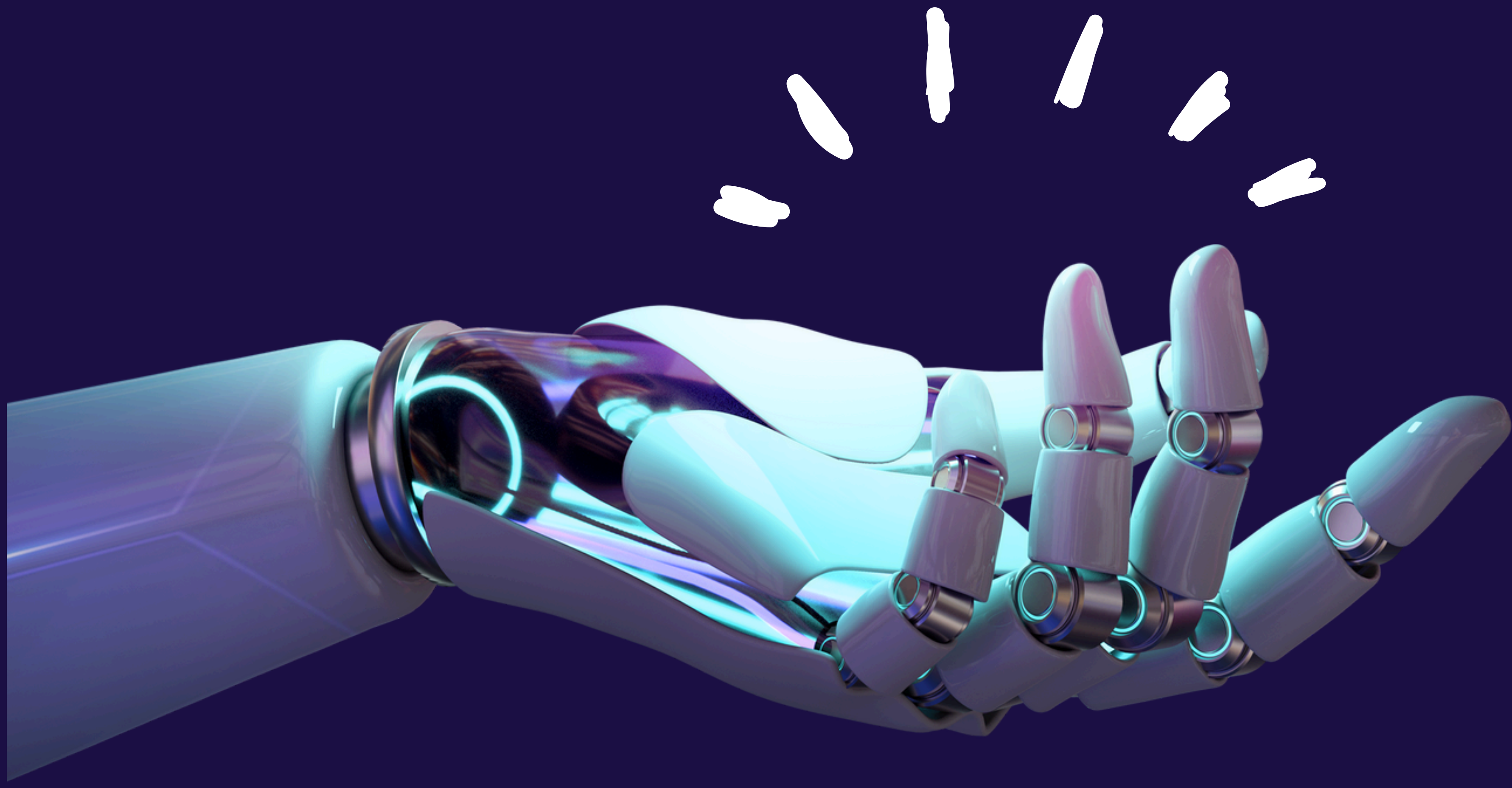
TKO_7096

Computer vision & sensor fusion

ASSIGNMENT 3: **TENSORFLOW OBJECT DETECTION API**

Dataset used: Satellite image dataset

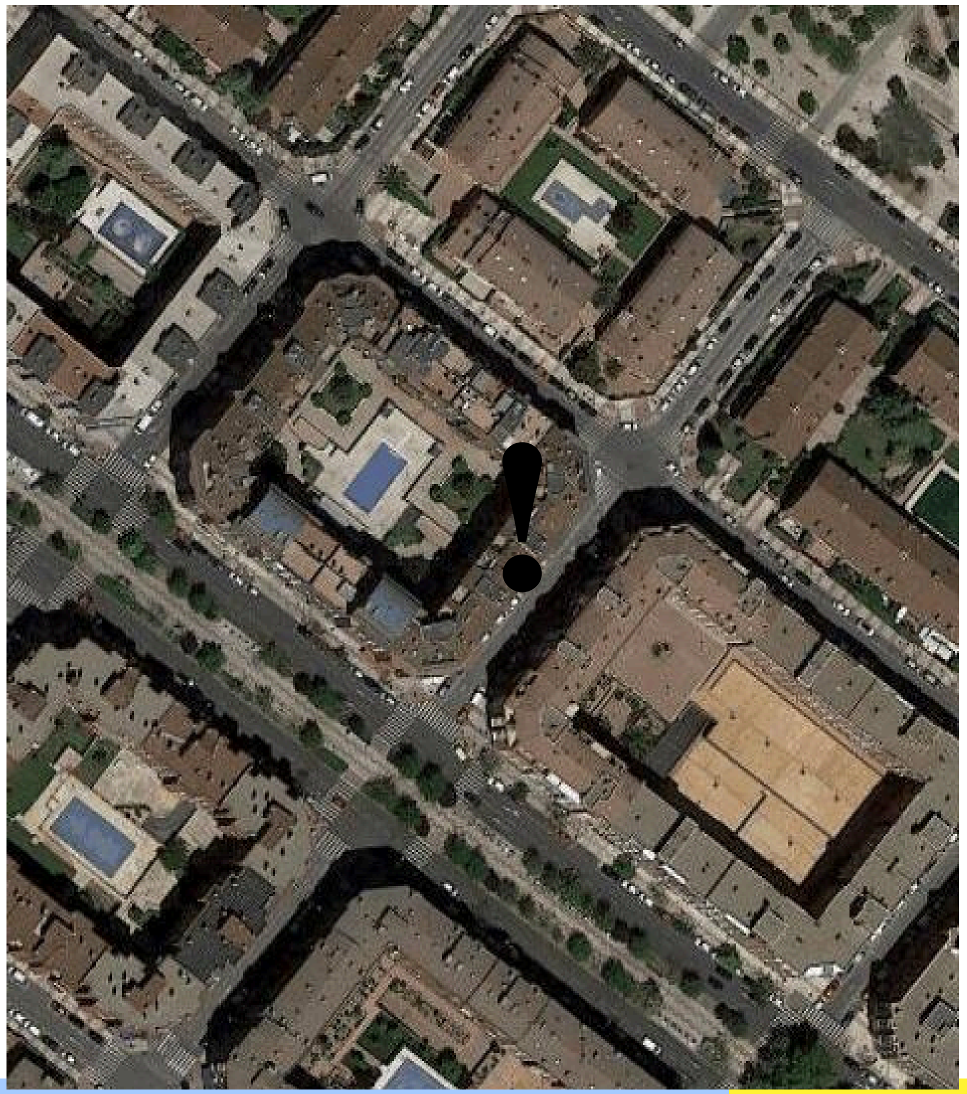


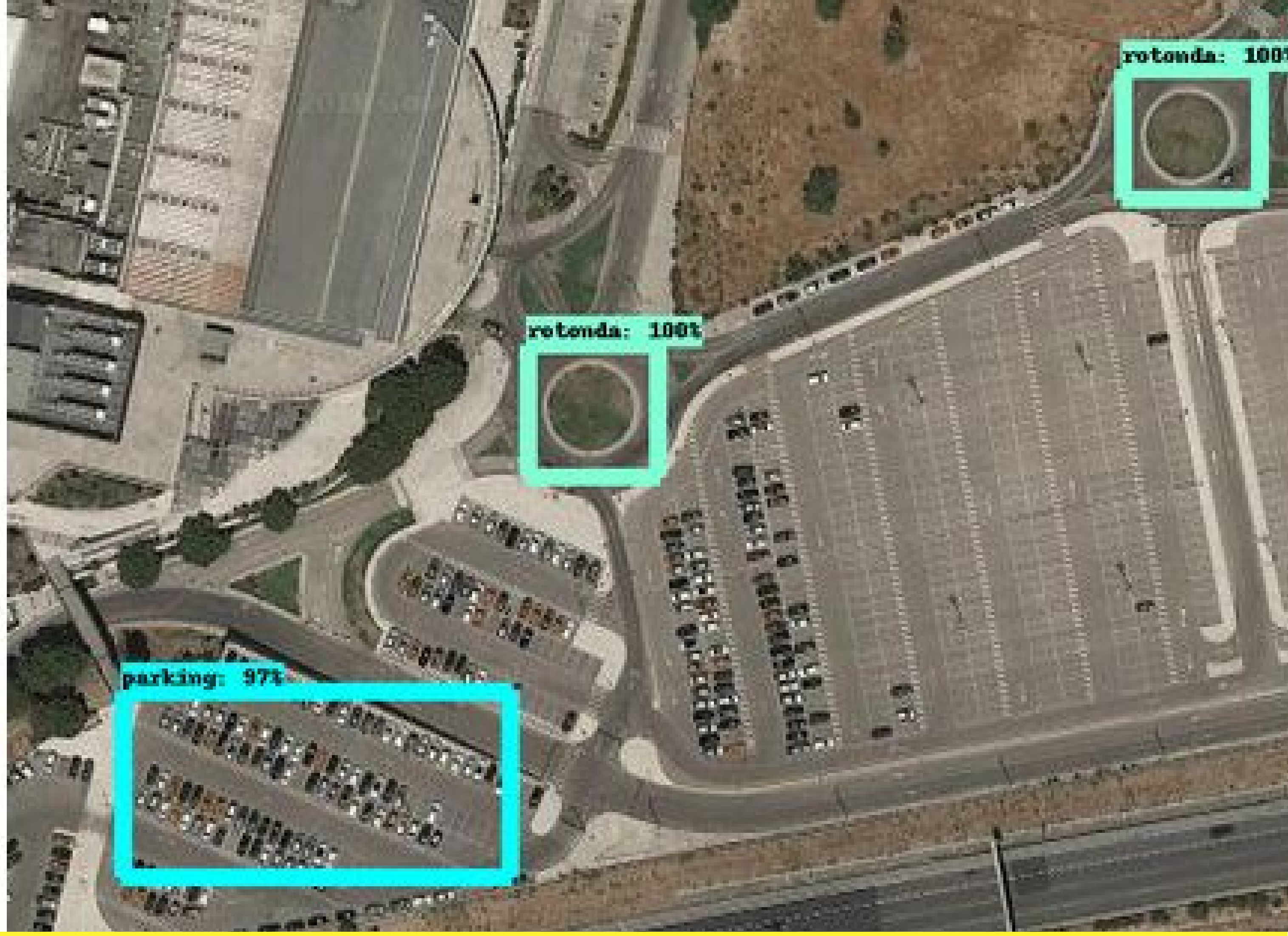
Submitted by:
Faiza Anan Noor
03/03/2024

UNIVERSITY OF TURKU

Dataset Information

Satellite images self annotated from Google maps presumably from different areas in spain.





The data had 3 object detection classes:

- **Piscina** = swimming pool
- **Rotonda** = traffic circle/ roundabouts
- **Parking** = parking

Train and val folders had one folder for .jpg images and then their corresponding annotations in xml files

Training: 200 satellite images
Validation: 40 images
Resolution: 640x640
https://www.kaggle.com/datasets/ancaco12/aerial-satellite-images

Visual inspection: Most of them are conventional.

- Pools mostly Rectangular
- Not considering uniqueness, eg. round parking lots, less instances of oval or varying roundabout shapes.
- Images are direct top down views, not slanting
- number of images is small for training and learning from such complex features
- Even with naked eye, sometimes I could not analyze parking spots. Mistook blue rectangular building spaces as pools

Implementation Details:

Steps:

- In this assignment, object recognition tasks on satellite imagery were carried out using TensorFlow's Object recognition API. Installing the required libraries and frameworks was the first step in environment configuration. Followed instructions in Slide 6.2.
- To prepare the dataset for usage in TensorFlow, I used the XML files, converted them to .CSV and then to the TFRecord format. We made sure the images are also .jpg.
- Configured the model configs(created & specified label paths, num_classes, fine_tune_checkpoin path, fine_tune_checkpoint_type, tfrecords for train, val, file and then trained and exported the model.

```
item{
 id:1
 name:'piscina'
}
item{
 id:2
 name:'parking'
}
item{
 id:3
 name:'rotonda'
}
```

satellite_label_map .pbtxt

Sample training log at 1200th step

```
LOSS/BOXCLASSIFIERLOSS/CLASSIFICATION_LOSS': 0.09342663,
 'LOSS/BOXCLASSIFIERLOSS/LOCALIZATION_LOSS': 0.1551219,
 'LOSS/RPNLOSS/LOCALIZATION_LOSS': 0.05255309,
 'LOSS/RPNLOSS/OBJECTNESS_LOSS': 0.0038900943,
 'LOSS/REGULARIZATION_LOSS': 0.0,
 'LOSS/TOTAL_LOSS': 0.30499172,
 'LEARNING_RATE': 0.0293332
```

Problems Faced:

1. TensorFlow v2 has many incompatibilities and version problems especially related to eager mode execution and executing “create_tfrecord.py” so used Regular Expressions(python package : re) to handle those
2. Google Colab used but its CPU takes forever to run.
3. There were so many instances of RAM limit excession so limited batch size and train steps and restarted.
4. The GPU takes long also and GPU shuts off before 12 hours of usage so created 3+ accounts for running.

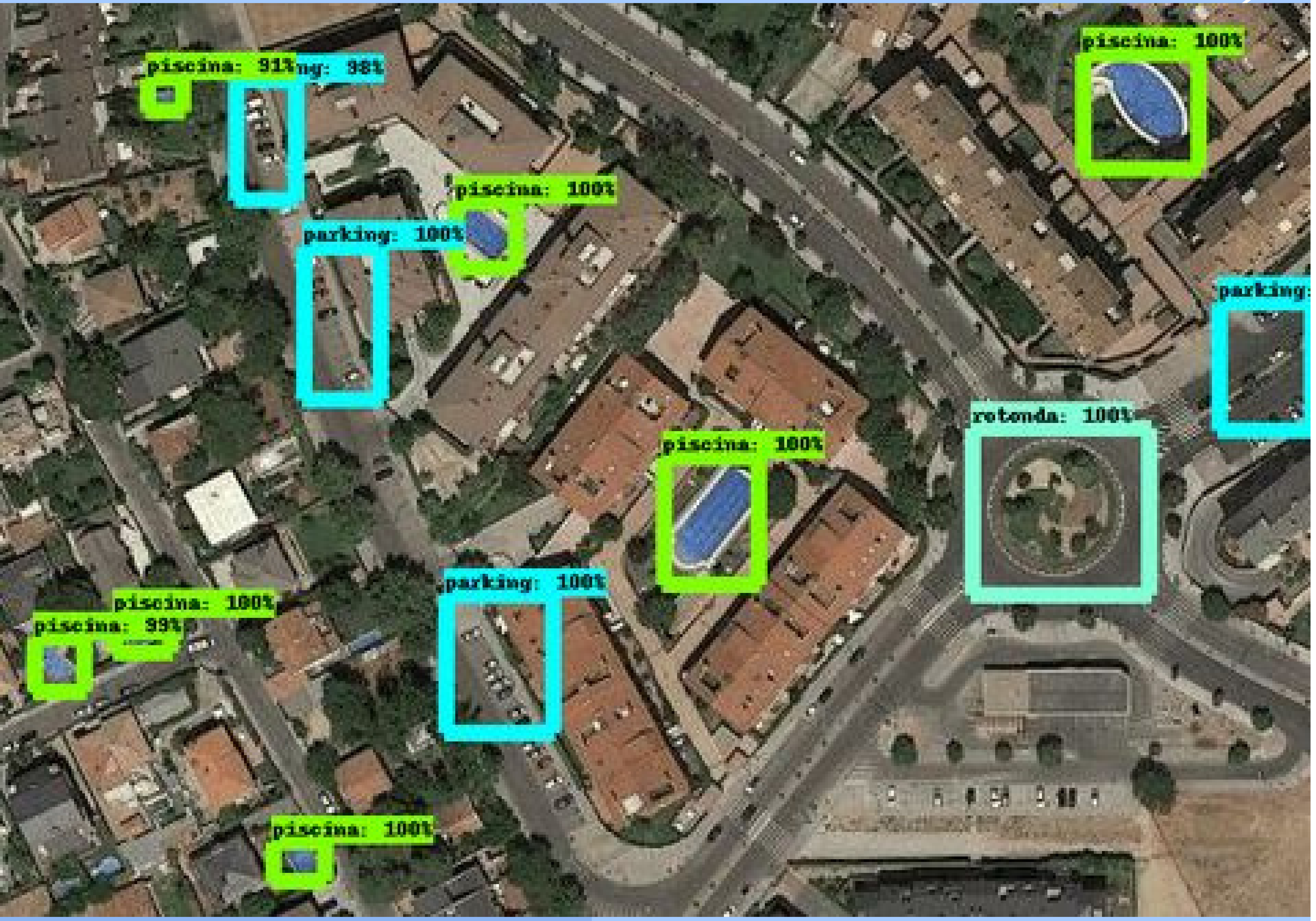
Number of steps 1200	tensorflow==2.13.0
Device used Google Colabs CPU and NVIDIA Tesla K80 Gpu, 12GM RAM	Batch Size: 08 Learning rate: adjusted Model: Faster R-CNN with Resnet-50 (v1)

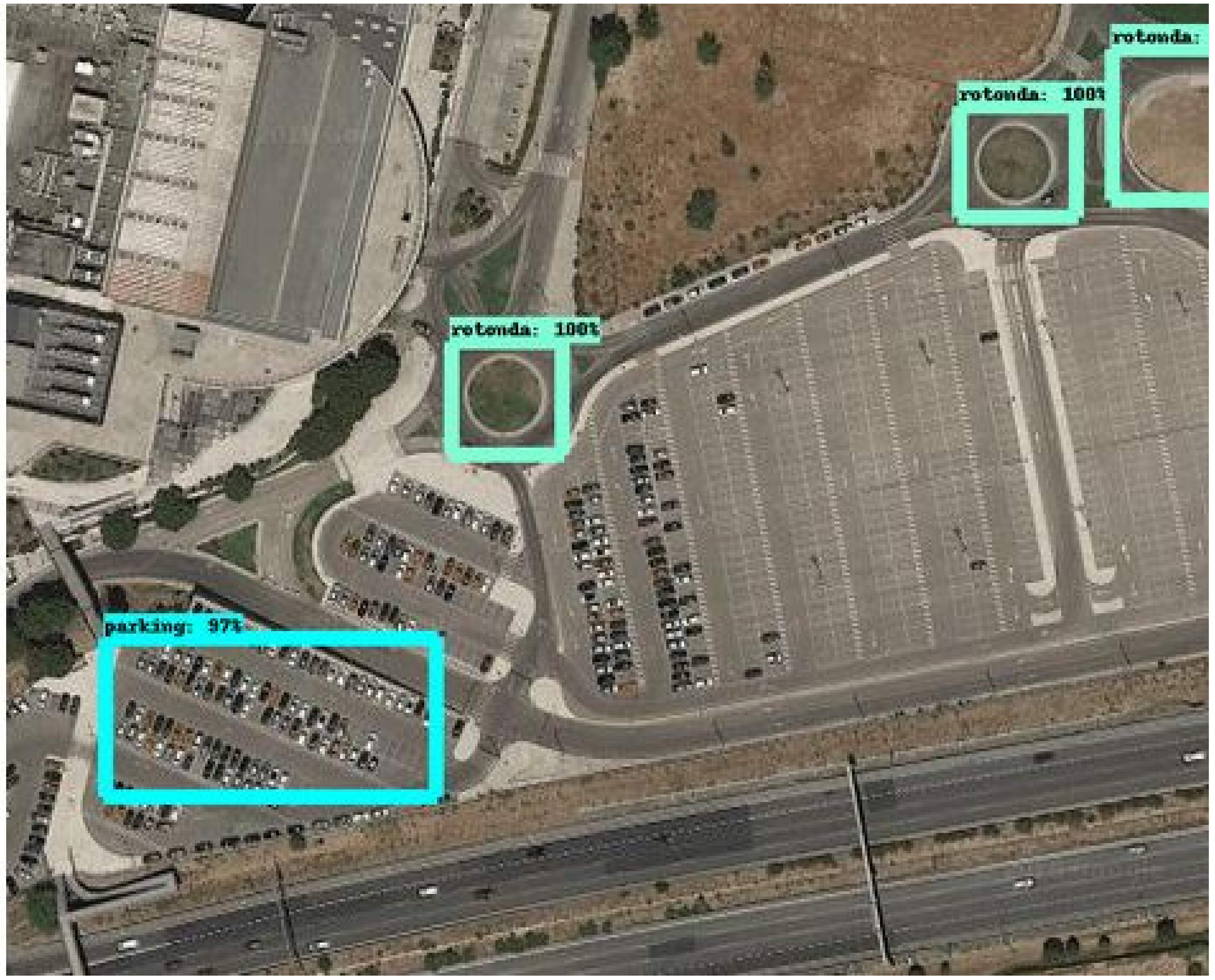
Model Evaluation

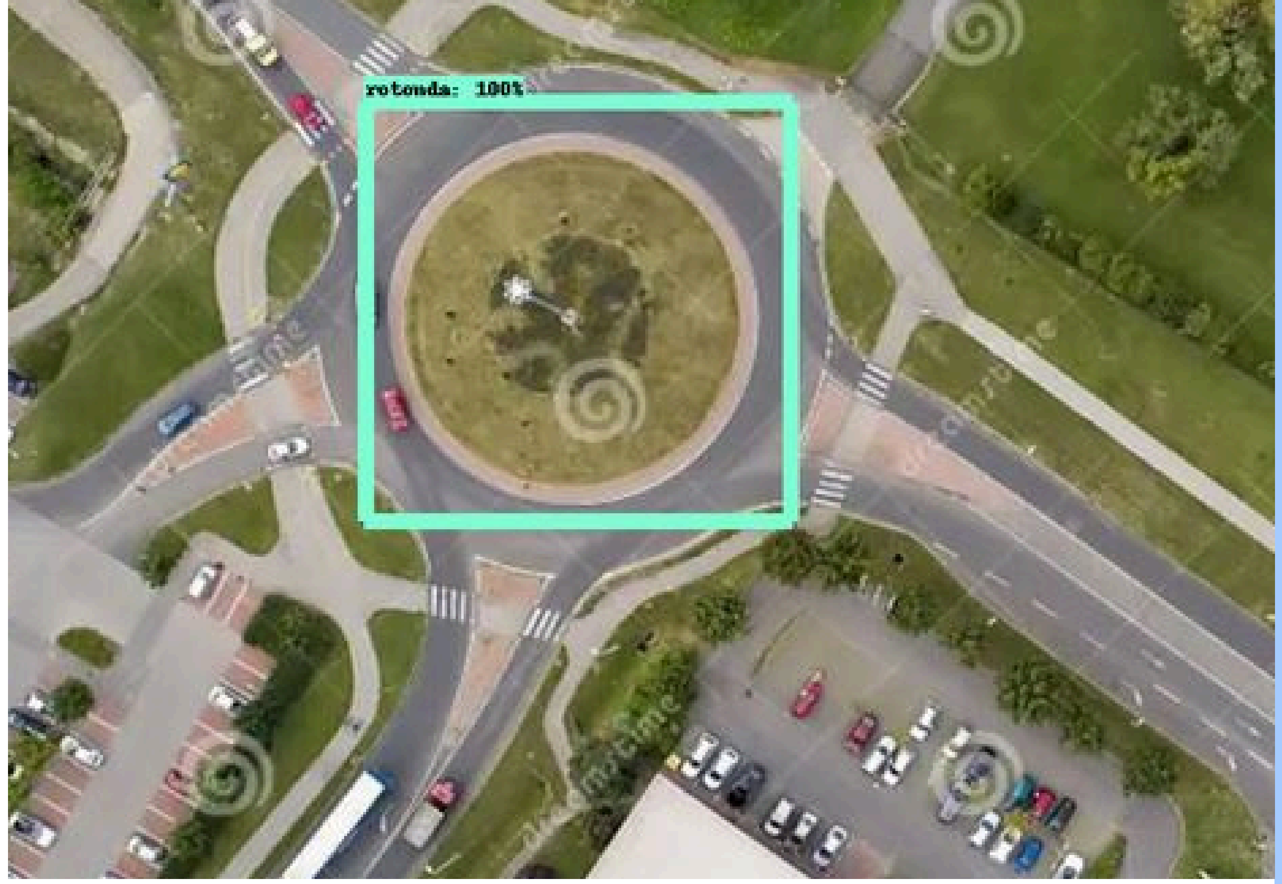
The trained model was exported with its variables, pb file etc and loaded in "**object_detection_tutorial.ipynb**" and evaluated against some test images. Fixed all paths and locations to labels, model directory, removed unnecessary functions and tested on some images

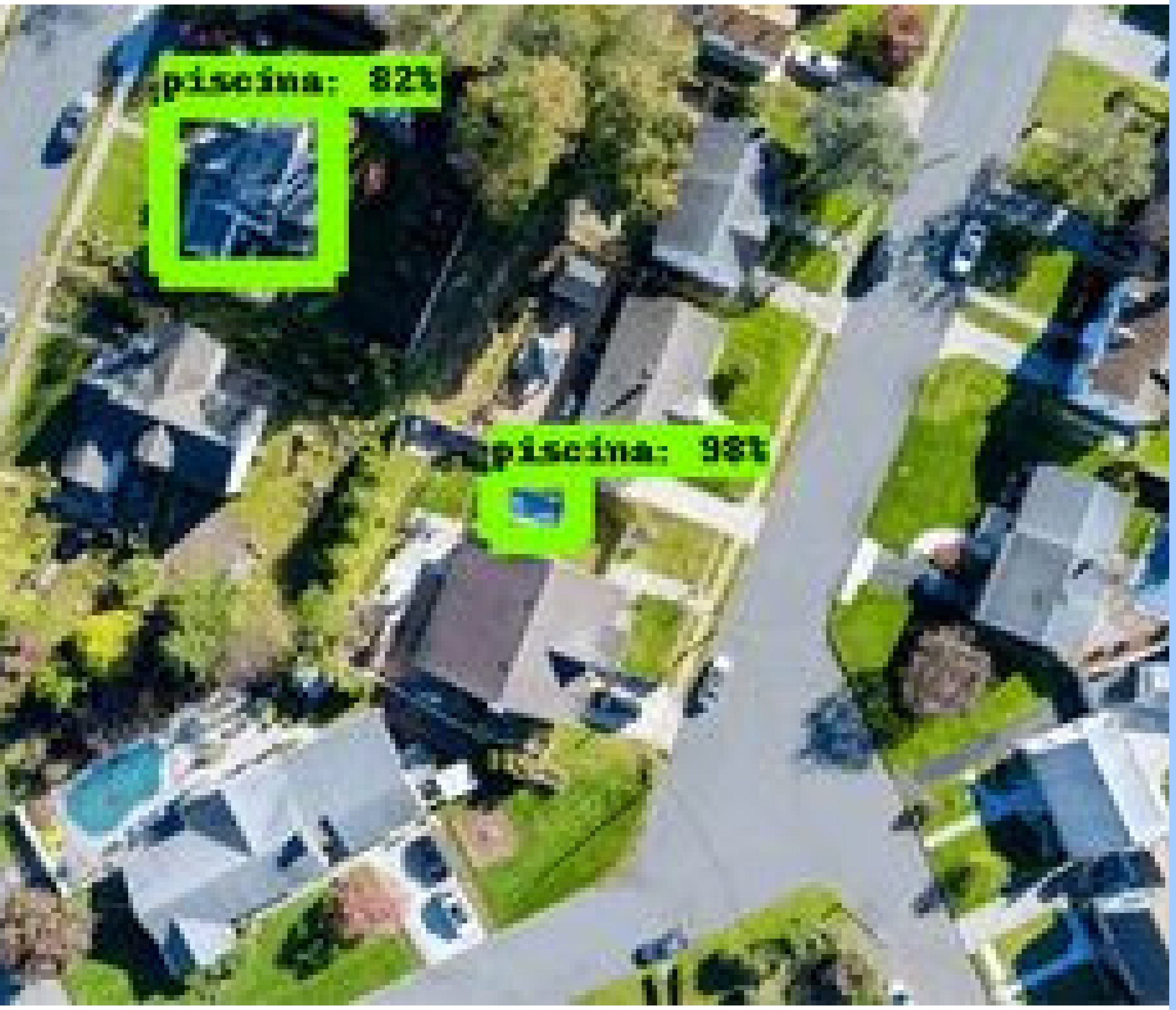
Examples in this slide show how it successfully detected parking, swimming pools and roundabouts for some images that are similar to the ones faced during training.

For images in the left in the next slide, it could detect some instances of rotonda, pool but not all instances of pool, parking etc. So our model did well on test images similar to the ones it faced during training but not other types.l









Project Conclusion

With several pre-trained models and customizable parameters to satisfy specific requirements, TensorFlow's Object Detection API provides a flexible and powerful tool for performing object identification tasks, despite having a challenging implementation. Although it is a highly strong tool, it must be handled carefully and compatibilities should be handled with properly.

Possible improvements & Directions:

1 **Bigger batch size:** more samples during training
2) **More data with more variations included.** Greater in-category variance necessitates greater data needs and taking into account if distinguishing characteristics suffice.
3) Training for a **longer period/steps** to allow efficient learning
4) Use techniques like k-fold **cross-validation** to evaluate the model's performance on different subsets of the data and ensure robustness.
5) Use of **robust devices** such as powerful GPU for unlimited training is a necessity for using this API.

Thank you!