supports a useful fine-tuning API which can be leveraged to improve the model previously trained on the RESIDE dataset.

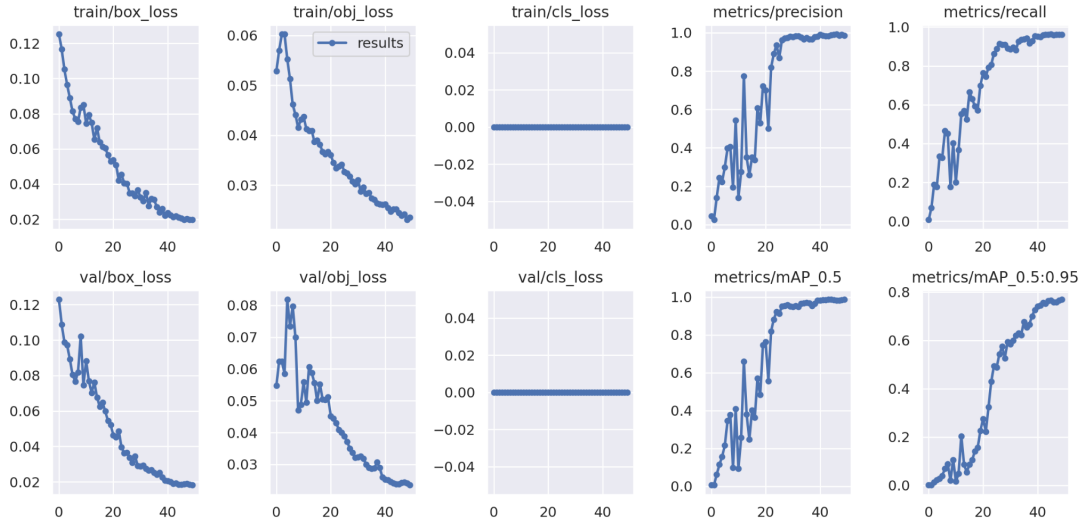
On the other hand, the problem with this model is that it is characterized by a homogeneous haze-removal approach, meaning that in practice, the model still tends to remove an equal amount of haze across the entire image. While this might not adapt well to the type of A2I2 dataset that we target, FFA has shown skyrocketing improvements over previous models, which is why we decided to dig further in the model.
In particular, we leveraged the aforementioned API to perform fine-tuning on our dataset with the following call :

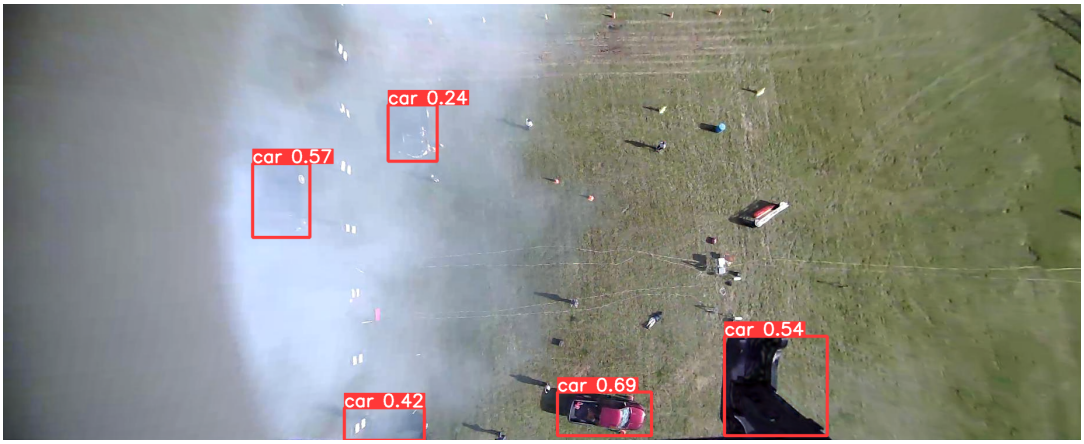
```
python main.py --net='ffa' --crop --crop_size=240 --blocks=19 --gps=3 --bs=2 --lr=0.0001
--trainset='ots_train' --testset='ots_test' --steps=1000000 --eval_step=20
```

Paying attention to decreasing the evaluation step parameter in order to update the model more often.

After that, we ran inference with the updated model on both our training and test images, and we noticed that the results slightly improved after running inferences more than one times recursively on the results from the previous inference. We then passed the arguments to the OBJ_DETECTION. So first we input as object detection both the training set and test set obtained from one cycle of FFA-net prediction, and keep the same hyperparameters of the previous experiment to see the impact of just one round of FFA-net, and then we experimented the effect of multiple inferences run recursively on both the training images and testing images, noting a slight increase in the downstream yolov5 finetuning. The directories of interest, summarized results and an example image for 3 passes of FFA inference on the input images for yolov5 finetuning are reported below:

yolov5/runs/train/exp13 yolov5/runs/detect/exp25 (conf=0.06)





This experiment, which shows a significantly higher mAP estimate, was erroneously not submitted to codalab.

DWGAN

-Given the poor performance increase in terms of mAp with FFA, we decided to place some additional effort in exploring other models which adopt non-homogeneous haze removal techniques. Among those, we found DWGAN net to be an interesting test case. DW-GAN uses a 2D discrete wavelet transform with a focus on non-homogeneous dehazing. It uses a two-branch network to deal with dehazing problems like loss of texture details due to complicated haze distribution and overfitting due to lack of training data.The DWT (Discrete Wavelet Transform) branch can retain more high-frequency knowledge in feature maps and to prevent over-fitting, ImageNet pre-trained Res2Net is adopted in the knowledge adaptation branch.

The main drawbacks coming with this models are the followings:
- the model accepts 1600x1200 images as input, thus it was necessary to preprocess our training and testing images by applying both resizing and padding. In the same way, we had to reformat the dehazed images before feeding them to YOLOv5.

-The model does not offer a fine-tuning API. For this reason, it was not possible to improve the model by adapting it to our specific domain, which is unfortunate since most of the images of the datasets used for training this model (RESIDE., NH-HAZE and DENSE-HAZE) were not taken with the top-down view which characterizes our A2I2 dataset.

Although the model is not optimal for our purposes, it is still offering promising results in the documentation provided by the creators, which is why we used it to augment our dataset with multiple inference experiments.

The summary of the results after applying object detection are showed in swept-firefly-8 experiment (yolo_v5/runs/train/exp10 and yolo_v5/runs/detect/exp28)

-AECR-Net

This model implements contrastive regularization (CR) built upon contrastive learning with an approach that looks at hazy images and clear images as negative and positive samples, respectively. Contrastive regularization aims to pull the restored image closer to the clear image and away from the hazy image in the latent space. It uses a dehazing network based on an autoencoder-like (AE) framework which implements adaptive mixup operation that benefits from preserving information flow adaptively and a dynamic feature enhancement module that improves the network's transformation capabilities.

First , we just want to see the impact of dehazing and how it affects the downstream object detection quality. So we just run inference on our training and test data (similarly to what was done for DWGAN). Then we pass the dehazed training images to the finetuning of YOLO and finally perform inference with the fine tuned yolo model on the dehazed test images.

SAme setting (50 epochs and 32 batch_size) yolo_v5/runs/train/exp15 (leafy-due on w&b) yolo_v5/runs/detect/exp29. Submission map 65.618

AUGMENTED DATASET EXPERIMENTS:

At this point, we decided to actually leverage an augmented dataset composed of different training sets obtained by the dehazing methods described above:

1st experiment:

-initial training set + DWGAN-dehazed training set

-DWGAN-dehazed test set

epochs=70

batch_size=40

runs/train/exp12 yolo_v5/runs/detect/exp21 (conf=0.1) exp22 (conf=0.3) **exp26(conf=0.06 submission map 70,61)** exp27(conf=0.04) (wandb folder volcan-rain)

SHOW HOW **EXP26** IS BETTER THAN EXP28 THANKS TO DATA AUGMENTATION (ex: new_062.jpg, and then prove it by showing the two submission) example

It was interesting that these settings improve the performance with respect to exp26 which only uses DWGAN images.

1b experiment:

ORIGINAL + DWGAN + FFA3

2nd experiment:

initial training set = tre sets of images obtained from FFA (tested recursively from 1 to 3 times)

epochs=50

batch_size=32

runs/train/exp16

3th experiment:

Further data augmentation: combine in the training set DWGAN, FFA, and AECRNET outputs