Sandra Pinel Varona y Alicia Sanguino Rodríguez

TRABAJO 2 - PROCESAMIENTO DE IMAGEN IA:

La tarea consiste en la detección de objetos con Deep Learning.

En este caso concreto, la base de datos va a estar compuesta de imágenes de cráteres en Marte y la Luna en las cuales: Existe una superficie que no es lisa completamente con ciertas protuberancias/alteraciones/modificaciones; dichas modificaciones presentan geometrías variadas, normalmente circulares; y hay que tener en cuenta que no todo en la superficie son objetos a detectar de interés.

El objetivo consistirá en hacer un modelo de detección de objetos que, ante una imagen de una superficie lunar/marciana, nos recuadre dónde se encuentran los posibles cráteres.

Dicho objetivo se lleva a cabo a través de estos pasos:

1. Hacer funcionar el código con los datos dados (el proporcionado por kaggle).
2. Modificar el código para comparar algoritmo.
    - Cambiar 'torchvision.models.detection' por otro modelo de detección.
    - Cambiar la función nueva 'get_model_bbox_alternativo()' que sea igual a 'get_model_bbox()' salvo cambiando el modelo a entrenar.
    - Entrenar el modelo y hacer predicciones. Dichas predicciones deberán ir pintadas en verde y se representarán de manera conjunta con el ground truth y el resultado del modelo que viene en el notebook original.

1. Hacer funcionar el código proporcionado en Kaggle:

In [ ]:
```
# Download TorchVision repo to use some files from
# references/detection
!pip install gdown
!gdown https://drive.google.com/uc?id=1hrHgANwgC8VyXLhXgxLYHTrILpeJ5bLl
!unzip /content/mars_and_moon.zip

!git clone https://github.com/pytorch/vision.git
!cd vision
!git checkout v0.8.2

!cp ./vision/references/detection/utils.py ./
!cp ./vision/references/detection/transforms.py ./
!cp ./vision/references/detection/coco_eval.py ./
!cp ./vision/references/detection/engine.py ./
!cp ./vision/references/detection/coco_utils.py ./

!pip install cython
# Install pycocotools, the version by default in Colab
!pip install -U 'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'
!pip install -U albumentations
!pip install -U opencv-python

#Copy and unify the train and validation datasets into one folder for images and another for labels
!mkdir ./train
!cp -a /kaggle/input/martianlunar-crater-detection-dataset/craters/train/images/. ./train/images/
!cp -a /kaggle/input/martianlunar-crater-detection-dataset/craters/valid/images/. ./train/images/
!cp -a /kaggle/input/martianlunar-crater-detection-dataset/craters/train/labels/. ./train/labels/
!cp -a /kaggle/input/martianlunar-crater-detection-dataset/craters/valid/labels/. ./train/labels/
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (4.6.6)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.13.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from gdown) (1.16.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.1)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.11.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->g
down) (2.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from reques
ts[socks]->gdown) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests[socks]->g
down) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests[soc
ks]->gdown) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests[soc
ks]->gdown) (2023.7.22)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from requests
[socks]->gdown) (1.7.1)
Downloading...
From: https://drive.google.com/uc?id=1hrHgANwgC8VyXLhXgxLYHTrILpeJ5bLl
To: /content/mars_and_moon.zip
100% 77.6M/77.6M [00:02<00:00, 26.1MB/s]
Archive:  /content/mars_and_moon.zip
  inflating: best.pt
  inflating: craters/test/images/010_png.rf.fcf5e274562ee69a325f9d7a0b30767f.jpg
```

```
inflating: craters/test/images/015_png.rf.7d5b2091b6339c9480a171a59c52c3b9.jpg
inflating: craters/test/images/019_png.rf.1930cd277f9bf0e3fa57f2dcfee0385f.jpg
inflating: craters/test/images/04_png.rf.81a7d6cbeb9dc09e5a8ecd40e185fc92.jpg
inflating: craters/test/images/mars_crater--100-_jpg.rf.a2ad5867efb2d73e86d9d980ca40a9fe.jpg
inflating: craters/test/images/mars_crater--108-_jpg.rf.9395f473f249e064dbfea078e2519a17.jpg
inflating: craters/test/images/mars_crater--116-_jpg.rf.2e550a693a8800808e68848484716b95.jpg
inflating: craters/test/images/mars_crater--117-_jpg.rf.b412b6593d102c5f9dda7bce79bb815c.jpg
inflating: craters/test/images/mars_crater--12-_jpg.rf.26060e7ca4ca8781bcaf011b9b70b7db.jpg
inflating: craters/test/images/mars_crater--25-_jpg.rf.a06bfa24b404b064ead471f56d636e0e.jpg
inflating: craters/test/images/mars_crater--31-_jpg.rf.5030dfd316db1570d5f2ad396fa87d4d.jpg
inflating: craters/test/images/mars_crater--32-_jpg.rf.2f0302a2f32a7b05f5c55e3dac97f1ea.jpg
inflating: craters/test/images/mars_crater--33-_jpg.rf.baf9c7be01ba1f4b954e0e5be0560c80.jpg
inflating: craters/test/images/mars_crater--46-_jpg.rf.369e88a43859f97a595d2091746600f4.jpg
inflating: craters/test/images/mars_crater--51-_jpg.rf.2f21cd1782f9da8bb8f1ada8efa134a7.jpg
inflating: craters/test/images/mars_crater--60-_jpg.rf.df89b1f82e171f73f5402c29b6637aa8.jpg
inflating: craters/test/images/mars_crater--65-_jpg.rf.34ab64f1f6ce5a4b360ea66eae77d4b9.jpg
inflating: craters/test/images/mars_crater--66-_jpg.rf.85564f58f27e684c94c8a0e484cc289e.jpg
inflating: craters/test/images/mars_crater--97-_jpg.rf.63347c2ec963cf5c4ab641e1ba872df1.jpg
inflating: craters/test/labels/010_png.rf.fcf5e274562ee69a325f9d7a0b30767f.txt
inflating: craters/test/labels/015_png.rf.7d5b2091b6339c9480a171a59c52c3b9.txt
inflating: craters/test/labels/019_png.rf.1930cd277f9bf0e3fa57f2dcfee0385f.txt
inflating: craters/test/labels/04_png.rf.81a7d6cbeb9dc09e5a8ecd40e185fc92.txt
inflating: craters/test/labels/mars_crater--100-_jpg.rf.a2ad5867efb2d73e86d9d980ca40a9fe.txt
inflating: craters/test/labels/mars_crater--108-_jpg.rf.9395f473f249e064dbfea078e2519a17.txt
inflating: craters/test/labels/mars_crater--116-_jpg.rf.2e550a693a8800808e68848484716b95.txt
inflating: craters/test/labels/mars_crater--117-_jpg.rf.b412b6593d102c5f9dda7bce79bb815c.txt
inflating: craters/test/labels/mars_crater--12-_jpg.rf.26060e7ca4ca8781bcaf011b9b70b7db.txt
inflating: craters/test/labels/mars_crater--25-_jpg.rf.a06bfa24b404b064ead471f56d636e0e.txt
inflating: craters/test/labels/mars_crater--31-_jpg.rf.5030dfd316db1570d5f2ad396fa87d4d.txt
inflating: craters/test/labels/mars_crater--32-_jpg.rf.2f0302a2f32a7b05f5c55e3dac97f1ea.txt
inflating: craters/test/labels/mars_crater--33-_jpg.rf.baf9c7be01ba1f4b954e0e5be0560c80.txt
inflating: craters/test/labels/mars_crater--46-_jpg.rf.369e88a43859f97a595d2091746600f4.txt
inflating: craters/test/labels/mars_crater--51-_jpg.rf.2f21cd1782f9da8bb8f1ada8efa134a7.txt
inflating: craters/test/labels/mars_crater--60-_jpg.rf.df89b1f82e171f73f5402c29b6637aa8.txt
inflating: craters/test/labels/mars_crater--65-_jpg.rf.34ab64f1f6ce5a4b360ea66eae77d4b9.txt
inflating: craters/test/labels/mars_crater--66-_jpg.rf.85564f58f27e684c94c8a0e484cc289e.txt
inflating: craters/test/labels/mars_crater--97-_jpg.rf.63347c2ec963cf5c4ab641e1ba872df1.txt
inflating: craters/train/images/011_png.rf.8ac312b4898f0106d10b76952a55d237.jpg
inflating: craters/train/images/012_png.rf.64da6ff4c62638096ee6e5bf689706bc.jpg
inflating: craters/train/images/013_png.rf.ee44d5aa33fd33a1ed62ae233180f505.jpg
inflating: craters/train/images/016_png.rf.1973f9540ae7f672257609a8e5721ab3.jpg
inflating: craters/train/images/017_png.rf.1504c0d3ecbf20af6bc5114ca197a0dd.jpg
inflating: craters/train/images/018_png.rf.2d4eed5581681fe83830e51634befdaf.jpg
inflating: craters/train/images/01_png.rf.4d2ebc5ed98ad1e69d667aadbce63d53.jpg
inflating: craters/train/images/020_png.rf.ce87f4889d7441275135633392f98ed7.jpg
inflating: craters/train/images/022_png.rf.00ae4e655a2774bada1e254641482935.jpg
inflating: craters/train/images/02_png.rf.610687947e4c92f77e6462104ec4b924.jpg
inflating: craters/train/images/03_png.rf.8f7b31e14642026833b7c0dcd1832862.jpg
inflating: craters/train/images/05_png.rf.844343145246e51e66a345419e1862bf.jpg
inflating: craters/train/images/06_png.rf.aaf8c66b9e4d5e99a3dc70bae7f62c07.jpg
inflating: craters/train/images/07_png.rf.3fba5e95c6827aa6c4132e70f2086555.jpg
inflating: craters/train/images/08_png.rf.944efdd0f108140b368d7ad2c37426df.jpg
inflating: craters/train/images/09_png.rf.3b796e77a5f0036af4cd4413fcbe07a5.jpg
inflating: craters/train/images/mars_crater--0-_jpg.rf.40c1dec94c66ab07d9da8c74fa58d6f8.jpg
inflating: craters/train/images/mars_crater--10-_jpg.rf.585b1aa305997e3055e86fcac72a806b.jpg
inflating: craters/train/images/mars_crater--101-_jpg.rf.8f4eb1c77ab9e64d2fd691a6e0fcd3ec.jpg
inflating: craters/train/images/mars_crater--102-_jpg.rf.9cdbcc724e2e7baf5c5c17aed91c769b.jpg
inflating: craters/train/images/mars_crater--103-_jpg.rf.61597458910fb0eed55b415c9cbac3bf.jpg
inflating: craters/train/images/mars_crater--105-_jpg.rf.338dccc756e50430460dfb6f6191d1c5.jpg
inflating: craters/train/images/mars_crater--106-_jpg.rf.05aa3dbbbdc9651cb8c51d26c2847805.jpg
inflating: craters/train/images/mars_crater--109-_jpg.rf.29afc1137114a2b872f484326254e949.jpg
inflating: craters/train/images/mars_crater--11-_jpg.rf.9e8de286e3d84938f34bb6bf35825343.jpg
inflating: craters/train/images/mars_crater--110-_jpg.rf.593f6a3d9aed98e7a08955e700765222.jpg
inflating: craters/train/images/mars_crater--111-_jpg.rf.215f88d6bd7a6d86560754be694fc2f3.jpg
inflating: craters/train/images/mars_crater--113-_jpg.rf.a7713d55de03a1a7487429d414e1005c.jpg
inflating: craters/train/images/mars_crater--118-_jpg.rf.0a8b3fb0e1332e576901e596ad55e30a.jpg
inflating: craters/train/images/mars_crater--119-_jpg.rf.5cd500a3dcc977038a875b5115f9eda4.jpg
inflating: craters/train/images/mars_crater--120-_jpg.rf.a870131b0b5974e8351a32ade2d5f0f3.jpg
inflating: craters/train/images/mars_crater--14-_jpg.rf.e050bff73578e2a9ad30e339b898f839.jpg
inflating: craters/train/images/mars_crater--15-_jpg.rf.c669035a909b835be84fa4fc67bf3f40.jpg
inflating: craters/train/images/mars_crater--16-_jpg.rf.bf5f6796b4ec5d351b47efae2e866934.jpg
inflating: craters/train/images/mars_crater--17-_jpg.rf.c051d4c804f2fed2aa32999cc9fd0b48.jpg
inflating: craters/train/images/mars_crater--18-_jpg.rf.8214037c811619bbe37024a557c9251f.jpg
inflating: craters/train/images/mars_crater--19-_jpg.rf.a9e7485a270816a553c3d7e0c3fddd30.jpg
inflating: craters/train/images/mars_crater--2-_jpg.rf.a65bb7ee68968509ac0b28df75e1f251.jpg
inflating: craters/train/images/mars_crater--20-_jpg.rf.cb4c2603f4a62bbe96328c29cec6d68e.jpg
inflating: craters/train/images/mars_crater--21-_jpg.rf.63688e718106a9b061bb702a298e7a54.jpg
inflating: craters/train/images/mars_crater--22-_jpg.rf.3ed23daa3923c22a459593cb449a7336.jpg
inflating: craters/train/images/mars_crater--24-_jpg.rf.4614bb6844fda70ea8a3b4f515cb7e8f.jpg
inflating: craters/train/images/mars_crater--26-_jpg.rf.e34074ddcd6c44fa17b7a7689f7ddfa4.jpg
inflating: craters/train/images/mars_crater--27-_jpg.rf.158fc0c6a694bba2f0afa44fd76a7f5a.jpg
inflating: craters/train/images/mars_crater--3-_jpg.rf.17bb27df5c5e6a501f790342675bc145.jpg
inflating: craters/train/images/mars_crater--30-_jpg.rf.0fb68bf5aece1859bc5f080e66b4fef1.jpg
inflating: craters/train/images/mars_crater--34-_jpg.rf.5617051b3d6a801330f8d4f7df9e3239.jpg
inflating: craters/train/images/mars_crater--35-_jpg.rf.826bc4b2a3a69c48144834f40d92439f.jpg
inflating: craters/train/images/mars_crater--36-_jpg.rf.0100a780ad4217b29c2dc9b46deec040.jpg
inflating: craters/train/images/mars_crater--37-_jpg.rf.c2a35d9de333416eb4d6d8f0d6b3310a.jpg
inflating: craters/train/images/mars_crater--38-_jpg.rf.e6f3636efb075ba61dafa478e6b21de2.jpg
inflating: craters/train/images/mars_crater--39-_jpg.rf.dbcaca888781a42f7bb3ce2c922da228.jpg
```

```
inflating: craters/train/images/mars_crater--4-_jpg.rf.b31f215c27d917bdb7c7c545db2bc36b.jpg
inflating: craters/train/images/mars_crater--41-_jpg.rf.5b367c681ccf0dfa9c5fddf287926d83.jpg
inflating: craters/train/images/mars_crater--42-_jpg.rf.0cea1f9d9cf45fb141abf4e9aaae4c84.jpg
inflating: craters/train/images/mars_crater--43-_jpg.rf.27ea7a65603205e491bc439c1d654cf4.jpg
inflating: craters/train/images/mars_crater--44-_jpg.rf.627a604f51a4479d1f1660c5cf2af5c4.jpg
inflating: craters/train/images/mars_crater--45-_jpg.rf.d69fa5e159ad8e5fe17b96deaba99c18.jpg
inflating: craters/train/images/mars_crater--48-_jpg.rf.cc5528a8dcf4a11085a0a01d5c9084bc.jpg
inflating: craters/train/images/mars_crater--49-_jpg.rf.5ea4703986c80bcfa03f0f99e4067c6f.jpg
inflating: craters/train/images/mars_crater--5-_jpg.rf.6acb650edc64d6e1a02a0eef00e95113.jpg
inflating: craters/train/images/mars_crater--50-_jpg.rf.02be16d199897982df8e6ea8f3e5818f.jpg
inflating: craters/train/images/mars_crater--52-_jpg.rf.1a227f21c518775052f847053022cd86.jpg
inflating: craters/train/images/mars_crater--53-_jpg.rf.20acc76fad4fecd456d4b0e77c9bbbb1.jpg
inflating: craters/train/images/mars_crater--54-_jpg.rf.2313c9f439999313bfb527534bbdc501.jpg
inflating: craters/train/images/mars_crater--55-_jpg.rf.452ea46e23156b2c86433eb9054b1a6a.jpg
inflating: craters/train/images/mars_crater--56-_jpg.rf.6f0dcbff4ea66014aef178b840d29238.jpg
inflating: craters/train/images/mars_crater--57-_jpg.rf.a783aaeaf00ae5d57bec64438050285f.jpg
inflating: craters/train/images/mars_crater--58-_jpg.rf.d668b2f63bd183cb629754db3ff6cf1f.jpg
inflating: craters/train/images/mars_crater--59-_jpg.rf.4f609e19d4f7f9987f696183edd87e99.jpg
inflating: craters/train/images/mars_crater--6-_jpg.rf.ebff635e74e807f0377c656009482d1c.jpg
inflating: craters/train/images/mars_crater--61-_jpg.rf.1398b749af6c18cf177cc97396129bd0.jpg
inflating: craters/train/images/mars_crater--62-_jpg.rf.18dd7779b62fb78d5f07060fec32079d.jpg
inflating: craters/train/images/mars_crater--68-_jpg.rf.e503921bd751aed3cfd662cda03ab6b6.jpg
inflating: craters/train/images/mars_crater--69-_jpg.rf.8677e1c58b60bd19c197032d3e749cda.jpg
inflating: craters/train/images/mars_crater--7-_jpg.rf.84b2a78c125aa0230712626eb381a2b4.jpg
inflating: craters/train/images/mars_crater--70-_jpg.rf.30c918cda3fcb18a9ff9ce047e0a93a0.jpg
inflating: craters/train/images/mars_crater--71-_jpg.rf.3370d6bdc02cbe6643f7509228f2ee10.jpg
inflating: craters/train/images/mars_crater--74-_jpg.rf.259a2b0955c98b2b468d63917ae13796.jpg
inflating: craters/train/images/mars_crater--75-_jpg.rf.210df1cf8ebc1b962b63621720ecf926.jpg
inflating: craters/train/images/mars_crater--76-_jpg.rf.050c14a337ce70de3519fe6049640831.jpg
inflating: craters/train/images/mars_crater--77-_jpg.rf.64732a16676a6c3222f67571cf1d6618.jpg
inflating: craters/train/images/mars_crater--78-_jpg.rf.d960fe84ae55c6b91be086d8e8a8ed32.jpg
inflating: craters/train/images/mars_crater--8-_jpg.rf.85af0fa193ad2d5b0087dc48ec7341c6.jpg
inflating: craters/train/images/mars_crater--80-_jpg.rf.55cff0d5b6e2587b15ba97e5f6a3a5f5.jpg
inflating: craters/train/images/mars_crater--81-_jpg.rf.147703ba93a36a6a753992752f0a1789.jpg
inflating: craters/train/images/mars_crater--82-_jpg.rf.c3ba3759df63259b6ba3df175475d3ce.jpg
inflating: craters/train/images/mars_crater--83-_jpg.rf.3a468d2f7715d1cf73d5a31d385e0c34.jpg
inflating: craters/train/images/mars_crater--84-_jpg.rf.dcfe6c285ac885441a1c95d818b188e1.jpg
inflating: craters/train/images/mars_crater--85-_jpg.rf.344998b46a84deb2b8802cdae98c5282.jpg
inflating: craters/train/images/mars_crater--86-_jpg.rf.81699898067809e7560109fb72a56670.jpg
inflating: craters/train/images/mars_crater--87-_jpg.rf.649d603d5617ce66e4a9420539fbfebb.jpg
inflating: craters/train/images/mars_crater--88-_jpg.rf.2f3b851dec48a94bfccc6519b4a45859.jpg
inflating: craters/train/images/mars_crater--9-_jpg.rf.64378015a647ef92689e24ea103644cd.jpg
inflating: craters/train/images/mars_crater--93-_jpg.rf.7426ac666cec74a8b0bc15d550b8699b.jpg
inflating: craters/train/images/mars_crater--94-_jpg.rf.a1d59711147871b0498b03c2042c7b56.jpg
inflating: craters/train/images/mars_crater--96-_jpg.rf.6dca8e5264f81785f8d718f92f9e4475.jpg
inflating: craters/train/images/mars_crater--99-_jpg.rf.09682f4dbe2f52ffe890061454306153.jpg
inflating: craters/train/labels.cache
inflating: craters/train/labels/011_png.rf.8ac312b4898f0106d10b76952a55d237.txt
inflating: craters/train/labels/012_png.rf.64da6ff4c62638096ee6e5bf689706bc.txt
inflating: craters/train/labels/013_png.rf.ee44d5aa33fd33a1ed62ae233180f505.txt
inflating: craters/train/labels/016_png.rf.1973f9540ae7f672257609a8e5721ab3.txt
inflating: craters/train/labels/017_png.rf.1504c0d3ecbf20af6bc5114ca197a0dd.txt
inflating: craters/train/labels/018_png.rf.2d4eed5581681fe83830e51634befdaf.txt
inflating: craters/train/labels/01_png.rf.4d2ebc5ed98ad1e69d667aadbce63d53.txt
inflating: craters/train/labels/020_png.rf.ce87f4889d7441275135633392f98ed7.txt
inflating: craters/train/labels/022_png.rf.00ae4e655a2774bada1e254641482935.txt
inflating: craters/train/labels/02_png.rf.610687947e4c92f77e6462104ec4b924.txt
inflating: craters/train/labels/03_png.rf.8f7b31e14642026833b7c0dcd1832862.txt
inflating: craters/train/labels/05_png.rf.844343145246e51e66a345419e1862bf.txt
inflating: craters/train/labels/06_png.rf.aaf8c66b9e4d5e99a3dc70bae7f62c07.txt
inflating: craters/train/labels/07_png.rf.3fba5e95c6827aa6c4132e70f2086555.txt
inflating: craters/train/labels/08_png.rf.944efdd0f108140b368d7ad2c37426df.txt
inflating: craters/train/labels/09_png.rf.3b796e77a5f0036af4cd4413fcbe07a5.txt
inflating: craters/train/labels/mars_crater--0-_jpg.rf.40c1dec94c66ab07d9da8c74fa58d6f8.txt
inflating: craters/train/labels/mars_crater--10-_jpg.rf.585b1aa305997e3055e86fcac72a806b.txt
inflating: craters/train/labels/mars_crater--101-_jpg.rf.8f4eb1c77ab9e64d2fd691a6e0fcd3ec.txt
inflating: craters/train/labels/mars_crater--102-_jpg.rf.9cdbcc724e2e7baf5c5c17aed91c769b.txt
inflating: craters/train/labels/mars_crater--103-_jpg.rf.61597458910fb0eed55b415c9cbac3bf.txt
inflating: craters/train/labels/mars_crater--105-_jpg.rf.338dccc756e50430460dfb6f6191d1c5.txt
inflating: craters/train/labels/mars_crater--106-_jpg.rf.05aa3dbbbdc9651cb8c51d26c2847805.txt
inflating: craters/train/labels/mars_crater--109-_jpg.rf.29afc1137114a2b872f484326254e949.txt
inflating: craters/train/labels/mars_crater--11-_jpg.rf.9e8de286e3d84938f34bb6bf35825343.txt
inflating: craters/train/labels/mars_crater--110-_jpg.rf.593f6a3d9aed98e7a08955e700765222.txt
inflating: craters/train/labels/mars_crater--111-_jpg.rf.215f88d6bd7a6d86560754be694fc2f3.txt
inflating: craters/train/labels/mars_crater--113-_jpg.rf.a7713d55de03a1a7487429d414e1005c.txt
inflating: craters/train/labels/mars_crater--118-_jpg.rf.0a8b3fb0e1332e576901e596ad55e30a.txt
inflating: craters/train/labels/mars_crater--119-_jpg.rf.5cd500a3dcc977038a875b5115f9eda4.txt
inflating: craters/train/labels/mars_crater--120-_jpg.rf.a870131b0b5974e8351a32ade2d5f0f3.txt
inflating: craters/train/labels/mars_crater--14-_jpg.rf.e050bff73578e2a9ad30e339b898f839.txt
inflating: craters/train/labels/mars_crater--15-_jpg.rf.c669035a909b835be84fa4fc67bf3f40.txt
inflating: craters/train/labels/mars_crater--16-_jpg.rf.bf5f6796b4ec5d351b47efae2e866934.txt
inflating: craters/train/labels/mars_crater--17-_jpg.rf.c051d4c804f2fed2aa32999cc9fd0b48.txt
inflating: craters/train/labels/mars_crater--18-_jpg.rf.8214037c811619bbe37024a557c9251f.txt
inflating: craters/train/labels/mars_crater--19-_jpg.rf.a9e7485a270816a553c3d7e0c3fddd30.txt
inflating: craters/train/labels/mars_crater--2-_jpg.rf.a65bb7ee68968509ac0b28df75e1f251.txt
inflating: craters/train/labels/mars_crater--20-_jpg.rf.cb4c2603f4a62bbe96328c29cec6d68e.txt
inflating: craters/train/labels/mars_crater--21-_jpg.rf.63688e718106a9b061bb702a298e7a54.txt
inflating: craters/train/labels/mars_crater--22-_jpg.rf.3ed23daa3923c22a459593cb449a7336.txt
inflating: craters/train/labels/mars_crater--24-_jpg.rf.4614bb6844fda70ea8a3b4f515cb7e8f.txt
```

```
inflating: craters/train/labels/mars_crater--26-_jpg.rf.e34074ddcd6c44fa17b7a7689f7ddfa4.txt
inflating: craters/train/labels/mars_crater--27-_jpg.rf.158fc0c6a694bba2f0afa44fd76a7f5a.txt
inflating: craters/train/labels/mars_crater--3-_jpg.rf.17bb27df5c5e6a501f790342675bc145.txt
inflating: craters/train/labels/mars_crater--30-_jpg.rf.0fb68bf5aece1859bc5f080e66b4fef1.txt
inflating: craters/train/labels/mars_crater--34-_jpg.rf.5617051b3d6a801330f8d4f7df9e3239.txt
inflating: craters/train/labels/mars_crater--35-_jpg.rf.826bc4b2a3a69c48144834f40d92439f.txt
inflating: craters/train/labels/mars_crater--36-_jpg.rf.0100a780ad4217b29c2dc9b46deec040.txt
inflating: craters/train/labels/mars_crater--37-_jpg.rf.c2a35d9de333416eb4d6d8f0d6b3310a.txt
inflating: craters/train/labels/mars_crater--38-_jpg.rf.e6f3636efb075ba61dafa478e6b21de2.txt
inflating: craters/train/labels/mars_crater--39-_jpg.rf.dbcaca888781a42f7bb3ce2c922da228.txt
inflating: craters/train/labels/mars_crater--4-_jpg.rf.b31f215c27d917bdb7c7c545db2bc36b.txt
inflating: craters/train/labels/mars_crater--41-_jpg.rf.5b367c681ccf0dfa9c5fddf287926d83.txt
inflating: craters/train/labels/mars_crater--42-_jpg.rf.0cea1f9d9cf45fb141abf4e9aaae4c84.txt
inflating: craters/train/labels/mars_crater--43-_jpg.rf.27ea7a65603205e491bc439c1d654cf4.txt
inflating: craters/train/labels/mars_crater--44-_jpg.rf.627a604f51a4479d1f1660c5cf2af5c4.txt
inflating: craters/train/labels/mars_crater--45-_jpg.rf.d69fa5e159ad8e5fe17b96deaba99c18.txt
inflating: craters/train/labels/mars_crater--48-_jpg.rf.cc5528a8dcf4a11085a0a01d5c9084bc.txt
inflating: craters/train/labels/mars_crater--49-_jpg.rf.5ea4703986c80bcfa03f0f99e4067c6f.txt
inflating: craters/train/labels/mars_crater--5-_jpg.rf.6acb650edc64d6e1a02a0eef00e95113.txt
inflating: craters/train/labels/mars_crater--50-_jpg.rf.02be16d199897982df8e6ea8f3e5818f.txt
inflating: craters/train/labels/mars_crater--52-_jpg.rf.1a227f21c518775052f847053022cd86.txt
inflating: craters/train/labels/mars_crater--53-_jpg.rf.20acc76fad4fecd456d4b0e77c9bbbb1.txt
inflating: craters/train/labels/mars_crater--54-_jpg.rf.2313c9f439999313bfb527534bbdc501.txt
inflating: craters/train/labels/mars_crater--55-_jpg.rf.452ea46e23156b2c86433eb9054b1a6a.txt
inflating: craters/train/labels/mars_crater--56-_jpg.rf.6f0dcbff4ea66014aef178b840d29238.txt
inflating: craters/train/labels/mars_crater--57-_jpg.rf.a783aaeaf00ae5d57bec64438050285f.txt
inflating: craters/train/labels/mars_crater--58-_jpg.rf.d668b2f63bd183cb629754db3ff6cf1f.txt
inflating: craters/train/labels/mars_crater--59-_jpg.rf.4f609e19d4f7f9987f696183edd87e99.txt
inflating: craters/train/labels/mars_crater--6-_jpg.rf.ebff635e74e807f0377c656009482d1c.txt
inflating: craters/train/labels/mars_crater--61-_jpg.rf.1398b749af6c18cf177cc97396129bd0.txt
inflating: craters/train/labels/mars_crater--62-_jpg.rf.18dd7779b62fb78d5f07060fec32079d.txt
inflating: craters/train/labels/mars_crater--68-_jpg.rf.e503921bd751aed3cfd662cda03ab6b6.txt
inflating: craters/train/labels/mars_crater--69-_jpg.rf.8677e1c58b60bd19c197032d3e749cda.txt
inflating: craters/train/labels/mars_crater--7-_jpg.rf.84b2a78c125aa0230712626eb381a2b4.txt
inflating: craters/train/labels/mars_crater--70-_jpg.rf.30c918cda3fcb18a9ff9ce047e0a93a0.txt
inflating: craters/train/labels/mars_crater--71-_jpg.rf.3370d6bdc02cbe6643f7509228f2ee10.txt
inflating: craters/train/labels/mars_crater--74-_jpg.rf.259a2b0955c98b2b468d63917ae13796.txt
inflating: craters/train/labels/mars_crater--75-_jpg.rf.210df1cf8ebc1b962b63621720ecf926.txt
inflating: craters/train/labels/mars_crater--76-_jpg.rf.050c14a337ce70de3519fe6049640831.txt
inflating: craters/train/labels/mars_crater--77-_jpg.rf.64732a16676a6c3222f67571cf1d6618.txt
inflating: craters/train/labels/mars_crater--78-_jpg.rf.d960fe84ae55c6b91be086d8e8a8ed32.txt
inflating: craters/train/labels/mars_crater--8-_jpg.rf.85af0fa193ad2d5b0087dc48ec7341c6.txt
inflating: craters/train/labels/mars_crater--80-_jpg.rf.55cff0d5b6e2587b15ba97e5f6a3a5f5.txt
inflating: craters/train/labels/mars_crater--81-_jpg.rf.147703ba93a36a6a753992752f0a1789.txt
inflating: craters/train/labels/mars_crater--82-_jpg.rf.c3ba3759df63259b6ba3df175475d3ce.txt
inflating: craters/train/labels/mars_crater--83-_jpg.rf.3a468d2f7715d1cf73d5a31d385e0c34.txt
inflating: craters/train/labels/mars_crater--84-_jpg.rf.dcfe6c285ac885441a1c95d818b188e1.txt
inflating: craters/train/labels/mars_crater--85-_jpg.rf.344998b46a84deb2b8802cdae98c5282.txt
inflating: craters/train/labels/mars_crater--86-_jpg.rf.81699898067809e7560109fb72a56670.txt
inflating: craters/train/labels/mars_crater--87-_jpg.rf.649d603d5617ce66e4a9420539fbfebb.txt
inflating: craters/train/labels/mars_crater--88-_jpg.rf.2f3b851dec48a94bfccc6519b4a45859.txt
inflating: craters/train/labels/mars_crater--9-_jpg.rf.64378015a647ef92689e24ea103644cd.txt
inflating: craters/train/labels/mars_crater--93-_jpg.rf.7426ac666cec74a8b0bc15d550b8699b.txt
inflating: craters/train/labels/mars_crater--94-_jpg.rf.a1d59711147871b0498b03c2042c7b56.txt
inflating: craters/train/labels/mars_crater--96-_jpg.rf.6dca8e5264f81785f8d718f92f9e4475.txt
inflating: craters/train/labels/mars_crater--99-_jpg.rf.09682f4dbe2f52ffe890061454306153.txt
inflating: craters/valid/images/014_png.rf.6b86c8afeb3ebca92872fd966d076adc.jpg
inflating: craters/valid/images/021_png.rf.575e0f1cb87a2cc439172de15816c2cc.jpg
inflating: craters/valid/images/023_png.rf.0a3106bdcc8e64bba283f961db77c9ec.jpg
inflating: craters/valid/images/024_png.rf.be24000681f4f356e2a027fb7ca0c552.jpg
inflating: craters/valid/images/mars_crater--1-_png.rf.f1b399f95708fd0ce520f23c9944ed44.jpg
inflating: craters/valid/images/mars_crater--104-_jpg.rf.2ff6036b976ac6bbba612dd48f144d4d.jpg
inflating: craters/valid/images/mars_crater--107-_jpg.rf.eb0f7061ed8c8af8be0e8af6b97226b0.jpg
inflating: craters/valid/images/mars_crater--115-_jpg.rf.8b1bf8c37395c894e88e543ccb4cc42f.jpg
inflating: craters/valid/images/mars_crater--13-_jpg.rf.f6fc01f728dfec8b32ad59d696a1a08d.jpg
inflating: craters/valid/images/mars_crater--2-_png.rf.08b565c861e9ac03e3cd1f9b0a6cddbe.jpg
inflating: craters/valid/images/mars_crater--23-_jpg.rf.ab2b0f47006c0c17260c312215dcfb88.jpg
inflating: craters/valid/images/mars_crater--28-_jpg.rf.8a4c9991d6739123bd0cbf018ea6f50d.jpg
inflating: craters/valid/images/mars_crater--29-_jpg.rf.719ff6d6c1aafd71147c173f7d224018.jpg
inflating: craters/valid/images/mars_crater--40-_jpg.rf.29e123c14a95b2e0fcdc93e9e8cff902.jpg
inflating: craters/valid/images/mars_crater--47-_jpg.rf.63eed086b1d39dc52405a398229dc586.jpg
inflating: craters/valid/images/mars_crater--63-_jpg.rf.ab0440022651f886c68744a55eb215d9.jpg
inflating: craters/valid/images/mars_crater--64-_jpg.rf.fd0838698f86dd546cdfcfea987b98b2.jpg
inflating: craters/valid/images/mars_crater--67-_jpg.rf.cf856f9fde7eb10f7f1524ebb0024502.jpg
inflating: craters/valid/images/mars_crater--72-_jpg.rf.4e01fbde7746b57082310e11b1ff03e2.jpg
inflating: craters/valid/images/mars_crater--73-_jpg.rf.91d7dc2cc9ae2d5dee41207d8992c3b3.jpg
inflating: craters/valid/images/mars_crater--79-_jpg.rf.984de20cb547cf4ee0324ebbf3fb7a88.jpg
inflating: craters/valid/images/mars_crater--89-_jpg.rf.a57ed45379b066a537e1203ac9827a27.jpg
inflating: craters/valid/images/mars_crater--91-_jpg.rf.9203caee622cf671580b91536e36e7e2.jpg
inflating: craters/valid/images/mars_crater--92-_jpg.rf.608e45488b76786c69755a901c1b73c9.jpg
inflating: craters/valid/images/mars_crater--95-_jpg.rf.671f7041e41a37f7faec12f3e9eb1d7d.jpg
inflating: craters/valid/images/mars_crater--98-_jpg.rf.441572398e83f5d7769a8e57248c9248.jpg
inflating: craters/valid/labels.cache
inflating: craters/valid/labels/014_png.rf.6b86c8afeb3ebca92872fd966d076adc.txt
inflating: craters/valid/labels/021_png.rf.575e0f1cb87a2cc439172de15816c2cc.txt
inflating: craters/valid/labels/023_png.rf.0a3106bdcc8e64bba283f961db77c9ec.txt
inflating: craters/valid/labels/024_png.rf.be24000681f4f356e2a027fb7ca0c552.txt
inflating: craters/valid/labels/mars_crater--1-_png.rf.f1b399f95708fd0ce520f23c9944ed44.txt
inflating: craters/valid/labels/mars_crater--104-_jpg.rf.2ff6036b976ac6bbba612dd48f144d4d.txt
```

```
  inflating: craters/valid/labels/mars_crater--107-_jpg.rf.eb0f7061ed8c8af8be0e8af6b97226b0.txt
  inflating: craters/valid/labels/mars_crater--115-_jpg.rf.8b1bf8c37395c894e88e543ccb4cc42f.txt
  inflating: craters/valid/labels/mars_crater--13-_jpg.rf.f6fc01f728dfec8b32ad59d696a1a08d.txt
  inflating: craters/valid/labels/mars_crater--2-_png.rf.08b565c861e9ac03e3cd1f9b0a6cddbe.txt
  inflating: craters/valid/labels/mars_crater--23-_jpg.rf.ab2b0f47006c0c17260c312215dcfb88.txt
  inflating: craters/valid/labels/mars_crater--28-_jpg.rf.8a4c9991d6739123bd0cbf018ea6f50d.txt
  inflating: craters/valid/labels/mars_crater--29-_jpg.rf.719ff6d6c1aafd71147c173f7d224018.txt
  inflating: craters/valid/labels/mars_crater--40-_jpg.rf.29e123c14a95b2e0fcdc93e9e8cff902.txt
  inflating: craters/valid/labels/mars_crater--47-_jpg.rf.63eed086b1d39dc52405a398229dc586.txt
  inflating: craters/valid/labels/mars_crater--63-_jpg.rf.ab0440022651f886c68744a55eb215d9.txt
  inflating: craters/valid/labels/mars_crater--64-_jpg.rf.fd0838698f86dd546cdfcfea987b98b2.txt
  inflating: craters/valid/labels/mars_crater--67-_jpg.rf.cf856f9fde7eb10f7f1524ebb0024502.txt
  inflating: craters/valid/labels/mars_crater--72-_jpg.rf.4e01fbde7746b57082310e11b1ff03e2.txt
  inflating: craters/valid/labels/mars_crater--73-_jpg.rf.91d7dc2cc9ae2d5dee41207d8992c3b3.txt
  inflating: craters/valid/labels/mars_crater--79-_jpg.rf.984de20cb547cf4ee0324ebbf3fb7a88.txt
  inflating: craters/valid/labels/mars_crater--89-_jpg.rf.a57ed45379b066a537e1203ac9827a27.txt
  inflating: craters/valid/labels/mars_crater--91-_jpg.rf.9203caee622cf671580b91536e36e7e2.txt
  inflating: craters/valid/labels/mars_crater--92-_jpg.rf.608e45488b76786c69755a901c1b73c9.txt
  inflating: craters/valid/labels/mars_crater--95-_jpg.rf.671f7041e41a37f7faec12f3e9eb1d7d.txt
  inflating: craters/valid/labels/mars_crater--98-_jpg.rf.441572398e83f5d7769a8e57248c9248.txt
Cloning into 'vision'...
remote: Enumerating objects: 424634, done.
remote: Counting objects: 100% (5096/5096), done.
remote: Compressing objects: 100% (324/324), done.
remote: Total 424634 (delta 4789), reused 5049 (delta 4764), pack-reused 419538
Receiving objects: 100% (424634/424634), 836.56 MiB | 24.98 MiB/s, done.
Resolving deltas: 100% (393869/393869), done.
fatal: not a git repository (or any of the parent directories): .git
Requirement already satisfied: cython in /usr/local/lib/python3.10/dist-packages (3.0.5)
Collecting git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI
  Cloning https://github.com/cocodataset/cocoapi.git to /tmp/pip-req-build-_4tg920u
  Running command git clone --filter=blob:none --quiet https://github.com/cocodataset/cocoapi.git /tmp/pip-req-
build-_4tg920u
  Resolved https://github.com/cocodataset/cocoapi.git to commit 8c9bcc3cf640524c4c20a9c40e89cb6a2f2fa0e9
  Preparing metadata (setup.py) ... done
Requirement already satisfied: setuptools>=18.0 in /usr/local/lib/python3.10/dist-packages (from pycocotools==2
.0) (67.7.2)
Requirement already satisfied: cython>=0.27.3 in /usr/local/lib/python3.10/dist-packages (from pycocotools==2.0
) (3.0.5)
Requirement already satisfied: matplotlib>=2.1.0 in /usr/local/lib/python3.10/dist-packages (from pycocotools==
2.0) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.
1.0->pycocotools==2.0) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0-
>pycocotools==2.0) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2
.1.0->pycocotools==2.0) (4.44.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2
.1.0->pycocotools==2.0) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->
pycocotools==2.0) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1
.0->pycocotools==2.0) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0
->pycocotools==2.0) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.
1.0->pycocotools==2.0) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib
>=2.1.0->pycocotools==2.0) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->
matplotlib>=2.1.0->pycocotools==2.0) (1.16.0)
Building wheels for collected packages: pycocotools
  Building wheel for pycocotools (setup.py) ... done
  Created wheel for pycocotools: filename=pycocotools-2.0-cp310-cp310-linux_x86_64.whl size=375458 sha256=b4b61
3bd77a62c977e020c856383838707512fcce512b08bc53092aa2513e5b1
  Stored in directory: /tmp/pip-ephem-wheel-cache-zpxfaoy9/wheels/39/61/b4/480fbddb4d3d6bc34083e7397bc6f5d1381f
79acc68e9f3511
Successfully built pycocotools
Installing collected packages: pycocotools
  Attempting uninstall: pycocotools
    Found existing installation: pycocotools 2.0.7
    Uninstalling pycocotools-2.0.7:
      Successfully uninstalled pycocotools-2.0.7
Successfully installed pycocotools-2.0
Requirement already satisfied: albumentations in /usr/local/lib/python3.10/dist-packages (1.3.1)
Requirement already satisfied: numpy>=1.11.1 in /usr/local/lib/python3.10/dist-packages (from albumentations) (
1.23.5)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from albumentations) (1
.11.3)
Requirement already satisfied: scikit-image>=0.16.1 in /usr/local/lib/python3.10/dist-packages (from albumentat
ions) (0.19.3)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from albumentations) (6.0.1)
Requirement already satisfied: qudida>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from albumentations) (
0.0.4)
Requirement already satisfied: opencv-python-headless>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from a
lbumentations) (4.8.1.78)
Requirement already satisfied: scikit-learn>=0.19.1 in /usr/local/lib/python3.10/dist-packages (from qudida>=0.
0.4->albumentations) (1.2.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from qudida>=0.0.4
```

```
->albumentations) (4.5.0)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.1
6.1->albumentations) (3.2.1)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/local/lib/python3.10/dist-packages
(from scikit-image>=0.16.1->albumentations) (9.4.0)
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0.
16.1->albumentations) (2.31.6)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-packages (from scikit-imag
e>=0.16.1->albumentations) (2023.9.26)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-image>
=0.16.1->albumentations) (1.4.1)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from scikit-image>=0
.16.1->albumentations) (23.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.1
9.1->qudida>=0.0.4->albumentations) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lea
rn>=0.19.1->qudida>=0.0.4->albumentations) (3.2.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.8.0.76)
Collecting opencv-python
  Downloading opencv_python-4.8.1.78-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (61.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.7/61.7 MB 10.3 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1
.23.5)
Installing collected packages: opencv-python
  Attempting uninstall: opencv-python
    Found existing installation: opencv-python 4.8.0.76
    Uninstalling opencv-python-4.8.0.76:
      Successfully uninstalled opencv-python-4.8.0.76
Successfully installed opencv-python-4.8.1.78
cp: cannot stat '/kaggle/input/martianlunar-crater-detection-dataset/craters/train/images/.': No such file or d
irectory
cp: cannot stat '/kaggle/input/martianlunar-crater-detection-dataset/craters/valid/images/.': No such file or d
irectory
cp: cannot stat '/kaggle/input/martianlunar-crater-detection-dataset/craters/train/labels/.': No such file or d
irectory
cp: cannot stat '/kaggle/input/martianlunar-crater-detection-dataset/craters/valid/labels/.': No such file or d
irectory
```

Este código realiza una serie de tareas con el objetivo de descargar archivos y repositorios específicos, instalar paquetes necesarios, y copiar y unificar conjuntos de datos en un formato determinado.

1. Descarga de recursos y dependencias

- Se instala el paquete `gdown` para descargar archivos desde Google Drive
- Descarga un archivo llamado **mars_and_moon.zip** desde Google Drive y descomprime su contenido.
- Se clona el repositorio de TorchVision de GitHub (`https://github.com/pytorch/vision.git`).
- Se accede al directorio del repositorio clonado.
- Se cambia a una versión específica del repositorio (`v0.8.2`).

2. Copia de archivos necesarios

- Se copian archivos importantes desde el repositorio de TorchVision a tu directorio actual. Estos archivos son esenciales para la detección de objetos

3. Instalación de bibliotecas adicionales

- Se instala el paquete `cython` .
- Se instala una versión específica de la biblioteca `pycocotools` desde su repositorio en GitHub. Esta biblioteca es necesaria para trabajar con el conjunto de datos COCO (Common Objects in Context)..
- Se instala una versión específica de `albumentations` (se suele usar para realizar aumentos de datos en las imágenes).
- Se instala una versión específica de `opencv-python` .

4. Copia y unificación de conjuntos de datos

- Se crea un directorio llamado `train` .
- Se copian las imágenes de los conjuntos de entrenamiento y validación a la carpeta train/images.
- Se copian las etiquetas (labels) correspondientes a las imágenes a la carpeta train/labels.

In [ ]:
```python
import os
import numpy as np
import torch
import torchvision
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
from engine import train_one_epoch, evaluate
import utils
import transforms as T
import albumentations as A
import cv2
import time
from albumentations.pytorch.transforms import ToTensorV2
import matplotlib
import matplotlib.pyplot as plt
```

```
import matplotlib.patches as patches
from  sklearn.model_selection import KFold
import random
```

Este código importa varias bibliotecas y módulos en Python, incluidos os, numpy, torch, torchvision, albumentations, cv2, time, matplotlib, y random. Además, el código importa varios módulos específicos de los paquetes importados, como FastRCNNPredictor de torchvision.models.detection.faster_rcnn (importa el modelo Faster R-CNN de TorchVision para su uso en la detección de objetos) y otrasfunciones personalizadas del módulo engine, transforms y utils.

In [ ]:
```
class CraterDataset(object):
    def __init__(self, root, transforms):
        self.root = root
        self.transforms = transforms
        # load all image files, sorting them to
        # ensure that they are aligned
        self.imgs = list(sorted(os.listdir(os.path.join(self.root, "images"))))
        self.annots = list(sorted(os.listdir(os.path.join(self.root, "labels"))))
        self.classes = ['Background','Crater']

    # Converts boundry box formats, this version assumes single class only!
    def convert_box_cord(self,bboxs, format_from, format_to, img_shape):
        if format_from == 'normxywh':
            if format_to == 'xyminmax':
                xw = bboxs[:, (1, 3)] * img_shape[1]
                yh = bboxs[:, (2, 4)] * img_shape[0]
                xmin = xw[:, 0] - xw[:, 1] / 2
                xmax = xw[:, 0] + xw[:, 1] / 2
                ymin = yh[:, 0] - yh[:, 1] / 2
                ymax = yh[:, 0] + yh[:, 1] / 2
                coords_converted = np.column_stack((xmin, ymin, xmax, ymax))

        return coords_converted

    def __getitem__(self, idx):
        # load images and boxes
        img_path = os.path.join(self.root, "images", self.imgs[idx])
        annot_path = os.path.join(self.root, "labels", self.annots[idx])
        img = cv2.imread(img_path)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB).astype(np.float32)
        img= img/255.0

        # retrieve bbox list and format to required type,
        # if annotation file is empty, fill dummy box with label 0
        if os.path.getsize(annot_path) != 0:
            bboxs = np.loadtxt(annot_path, ndmin=2)
            bboxs = self.convert_box_cord(bboxs, 'normxywh', 'xyminmax', img.shape)
            num_objs = len(bboxs)
            bboxs = torch.as_tensor(bboxs, dtype=torch.float32)
            # there is only one class
            labels = torch.ones((num_objs,), dtype=torch.int64)
            # suppose all instances are not crowd
            iscrowd = torch.zeros((num_objs,), dtype=torch.int64)
        else:
            bboxs = torch.as_tensor([[0, 0, 640, 640]], dtype=torch.float32)
            labels = torch.zeros((1,), dtype=torch.int64)
            iscrowd = torch.zeros((1,), dtype=torch.int64)

        area = (bboxs[:, 3] - bboxs[:, 1]) * (bboxs[:, 2] - bboxs[:, 0])
        image_id = torch.tensor([idx])

        target = {}
        target["boxes"] = bboxs
        target["labels"] = labels
        target["image_id"] = image_id
        target["area"] = area
        target["iscrowd"] = iscrowd

        if self.transforms is not None:
            sample = self.transforms(image=img,
                                     bboxes=target['boxes'],
                                     labels=labels)
        img = sample['image']
        target['boxes'] = torch.tensor(sample['bboxes'])
        target['labels'] = torch.tensor(sample['labels'])
        if target['boxes'].ndim == 1:
            target['boxes'] = torch.as_tensor([[0, 0, 640, 640]], dtype=torch.float32)
            target['labels'] = torch.zeros((1,), dtype=torch.int64)
        return img, target

    def __len__(self):
        return len(self.imgs)
```

Este código define una clase llamada CraterDataset que se utiliza para cargar y manejar un conjunto de datos de detección de cráteres. La clase se utiliza típicamente como un generador de datos personalizado en el contexto de entrenamiento de un modelo de detección de objetos. Las funciones y métodos que se definen en el código son los siguientes:

init: Este método inicializa la instancia de la clase CraterDataset. Toma como parámetros root que representa la ruta de la carpeta raíz del conjunto de datos y transforms que representa las transformaciones a aplicar en las imágenes y las cajas delimitadoras (etiquetas).

convert_box_cord: Este método convierte las coordenadas de las cajas delimitadoras de un formato a otro, como de 'nor(malizado)mxywh' a 'xyminmax'. Las coordenadas normalizadas se refieren a las coordenadas xy del centro de la caja y su ancho y alto en relación con las dimensiones de la imagen. En resumen, devuelve las coordenadas de las cajas en el formato deseado. Se asume que solo hay una clase en este caso.

getitem: Este método se utiliza para obtener un elemento específico del conjunto de datos en función de su índice idx. Carga la imagen y las anotaciones correspondientes. Carga la imagen, realiza conversiones necesarias y normaliza sus valores. Lee las coordenadas de las cajas de anotación desde el archivo y las convierte al formato requerido. Prepara etiquetas y otros datos, como el área y la identificación de la imagen. Aplica transformaciones si se han proporcionado a la imagen y las etiquetas. Devuelbe la imagen y las etiquetas pricesadas

len: Este método devuelve la longitud total del conjunto de datos.

In [ ]:
```python
def get_model_bbox(num_classes):
    # load an instance segmentation model pre-trained on COCO
    model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)

    # get number of input features for the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    # replace the pre-trained head with a new one
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    return model
```

Código importante porque luego lo modificaremos para cambiar el modelo o método de detección de objetos:

La función **get_model_bbox** toma num_classes como parámetro, que representa el número de clases diferentes que el modelo debe ser capaz de detectar. El modelo preentrenado utilizado es una red neuronal convolucional basada en ResNet-50 con una cabeza de detección rápida **(Faster R-CNN)**. Este modelo ha sido preentrenado en el conjunto de datos COCO para realizar tareas de detección de objetos. Se obtiene el número de características de entrada para el clasificador del modelo. La cabeza de predicción preentrenada se reemplaza por una nueva instancia de FastRCNNPredictor con el número correcto de clases de salida.

In [ ]:
```python
def get_transform(train):
    if train:
        return A.Compose([
            # A.Flip(p=0.5),
            # A.RandomResizedCrop(height=640,width=640,p=0.4),
            # # A.Perspective(p=0.4),
            # A.Rotate(p=0.5),
            # # A.Transpose(p=0.3),
            ToTensorV2(p=1.0)],
            bbox_params=A.BboxParams(format='pascal_voc',min_visibility=0.4, label_fields=['labels']))
    else:
        return A.Compose([ToTensorV2(p=1.0)],
                         bbox_params=A.BboxParams(format='pascal_voc', min_visibility=0.5, label_fields=['label
```

La función **get_transform** devuelve una serie de transformaciones de imagen que se utilizan durante el entrenamiento y la evaluación de un modelo de detección de objetos. La función acepta un parámetro booleano train, que indica si las transformaciones deben aplicarse durante el entrenamiento o la evaluación. En este caso:

- Si train es *True* (es decir, si se está realizando el entrenamiento), se utilizan varias transformaciones de aumento de datos, como voltear, recortar, rotar y convertir la imagen en un tensor, junto con los parámetros de la caja delimitadora definidos en bbox_params.
- Si train es *False* (es decir, si se está realizando la evaluación), solo se convierte la imagen en un tensor, sin aplicar otras transformaciones de aumento de datos.

In [ ]:
```python
def reset_weights(m):
  '''
    Try resetting model weights to avoid
    weight leakage.
  '''
  for layer in m.children():
    if hasattr(layer, 'reset_parameters'):
        print(f'Reset trainable parameters of layer = {layer}')
        layer.reset_parameters()
```

La función reset_weights toma un modelo 'm' como entrada y trata de restablecer sus pesos para evitar la posible fuga de pesos o la influencia de los pesos preentrenados en un modelo. Lo que hace esta función es lo siguiente:

1. Itera sobre todas las capas del modelo utilizando **m.children()**.
2. Para cada capa, comprueba si tiene un método llamado reset_parameters.
3. Si la capa tiene el método reset_parameters, lo llama para restablecer los parámetros de la capa.

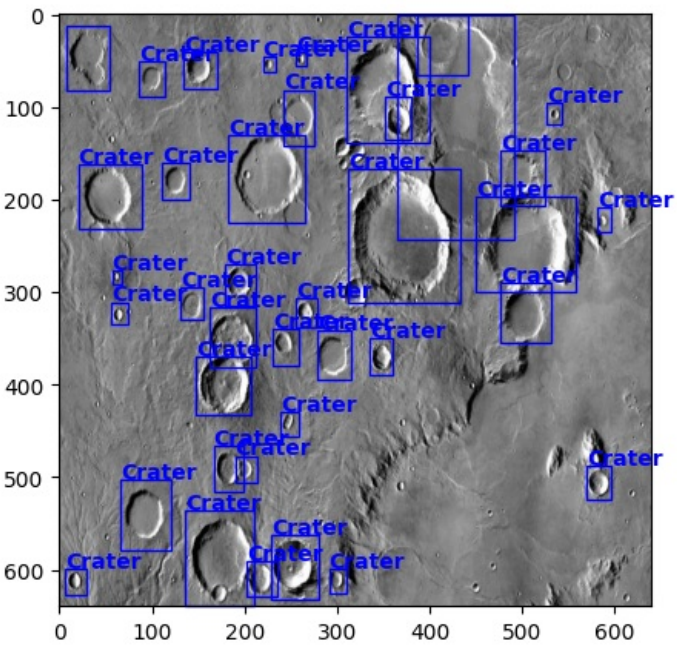
En resumen, esta función se utiliza para restablecer los pesos de las capas del modelo, lo que puede ser útil en situaciones en las que deseas comenzar un nuevo entrenamiento desde cero sin que los pesos previamente entrenados tengan un impacto en el nuevo entrenamiento. Esto es particularmente útil cuando se ajustan modelos preentrenados para una nueva tarea o cuando se quiere evitar el "derrame" de pesos entre modelos diferentes. La función imprime un mensaje indicando qué capas han tenido sus parámetros restablecidos.

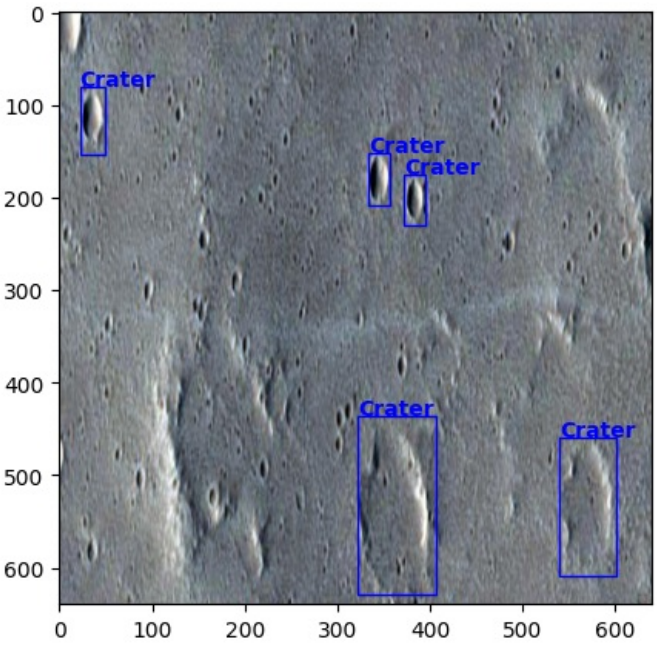
In [ ]:
```python
# Function to visualize bounding boxes in the image
def plot_img_bbox(img, target):
    # plot the image and bboxes
    # Bounding boxes are defined as follows: x-min y-min width height
    fig, a = plt.subplots(1, 1)
    fig.set_size_inches(5, 5)
    a.imshow(img.permute((1,2,0)))
    for box in (target['boxes']):
        x, y, width, height = box[0], box[1], box[2] - box[0], box[3] - box[1]
        rect = patches.Rectangle((x, y),
                                 width, height,
                                 edgecolor='b',
                                 facecolor='none',
                                 clip_on=False)
        a.annotate('Crater', (x,y-20), color='blue', weight='bold',
                   fontsize=10, ha='left', va='top')

        # Draw the bounding box on top of the image
        a.add_patch(rect)
    plt.show()
```

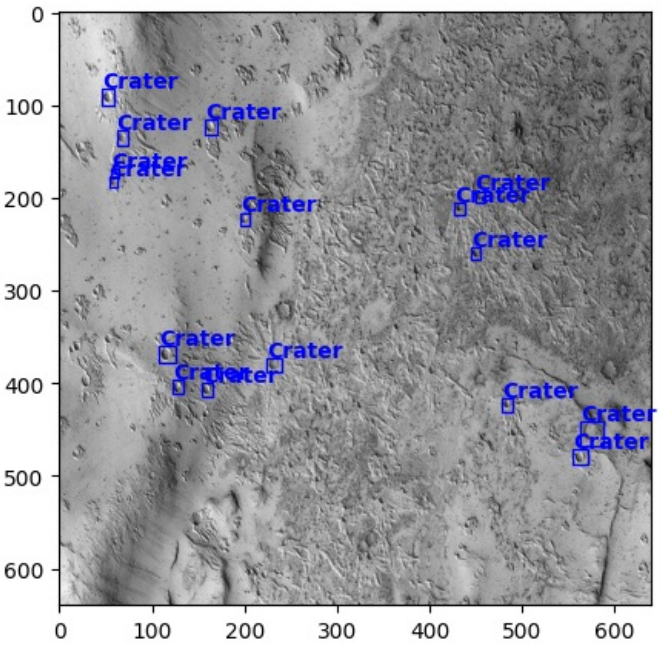
La función `plot_img_bbox` se utiliza para visualizar imágenes junto con las cajas delimitadoras correspondientes. La función sigue los siguientes pasos:

1. Toma una imagen `img` y un diccionario `target` como entrada. El diccionario `target` contiene información sobre las cajas delimitadoras de los objetos detectados en la imagen.
2. Crea una figura y un eje utilizando `plt.subplots(1, 1)` y `fig, a`.
3. Ajusta el tamaño de la figura a 5 pulgadas por 5 pulgadas con `fig.set_size_inches(5, 5)`.
4. Muestra la imagen en el eje con `a.imshow(img.permute((1,2,0)))`.
5. Itera a través de las cajas delimitadoras en `target['boxes']` y dibuja rectángulos correspondientes en la imagen con `patches.Rectangle`. La función también agrega anotaciones de texto indicando la etiqueta "Crater" sobre cada caja delimitadora.
6. Muestra la imagen con las cajas delimitadoras dibujadas utilizando `plt.show()`.

In [ ]:
```python
dataset = CraterDataset('/content/craters/train', get_transform(train=True))
# Prints an example of image with annotations
for i in random.sample(range(1, 100), 3):
    img, target = dataset[i]
    plot_img_bbox(img, target)
```







Este fragmento de código utiliza la clase `CraterDataset` previamente definida para cargar un conjunto de datos de detección de cráteres desde la carpeta '/content/craters/train' y aplica transformaciones de datos utilizando la función `get_transform` con el parámetro `train=True`.

Posteriormente, el código selecciona aleatoriamente tres muestras del conjunto de datos y visualiza cada una de ellas utilizando la función `plot_img_bbox`, que muestra la imagen junto con las cajas delimitadoras correspondientes. El objetivo de este fragmento de código es mostrar ejemplos de imágenes con anotaciones de cráteres para ayudar en la verificación y la comprensión visual del conjunto de datos de detección de cráteres.

In [ ]:
```python
# train on the GPU or on the CPU, if a GPU is not available
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
k_folds = 7
num_epochs = 5


# our dataset has two classes only - background and crater
num_classes = 2
# use our dataset and defined transformations
dataset = CraterDataset('/content/craters/train', get_transform(train=True))
dataset_val = CraterDataset('/content/craters/train', get_transform(train=False))
```

```python
# Define the K-fold Cross Validator
kfold = KFold(n_splits=k_folds, shuffle=True)

# Start print
print('--------------------------------')

# K-fold Cross Validation model evaluation
for fold, (train_ids, val_ids) in enumerate(kfold.split(dataset)):
    print(f'FOLD {fold}')
    print('--------------------------------')

    dataset_subset = torch.utils.data.Subset(dataset, list(train_ids))
    dataset_val_subset = torch.utils.data.Subset(dataset_val, list(val_ids))

    # define training and validation data loaders
    data_loader = torch.utils.data.DataLoader(
            dataset_subset, batch_size=8, shuffle=True, num_workers=2,
        collate_fn=utils.collate_fn)

    data_loader_val = torch.utils.data.DataLoader(
        dataset_val_subset, batch_size=1, shuffle=False, num_workers=2,
        collate_fn=utils.collate_fn)

    # get the model using our helper function
    model = get_model_bbox(num_classes)

    #model.apply(reset_weights) # Check if beneficial

    # move model to the right device
    model.to(device)

    # construct an optimizer
    params = [p for p in model.parameters() if p.requires_grad]
    optimizer = torch.optim.SGD(params, lr=0.005,  # Check if beneficial
                                momentum=0.9, weight_decay=0)

    # and a learning rate scheduler
    lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                                   step_size=10,
                                                   gamma=0.1)

    # let's train!
    for epoch in range(num_epochs):


        # train for one epoch, printing every 50 iterations
        train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=50)
        # update the learning rate
        lr_scheduler.step()
```

```
--------------------------------
FOLD 0
--------------------------------
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretraine
d' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a w
eight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current beha
vior is equivalent to passing `weights=FasterRCNN_ResNet50_FPN_Weights.COCO_V1`. You can also use `weights=Fast
erRCNN_ResNet50_FPN_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/fasterrcnn_resnet50_fpn_coco-258fb6c6.pth" to /root/.cache/to
rch/hub/checkpoints/fasterrcnn_resnet50_fpn_coco-258fb6c6.pth
100%|██████████| 160M/160M [00:02<00:00, 65.3MB/s]
Epoch: [0]  [ 0/11]  eta: 0:01:50  lr: 0.000504  loss: 0.9679 (0.9679)  loss_classifier: 0.7041 (0.7041)  loss_
box_reg: 0.1153 (0.1153)  loss_objectness: 0.1332 (0.1332)  loss_rpn_box_reg: 0.0153 (0.0153)  time: 10.0518  d
ata: 0.8384  max mem: 7206
Epoch: [0]  [10/11]  eta: 0:00:02  lr: 0.005000  loss: 1.1416 (1.1287)  loss_classifier: 0.3819 (0.4371)  loss_
box_reg: 0.3556 (0.3371)  loss_objectness: 0.2422 (0.3079)  loss_rpn_box_reg: 0.0386 (0.0466)  time: 2.1460  da
ta: 0.1049  max mem: 7366
Epoch: [0] Total time: 0:00:23 (2.1522 s / it)
Epoch: [1]  [ 0/11]  eta: 0:00:19  lr: 0.005000  loss: 0.6429 (0.6429)  loss_classifier: 0.2608 (0.2608)  loss_
box_reg: 0.3091 (0.3091)  loss_objectness: 0.0587 (0.0587)  loss_rpn_box_reg: 0.0142 (0.0142)  time: 1.7953  da
ta: 0.3579  max mem: 7366
Epoch: [1]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.6756 (0.7514)  loss_classifier: 0.2394 (0.2375)  loss_
box_reg: 0.3241 (0.3465)  loss_objectness: 0.0730 (0.1072)  loss_rpn_box_reg: 0.0426 (0.0603)  time: 1.4254  da
ta: 0.0633  max mem: 7366
Epoch: [1] Total time: 0:00:15 (1.4314 s / it)
Epoch: [2]  [ 0/11]  eta: 0:00:19  lr: 0.005000  loss: 0.5470 (0.5470)  loss_classifier: 0.1697 (0.1697)  loss_
box_reg: 0.2959 (0.2959)  loss_objectness: 0.0623 (0.0623)  loss_rpn_box_reg: 0.0192 (0.0192)  time: 1.7951  da
ta: 0.3562  max mem: 7366
Epoch: [2]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.5436 (0.5363)  loss_classifier: 0.1774 (0.1689)  loss_
box_reg: 0.2707 (0.2713)  loss_objectness: 0.0601 (0.0601)  loss_rpn_box_reg: 0.0295 (0.0359)  time: 1.4242  da
ta: 0.0597  max mem: 7366
Epoch: [2] Total time: 0:00:15 (1.4299 s / it)
Epoch: [3]  [ 0/11]  eta: 0:00:20  lr: 0.005000  loss: 0.6840 (0.6840)  loss_classifier: 0.2175 (0.2175)  loss_
box_reg: 0.3399 (0.3399)  loss_objectness: 0.0769 (0.0769)  loss_rpn_box_reg: 0.0497 (0.0497)  time: 1.8625  da
ta: 0.3780  max mem: 7366
Epoch: [3]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4765 (0.4549)  loss_classifier: 0.1581 (0.1494)  loss_
```

```
box_reg: 0.2382 (0.2342)  loss_objectness: 0.0334 (0.0420)  loss_rpn_box_reg: 0.0257 (0.0293)  time: 1.4568  da
ta: 0.0618  max mem: 7366
Epoch: [3] Total time: 0:00:16 (1.4627 s / it)
Epoch: [4]  [ 0/11]  eta: 0:00:20  lr: 0.005000  loss: 0.3431 (0.3431)  loss_classifier: 0.1095 (0.1095)  loss_
box_reg: 0.1739 (0.1739)  loss_objectness: 0.0360 (0.0360)  loss_rpn_box_reg: 0.0237 (0.0237)  time: 1.8787  da
ta: 0.3760  max mem: 7367
Epoch: [4]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.3750 (0.4064)  loss_classifier: 0.1468 (0.1363)  loss_
box_reg: 0.2009 (0.2129)  loss_objectness: 0.0261 (0.0312)  loss_rpn_box_reg: 0.0237 (0.0261)  time: 1.4747  da
ta: 0.0620  max mem: 7367
Epoch: [4] Total time: 0:00:16 (1.4857 s / it)
FOLD 1
--------------------------------
Epoch: [0]  [ 0/11]  eta: 0:00:22  lr: 0.000504  loss: 1.1431 (1.1431)  loss_classifier: 0.4287 (0.4287)  loss_
box_reg: 0.3136 (0.3136)  loss_objectness: 0.3792 (0.3792)  loss_rpn_box_reg: 0.0216 (0.0216)  time: 2.0831  da
ta: 0.4860  max mem: 7367
Epoch: [0]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.9436 (1.0082)  loss_classifier: 0.3069 (0.3310)  loss_
box_reg: 0.3136 (0.3448)  loss_objectness: 0.2518 (0.2901)  loss_rpn_box_reg: 0.0301 (0.0423)  time: 1.5149  da
ta: 0.0710  max mem: 7367
Epoch: [0] Total time: 0:00:16 (1.5221 s / it)
Epoch: [1]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.7749 (0.7749)  loss_classifier: 0.2488 (0.2488)  loss_
box_reg: 0.4259 (0.4259)  loss_objectness: 0.0591 (0.0591)  loss_rpn_box_reg: 0.0410 (0.0410)  time: 1.9511  da
ta: 0.4048  max mem: 7367
Epoch: [1]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.6573 (0.6580)  loss_classifier: 0.1904 (0.2059)  loss_
box_reg: 0.2946 (0.3232)  loss_objectness: 0.0563 (0.0854)  loss_rpn_box_reg: 0.0317 (0.0436)  time: 1.5562  da
ta: 0.0644  max mem: 7367
Epoch: [1] Total time: 0:00:17 (1.5622 s / it)
Epoch: [2]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.3862 (0.3862)  loss_classifier: 0.1077 (0.1077)  loss_
box_reg: 0.2000 (0.2000)  loss_objectness: 0.0542 (0.0542)  loss_rpn_box_reg: 0.0242 (0.0242)  time: 1.9441  da
ta: 0.3755  max mem: 7367
Epoch: [2]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4820 (0.4833)  loss_classifier: 0.1528 (0.1533)  loss_
box_reg: 0.2509 (0.2461)  loss_objectness: 0.0532 (0.0547)  loss_rpn_box_reg: 0.0275 (0.0292)  time: 1.5788  da
ta: 0.0603  max mem: 7367
Epoch: [2] Total time: 0:00:17 (1.5849 s / it)
Epoch: [3]  [ 0/11]  eta: 0:00:22  lr: 0.005000  loss: 0.3967 (0.3967)  loss_classifier: 0.1246 (0.1246)  loss_
box_reg: 0.2185 (0.2185)  loss_objectness: 0.0318 (0.0318)  loss_rpn_box_reg: 0.0218 (0.0218)  time: 2.0885  da
ta: 0.4659  max mem: 7367
Epoch: [3]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4050 (0.4280)  loss_classifier: 0.1365 (0.1369)  loss_
box_reg: 0.2185 (0.2257)  loss_objectness: 0.0351 (0.0380)  loss_rpn_box_reg: 0.0218 (0.0274)  time: 1.5874  da
ta: 0.0725  max mem: 7367
Epoch: [3] Total time: 0:00:17 (1.5971 s / it)
Epoch: [4]  [ 0/11]  eta: 0:00:23  lr: 0.005000  loss: 0.4118 (0.4118)  loss_classifier: 0.1123 (0.1123)  loss_
box_reg: 0.1820 (0.1820)  loss_objectness: 0.0615 (0.0615)  loss_rpn_box_reg: 0.0560 (0.0560)  time: 2.1395  da
ta: 0.4970  max mem: 7367
Epoch: [4]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.3397 (0.3827)  loss_classifier: 0.1135 (0.1274)  loss_
box_reg: 0.1820 (0.2029)  loss_objectness: 0.0276 (0.0276)  loss_rpn_box_reg: 0.0242 (0.0248)  time: 1.5532  da
ta: 0.0745  max mem: 7368
Epoch: [4] Total time: 0:00:17 (1.5650 s / it)
FOLD 2
--------------------------------
Epoch: [0]  [ 0/11]  eta: 0:00:20  lr: 0.000504  loss: 2.0973 (2.0973)  loss_classifier: 1.1516 (1.1516)  loss_
box_reg: 0.1972 (0.1972)  loss_objectness: 0.7174 (0.7174)  loss_rpn_box_reg: 0.0311 (0.0311)  time: 1.9017  da
ta: 0.3598  max mem: 7368
Epoch: [0]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 1.4706 (1.3245)  loss_classifier: 0.4958 (0.5884)  loss_
box_reg: 0.3552 (0.3564)  loss_objectness: 0.2586 (0.3317)  loss_rpn_box_reg: 0.0363 (0.0481)  time: 1.5309  da
ta: 0.0592  max mem: 7368
Epoch: [0] Total time: 0:00:16 (1.5371 s / it)
Epoch: [1]  [ 0/11]  eta: 0:00:23  lr: 0.005000  loss: 0.9844 (0.9844)  loss_classifier: 0.4252 (0.4252)  loss_
box_reg: 0.4155 (0.4155)  loss_objectness: 0.0612 (0.0612)  loss_rpn_box_reg: 0.0826 (0.0826)  time: 2.1496  da
ta: 0.5110  max mem: 7368
Epoch: [1]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.8089 (0.8069)  loss_classifier: 0.2462 (0.2508)  loss_
box_reg: 0.3607 (0.3634)  loss_objectness: 0.0879 (0.1313)  loss_rpn_box_reg: 0.0539 (0.0615)  time: 1.5805  da
ta: 0.0712  max mem: 7368
Epoch: [1] Total time: 0:00:17 (1.5919 s / it)
Epoch: [2]  [ 0/11]  eta: 0:00:23  lr: 0.005000  loss: 0.6949 (0.6949)  loss_classifier: 0.2292 (0.2292)  loss_
box_reg: 0.3630 (0.3630)  loss_objectness: 0.0691 (0.0691)  loss_rpn_box_reg: 0.0336 (0.0336)  time: 2.1056  da
ta: 0.4896  max mem: 7368
Epoch: [2]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.5688 (0.6070)  loss_classifier: 0.1869 (0.2086)  loss_
box_reg: 0.2850 (0.2892)  loss_objectness: 0.0662 (0.0714)  loss_rpn_box_reg: 0.0336 (0.0379)  time: 1.5869  da
ta: 0.0758  max mem: 7368
Epoch: [2] Total time: 0:00:17 (1.5933 s / it)
Epoch: [3]  [ 0/11]  eta: 0:00:22  lr: 0.005000  loss: 0.3602 (0.3602)  loss_classifier: 0.1070 (0.1070)  loss_
box_reg: 0.2070 (0.2070)  loss_objectness: 0.0277 (0.0277)  loss_rpn_box_reg: 0.0185 (0.0185)  time: 2.0110  da
ta: 0.4337  max mem: 7368
Epoch: [3]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4780 (0.4803)  loss_classifier: 0.1649 (0.1597)  loss_
box_reg: 0.2519 (0.2427)  loss_objectness: 0.0360 (0.0471)  loss_rpn_box_reg: 0.0266 (0.0308)  time: 1.5680  da
ta: 0.0698  max mem: 7368
Epoch: [3] Total time: 0:00:17 (1.5741 s / it)
Epoch: [4]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.3777 (0.3777)  loss_classifier: 0.1534 (0.1534)  loss_
box_reg: 0.1878 (0.1878)  loss_objectness: 0.0281 (0.0281)  loss_rpn_box_reg: 0.0085 (0.0085)  time: 1.9342  da
ta: 0.3701  max mem: 7368
Epoch: [4]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4280 (0.4582)  loss_classifier: 0.1489 (0.1550)  loss_
box_reg: 0.2124 (0.2350)  loss_objectness: 0.0336 (0.0393)  loss_rpn_box_reg: 0.0283 (0.0290)  time: 1.5487  da
ta: 0.0650  max mem: 7368
Epoch: [4] Total time: 0:00:17 (1.5586 s / it)
FOLD 3
--------------------------------
Epoch: [0]  [ 0/11]  eta: 0:00:24  lr: 0.000504  loss: 2.3694 (2.3694)  loss_classifier: 0.7868 (0.7868)  loss_
box_reg: 0.3152 (0.3152)  loss_objectness: 1.2089 (1.2089)  loss_rpn_box_reg: 0.0585 (0.0585)  time: 2.2451  da
ta: 0.5599  max mem: 7368
```

```
Epoch: [0]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.9874 (1.2505)  loss_classifier: 0.3631 (0.4746)  loss_
box_reg: 0.2988 (0.3049)  loss_objectness: 0.2875 (0.4321)  loss_rpn_box_reg: 0.0387 (0.0389)  time: 1.5889  da
ta: 0.0822  max mem: 7368
Epoch: [0] Total time: 0:00:17 (1.6051 s / it)
Epoch: [1]  [ 0/11]  eta: 0:00:26  lr: 0.005000  loss: 0.7438 (0.7438)  loss_classifier: 0.3418 (0.3418)  loss_
box_reg: 0.3410 (0.3410)  loss_objectness: 0.0470 (0.0470)  loss_rpn_box_reg: 0.0140 (0.0140)  time: 2.3759  da
ta: 0.7447  max mem: 7368
Epoch: [1]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.7107 (0.7081)  loss_classifier: 0.2207 (0.2256)  loss_
box_reg: 0.3410 (0.3257)  loss_objectness: 0.0688 (0.1034)  loss_rpn_box_reg: 0.0356 (0.0535)  time: 1.6042  da
ta: 0.0941  max mem: 7368
Epoch: [1] Total time: 0:00:17 (1.6104 s / it)
Epoch: [2]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.4963 (0.4963)  loss_classifier: 0.1545 (0.1545)  loss_
box_reg: 0.2411 (0.2411)  loss_objectness: 0.0838 (0.0838)  loss_rpn_box_reg: 0.0169 (0.0169)  time: 1.9473  da
ta: 0.3555  max mem: 7368
Epoch: [2]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4963 (0.5519)  loss_classifier: 0.1545 (0.1765)  loss_
box_reg: 0.2411 (0.2812)  loss_objectness: 0.0652 (0.0627)  loss_rpn_box_reg: 0.0300 (0.0316)  time: 1.5556  da
ta: 0.0598  max mem: 7368
Epoch: [2] Total time: 0:00:17 (1.5618 s / it)
Epoch: [3]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.3995 (0.3995)  loss_classifier: 0.1393 (0.1393)  loss_
box_reg: 0.2229 (0.2229)  loss_objectness: 0.0293 (0.0293)  loss_rpn_box_reg: 0.0080 (0.0080)  time: 1.9321  da
ta: 0.3677  max mem: 7368
Epoch: [3]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4720 (0.4609)  loss_classifier: 0.1541 (0.1511)  loss_
box_reg: 0.2429 (0.2424)  loss_objectness: 0.0422 (0.0408)  loss_rpn_box_reg: 0.0308 (0.0265)  time: 1.5564  da
ta: 0.0636  max mem: 7368
Epoch: [3] Total time: 0:00:17 (1.5635 s / it)
Epoch: [4]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.3145 (0.3145)  loss_classifier: 0.1070 (0.1070)  loss_
box_reg: 0.1699 (0.1699)  loss_objectness: 0.0259 (0.0259)  loss_rpn_box_reg: 0.0118 (0.0118)  time: 1.9514  da
ta: 0.3808  max mem: 7368
Epoch: [4]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4877 (0.4378)  loss_classifier: 0.1614 (0.1488)  loss_
box_reg: 0.2634 (0.2342)  loss_objectness: 0.0295 (0.0309)  loss_rpn_box_reg: 0.0215 (0.0239)  time: 1.5527  da
ta: 0.0646  max mem: 7368
Epoch: [4] Total time: 0:00:17 (1.5626 s / it)
FOLD 4
--------------------------------
Epoch: [0]  [ 0/11]  eta: 0:00:25  lr: 0.000504  loss: 1.5347 (1.5347)  loss_classifier: 0.8929 (0.8929)  loss_
box_reg: 0.3255 (0.3255)  loss_objectness: 0.2997 (0.2997)  loss_rpn_box_reg: 0.0166 (0.0166)  time: 2.3520  da
ta: 0.6721  max mem: 7368
Epoch: [0]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 1.1489 (1.3171)  loss_classifier: 0.4731 (0.5184)  loss_
box_reg: 0.3255 (0.3451)  loss_objectness: 0.2207 (0.4056)  loss_rpn_box_reg: 0.0486 (0.0481)  time: 1.5983  da
ta: 0.0897  max mem: 7368
Epoch: [0] Total time: 0:00:17 (1.6056 s / it)
Epoch: [1]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.7532 (0.7532)  loss_classifier: 0.3378 (0.3378)  loss_
box_reg: 0.3286 (0.3286)  loss_objectness: 0.0534 (0.0534)  loss_rpn_box_reg: 0.0334 (0.0334)  time: 1.9984  da
ta: 0.4120  max mem: 7368
Epoch: [1]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.7763 (0.7647)  loss_classifier: 0.2578 (0.2418)  loss_
box_reg: 0.3541 (0.3450)  loss_objectness: 0.0833 (0.1171)  loss_rpn_box_reg: 0.0372 (0.0609)  time: 1.5626  da
ta: 0.0647  max mem: 7368
Epoch: [1] Total time: 0:00:17 (1.5690 s / it)
Epoch: [2]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.5142 (0.5142)  loss_classifier: 0.1580 (0.1580)  loss_
box_reg: 0.2620 (0.2620)  loss_objectness: 0.0766 (0.0766)  loss_rpn_box_reg: 0.0175 (0.0175)  time: 1.9516  da
ta: 0.3854  max mem: 7368
Epoch: [2]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.5142 (0.5403)  loss_classifier: 0.1599 (0.1704)  loss_
box_reg: 0.2620 (0.2610)  loss_objectness: 0.0692 (0.0686)  loss_rpn_box_reg: 0.0340 (0.0403)  time: 1.5566  da
ta: 0.0638  max mem: 7368
Epoch: [2] Total time: 0:00:17 (1.5679 s / it)
Epoch: [3]  [ 0/11]  eta: 0:00:25  lr: 0.005000  loss: 0.2407 (0.2407)  loss_classifier: 0.0679 (0.0679)  loss_
box_reg: 0.1343 (0.1343)  loss_objectness: 0.0238 (0.0238)  loss_rpn_box_reg: 0.0148 (0.0148)  time: 2.3532  da
ta: 0.7236  max mem: 7368
Epoch: [3]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4643 (0.4727)  loss_classifier: 0.1611 (0.1592)  loss_
box_reg: 0.2263 (0.2370)  loss_objectness: 0.0424 (0.0444)  loss_rpn_box_reg: 0.0344 (0.0321)  time: 1.5888  da
ta: 0.0951  max mem: 7368
Epoch: [3] Total time: 0:00:17 (1.5952 s / it)
Epoch: [4]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.5727 (0.5727)  loss_classifier: 0.1915 (0.1915)  loss_
box_reg: 0.3141 (0.3141)  loss_objectness: 0.0292 (0.0292)  loss_rpn_box_reg: 0.0378 (0.0378)  time: 1.9716  da
ta: 0.3980  max mem: 7368
Epoch: [4]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.3670 (0.4064)  loss_classifier: 0.1316 (0.1373)  loss_
box_reg: 0.1900 (0.2019)  loss_objectness: 0.0323 (0.0385)  loss_rpn_box_reg: 0.0267 (0.0287)  time: 1.5581  da
ta: 0.0651  max mem: 7368
Epoch: [4] Total time: 0:00:17 (1.5647 s / it)
FOLD 5
--------------------------------
Epoch: [0]  [ 0/11]  eta: 0:00:22  lr: 0.000504  loss: 2.6834 (2.6834)  loss_classifier: 1.1206 (1.1206)  loss_
box_reg: 0.3796 (0.3796)  loss_objectness: 1.1325 (1.1325)  loss_rpn_box_reg: 0.0506 (0.0506)  time: 2.0315  da
ta: 0.4251  max mem: 7368
Epoch: [0]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 1.0846 (1.3300)  loss_classifier: 0.5145 (0.6097)  loss_
box_reg: 0.3445 (0.3176)  loss_objectness: 0.2125 (0.3595)  loss_rpn_box_reg: 0.0335 (0.0432)  time: 1.5703  da
ta: 0.0683  max mem: 7368
Epoch: [0] Total time: 0:00:17 (1.5768 s / it)
Epoch: [1]  [ 0/11]  eta: 0:00:24  lr: 0.005000  loss: 1.1896 (1.1896)  loss_classifier: 0.5045 (0.5045)  loss_
box_reg: 0.5542 (0.5542)  loss_objectness: 0.0943 (0.0943)  loss_rpn_box_reg: 0.0366 (0.0366)  time: 2.2594  da
ta: 0.6504  max mem: 7368
Epoch: [1]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.7551 (0.7412)  loss_classifier: 0.2279 (0.2523)  loss_
box_reg: 0.2800 (0.3200)  loss_objectness: 0.1073 (0.1120)  loss_rpn_box_reg: 0.0429 (0.0570)  time: 1.5799  da
ta: 0.0841  max mem: 7368
Epoch: [1] Total time: 0:00:17 (1.5864 s / it)
Epoch: [2]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.6807 (0.6807)  loss_classifier: 0.2125 (0.2125)  loss_
box_reg: 0.3307 (0.3307)  loss_objectness: 0.0939 (0.0939)  loss_rpn_box_reg: 0.0436 (0.0436)  time: 1.9605  da
ta: 0.3936  max mem: 7368
Epoch: [2]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.5318 (0.5821)  loss_classifier: 0.2015 (0.1978)  loss_
```

```
box_reg: 0.2902 (0.2754)  loss_objectness: 0.0623 (0.0702)  loss_rpn_box_reg: 0.0399 (0.0386)  time: 1.5635  da
ta: 0.0646  max mem: 7368
Epoch: [2] Total time: 0:00:17 (1.5699 s / it)
Epoch: [3]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.4525 (0.4525)  loss_classifier: 0.1905 (0.1905)  loss_
box_reg: 0.1677 (0.1677)  loss_objectness: 0.0762 (0.0762)  loss_rpn_box_reg: 0.0182 (0.0182)  time: 1.9987  da
ta: 0.4146  max mem: 7368
Epoch: [3]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4979 (0.5175)  loss_classifier: 0.1776 (0.1842)  loss_
box_reg: 0.2511 (0.2471)  loss_objectness: 0.0430 (0.0519)  loss_rpn_box_reg: 0.0308 (0.0343)  time: 1.5573  da
ta: 0.0674  max mem: 7368
Epoch: [3] Total time: 0:00:17 (1.5644 s / it)
Epoch: [4]  [ 0/11]  eta: 0:00:22  lr: 0.005000  loss: 0.3884 (0.3884)  loss_classifier: 0.1772 (0.1772)  loss_
box_reg: 0.1503 (0.1503)  loss_objectness: 0.0426 (0.0426)  loss_rpn_box_reg: 0.0184 (0.0184)  time: 2.0025  da
ta: 0.4252  max mem: 7368
Epoch: [4]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4659 (0.4429)  loss_classifier: 0.1647 (0.1608)  loss_
box_reg: 0.2370 (0.2152)  loss_objectness: 0.0370 (0.0400)  loss_rpn_box_reg: 0.0233 (0.0269)  time: 1.5609  da
ta: 0.0682  max mem: 7368
Epoch: [4] Total time: 0:00:17 (1.5724 s / it)
FOLD 6
--------------------------------
Epoch: [0]  [ 0/11]  eta: 0:00:21  lr: 0.000504  loss: 2.2202 (2.2202)  loss_classifier: 0.7471 (0.7471)  loss_
box_reg: 0.3345 (0.3345)  loss_objectness: 1.0857 (1.0857)  loss_rpn_box_reg: 0.0529 (0.0529)  time: 1.9465  da
ta: 0.3544  max mem: 7368
Epoch: [0]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 1.2663 (1.3242)  loss_classifier: 0.4882 (0.4784)  loss_
box_reg: 0.3929 (0.3604)  loss_objectness: 0.2516 (0.4356)  loss_rpn_box_reg: 0.0468 (0.0497)  time: 1.5498  da
ta: 0.0604  max mem: 7368
Epoch: [0] Total time: 0:00:17 (1.5570 s / it)
Epoch: [1]  [ 0/11]  eta: 0:00:22  lr: 0.005000  loss: 1.1208 (1.1208)  loss_classifier: 0.4384 (0.4384)  loss_
box_reg: 0.5390 (0.5390)  loss_objectness: 0.0962 (0.0962)  loss_rpn_box_reg: 0.0471 (0.0471)  time: 2.0651  da
ta: 0.4584  max mem: 7368
Epoch: [1]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.7013 (0.7553)  loss_classifier: 0.1932 (0.2254)  loss_
box_reg: 0.2957 (0.3394)  loss_objectness: 0.0985 (0.1290)  loss_rpn_box_reg: 0.0471 (0.0614)  time: 1.5860  da
ta: 0.0716  max mem: 7368
Epoch: [1] Total time: 0:00:17 (1.5924 s / it)
Epoch: [2]  [ 0/11]  eta: 0:00:21  lr: 0.005000  loss: 0.5719 (0.5719)  loss_classifier: 0.1790 (0.1790)  loss_
box_reg: 0.2702 (0.2702)  loss_objectness: 0.1077 (0.1077)  loss_rpn_box_reg: 0.0151 (0.0151)  time: 1.9774  da
ta: 0.3892  max mem: 7368
Epoch: [2]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.5719 (0.5614)  loss_classifier: 0.1790 (0.1798)  loss_
box_reg: 0.2702 (0.2699)  loss_objectness: 0.0636 (0.0746)  loss_rpn_box_reg: 0.0278 (0.0371)  time: 1.5563  da
ta: 0.0631  max mem: 7368
Epoch: [2] Total time: 0:00:17 (1.5665 s / it)
Epoch: [3]  [ 0/11]  eta: 0:00:24  lr: 0.005000  loss: 0.7305 (0.7305)  loss_classifier: 0.2340 (0.2340)  loss_
box_reg: 0.3017 (0.3017)  loss_objectness: 0.1292 (0.1292)  loss_rpn_box_reg: 0.0656 (0.0656)  time: 2.2039  da
ta: 0.6025  max mem: 7368
Epoch: [3]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4915 (0.5037)  loss_classifier: 0.1602 (0.1708)  loss_
box_reg: 0.2474 (0.2500)  loss_objectness: 0.0403 (0.0492)  loss_rpn_box_reg: 0.0390 (0.0336)  time: 1.5671  da
ta: 0.0808  max mem: 7368
Epoch: [3] Total time: 0:00:17 (1.5733 s / it)
Epoch: [4]  [ 0/11]  eta: 0:00:22  lr: 0.005000  loss: 0.4910 (0.4910)  loss_classifier: 0.1651 (0.1651)  loss_
box_reg: 0.2530 (0.2530)  loss_objectness: 0.0361 (0.0361)  loss_rpn_box_reg: 0.0368 (0.0368)  time: 2.0428  da
ta: 0.4477  max mem: 7368
Epoch: [4]  [10/11]  eta: 0:00:01  lr: 0.005000  loss: 0.4191 (0.4613)  loss_classifier: 0.1504 (0.1559)  loss_
box_reg: 0.2259 (0.2344)  loss_objectness: 0.0313 (0.0414)  loss_rpn_box_reg: 0.0287 (0.0295)  time: 1.5661  da
ta: 0.0666  max mem: 7368
Epoch: [4] Total time: 0:00:17 (1.5725 s / it)
```

Se ha cambiado la parte de las funciones k-folds y num_epochs. Se han disminuido los números que queremos en cada una (ahora correspondientes a 7 y 5 respectivamente) puesto que no queremos que ejecute todo, solamente queremos saber si efectivamente el código funciona y así, no perder mucho tiempo. Igualmente hay que poner un número considerable para que el entrenamiento sea bueno y que la inferencia luego sea precisa. Por otro lado, también hemos eliminado la parte de la evaluación, ya que al ejecutar da error porque no tenemos la capacidad de poder evaluar el modelo.

Este fragmento de código realiza una validación cruzada K-fold para evaluar el rendimiento de un modelo de detección de objetos en un conjunto de datos de detección de cráteres. Lo que hace el código es lo siguiente:

1. Determina si se usará una GPU (`cuda`) o una CPU (`cpu`) para el entrenamiento y la evaluación del modelo, según la disponibilidad de GPU en el sistema.
2. Se define el número de clases (num_classes) en el problema de detección de objetos. En este caso, hay dos clases: "background" y "crater".
3. Se crea una instancia del conjunto de datos de entrenamiento (dataset) y del conjunto de validación (dataset_val) utilizando las rutas especificadas y las transformaciones apropiadas.
4. Define el número de pliegues `k_folds` y el número de épocas `num_epochs` para el proceso de validación cruzada y el entrenamiento del modelo.
5. Crea instancias de `CraterDataset` para el conjunto de datos de entrenamiento y validación, con transformaciones específicas aplicadas a cada uno.
6. Inicializa un objeto `KFold` para realizar la validación cruzada K-fold.
7. Itera sobre cada pliegue y divide el conjunto de datos en conjuntos de entrenamiento y validación correspondientes.

8. Crea cargadores de datos para el conjunto de datos de entrenamiento y validación utilizando `DataLoader`.

9. Obtiene el modelo utilizando la función `get_model_bbox` y lo mueve al dispositivo correspondiente.

10. Construye un optimizador y un programador de tasas de aprendizaje.

11. Inicia el bucle de entrenamiento en el que se entrena el modelo para un número específico de épocas, utilizando la función `train_one_epoch` para cada época y actualizando la tasa de aprendizaje utilizando el programador de tasas de aprendizaje.

In [ ]:
```
num_epochs = 5
# our dataset has two classes only - background and crater
num_classes = 2
# use our dataset and defined transformations
dataset = CraterDataset('/content/craters/train', get_transform(train=True))
dataset_test = CraterDataset('/content/craters/test', get_transform(train=False))

# define training and validation data loaders
data_loader = torch.utils.data.DataLoader(
        dataset, batch_size=8, shuffle=True, num_workers=2,
    collate_fn=utils.collate_fn)

data_loader_test = torch.utils.data.DataLoader(
    dataset_test, batch_size=1, shuffle=False, num_workers=2,
    collate_fn=utils.collate_fn)

# get the model using our helper function
model = get_model_bbox(num_classes)

'''
Use this to reset all trainable weights
model.apply(reset_weights)
'''

# move model to the right device
model.to(device)

# construct an optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,  # Feel free to play with values
                            momentum=0.9, weight_decay=0)

# Defining learning rate scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                               step_size=20,
                                               gamma=0.2)


result_mAP = []
best_epoch = None

# Let's train!
for epoch in range(num_epochs):


    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=50)
    # update the learning rate
    lr_scheduler.step()

# Saving the last model
save_path = os.path.join(f'Crater_noaug_sgd_2batch-lastepoch{num_epochs-1}.pth')
torch.save(model.state_dict(), save_path)
print(f'model from last epoch(no.{num_epochs-1}) saved')
```

```
Epoch: [0]  [ 0/13]  eta: 0:00:28  lr: 0.000421  loss: 2.1663 (2.1663)  loss_classifier: 0.6088 (0.6088)  loss_
box_reg: 0.4250 (0.4250)  loss_objectness: 1.0880 (1.0880)  loss_rpn_box_reg: 0.0446 (0.0446)  time: 2.1567  da
ta: 0.5888  max mem: 7368
Epoch: [0]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 1.0943 (1.1267)  loss_classifier: 0.3712 (0.3945)  loss_
box_reg: 0.3765 (0.3679)  loss_objectness: 0.1677 (0.3224)  loss_rpn_box_reg: 0.0429 (0.0420)  time: 1.4777  da
ta: 0.0765  max mem: 7368
Epoch: [0] Total time: 0:00:19 (1.4877 s / it)
Epoch: [1]  [ 0/13]  eta: 0:00:28  lr: 0.005000  loss: 0.5666 (0.5666)  loss_classifier: 0.1824 (0.1824)  loss_
box_reg: 0.2722 (0.2722)  loss_objectness: 0.0523 (0.0523)  loss_rpn_box_reg: 0.0597 (0.0597)  time: 2.2175  da
ta: 0.6738  max mem: 7368
Epoch: [1]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.6565 (0.6748)  loss_classifier: 0.2123 (0.2089)  loss_
box_reg: 0.3090 (0.3322)  loss_objectness: 0.0564 (0.0829)  loss_rpn_box_reg: 0.0466 (0.0508)  time: 1.5113  da
ta: 0.0847  max mem: 7368
Epoch: [1] Total time: 0:00:19 (1.5171 s / it)
Epoch: [2]  [ 0/13]  eta: 0:00:24  lr: 0.005000  loss: 0.6086 (0.6086)  loss_classifier: 0.1604 (0.1604)  loss_
box_reg: 0.2652 (0.2652)  loss_objectness: 0.1022 (0.1022)  loss_rpn_box_reg: 0.0808 (0.0808)  time: 1.9119  da
ta: 0.3906  max mem: 7368
Epoch: [2]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.4417 (0.4692)  loss_classifier: 0.1579 (0.1524)  loss_
box_reg: 0.2330 (0.2310)  loss_objectness: 0.0495 (0.0504)  loss_rpn_box_reg: 0.0300 (0.0354)  time: 1.4707  da
ta: 0.0598  max mem: 7368
Epoch: [2] Total time: 0:00:19 (1.4760 s / it)
Epoch: [3]  [ 0/13]  eta: 0:00:25  lr: 0.005000  loss: 0.5606 (0.5606)  loss_classifier: 0.1753 (0.1753)  loss_
box_reg: 0.3084 (0.3084)  loss_objectness: 0.0460 (0.0460)  loss_rpn_box_reg: 0.0309 (0.0309)  time: 1.9492  da
ta: 0.4028  max mem: 7368
Epoch: [3]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.4485 (0.4328)  loss_classifier: 0.1518 (0.1464)  loss_
box_reg: 0.2300 (0.2190)  loss_objectness: 0.0299 (0.0377)  loss_rpn_box_reg: 0.0292 (0.0296)  time: 1.4898  da
ta: 0.0606  max mem: 7368
Epoch: [3] Total time: 0:00:19 (1.5003 s / it)
Epoch: [4]  [ 0/13]  eta: 0:00:28  lr: 0.005000  loss: 0.2045 (0.2045)  loss_classifier: 0.0727 (0.0727)  loss_
box_reg: 0.1015 (0.1015)  loss_objectness: 0.0202 (0.0202)  loss_rpn_box_reg: 0.0100 (0.0100)  time: 2.2118  da
ta: 0.6196  max mem: 7368
Epoch: [4]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.4186 (0.4013)  loss_classifier: 0.1278 (0.1321)  loss_
box_reg: 0.2158 (0.2094)  loss_objectness: 0.0282 (0.0339)  loss_rpn_box_reg: 0.0208 (0.0259)  time: 1.5110  da
ta: 0.0775  max mem: 7368
Epoch: [4] Total time: 0:00:19 (1.5164 s / it)
model from last epoch(no.4) saved
```

En este código, de nuevo, se ha eliminado también la parte de evaluación debido a los límites de capacidad del entorno de ejecución. Y se ha bajado el número de Epochs (número de iteraciones) por la misma razón de antes, que queremos saber que el código funciona pero no tardar mucho tiempo en ejecutarlo.

Este fragmento de código realiza un proceso de entrenamiento de un modelo de detección de objetos en un conjunto de datos de detección de cráteres y guarda el modelo entrenado al final del proceso. El código hace los siguientes pasos:

1. Define el número de épocas `num_epochs` para el proceso de entrenamiento.
2. Crea instancias de `CraterDataset` para el conjunto de datos de entrenamiento y prueba, con transformaciones específicas aplicadas a cada uno.
3. Inicializa cargadores de datos para el conjunto de datos de entrenamiento y prueba utilizando `DataLoader`.
4. Obtiene el modelo utilizando la función `get_model_bbox`.
5. Mueve el modelo al dispositivo adecuado.
6. Construye un optimizador y un programador de tasas de aprendizaje.
7. Inicia el bucle de entrenamiento en el que se entrena el modelo para el número especificado de épocas, utilizando la función `train_one_epoch` para cada época y actualizando la tasa de aprendizaje utilizando el programador de tasas de aprendizaje.
8. Guarda el modelo entrenado al final del proceso en un archivo específico utilizando `torch.save`.

In [ ]:
```
dataset_test = CraterDataset('/content/craters/test', get_transform(train=False))

data_loader_test = torch.utils.data.DataLoader(
    dataset_test, batch_size=1, shuffle=False, num_workers=2,
    collate_fn=utils.collate_fn)

model = get_model_bbox(num_classes)
```

Hemos eliminado la parte de evaluación del modelo (por la misma razón que los anteriores códigos).

Este fragmento de código carga un conjunto de datos de prueba (`dataset_test`) utilizando la clase `CraterDataset` y define un cargador de datos (`data_loader_test`) para el conjunto de datos de prueba. Además, se obtiene un modelo utilizando la función `get_model_bbox` para su uso en la evaluación del conjunto de datos de prueba.

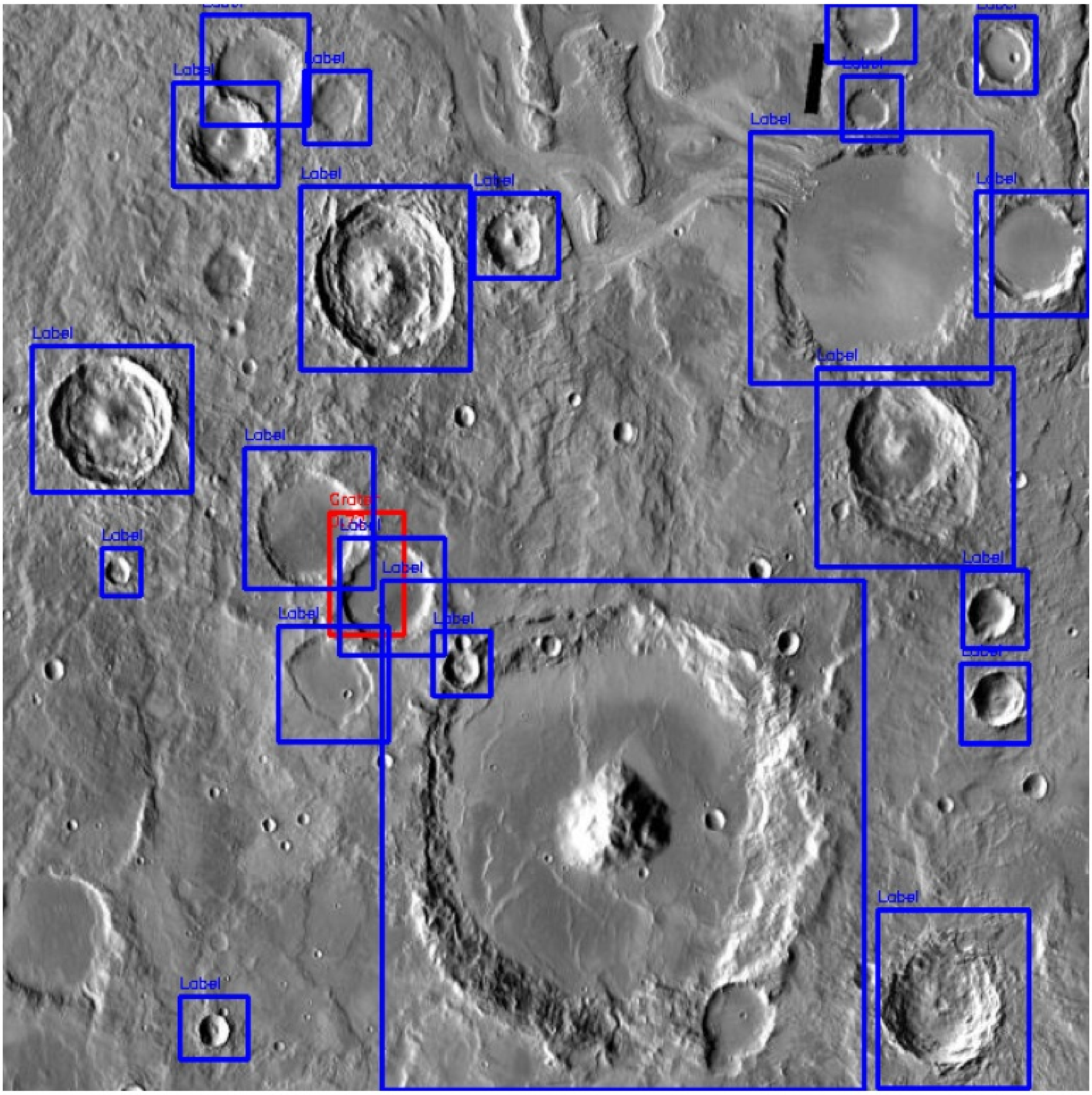
In [ ]:
```
# Define colors for bounding boxes
color_inference = np.array([0.0,0.0,255.0])
color_label = np.array([255.0,0.0,0.0])

# Score value thershold for displaying predictions
detection_threshold = 0.7
# to count the total number of images iterated through
frame_count = 0
# to keep adding the FPS for each image
total_fps = 0

!mkdir ./results
```

Este fragmento de código define algunas variables para el proceso de visualización de los resultados de la detección de objetos en un conjunto de datos de detección de cráteres. Entonces, hace las siguientes funciones:

1. Define los colores en formato de matriz para las cajas delimitadoras y las etiquetas.
2. Establece un umbral de valor de puntuación para la visualización de predicciones de detección.
3. Inicializa las variables `frame_count` y `total_fps` para realizar un seguimiento del recuento total de imágenes y del tiempo de fotogramas por segundo.
4. Crea un directorio llamado "results" donde se almacenarán los resultados de la detección.

In [ ]:
```
model.eval()
# Verificar la disponibilidad de CUDA y configurar el dispositivo
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Mover el modelo y los datos al dispositivo correspondiente
model.to(device)
model_image = model.to(device)

for i,data in enumerate(data_loader_test):
    # get the image file name for predictions file name
    image_name = 'image no:' + str(int(data[1][0]['image_id']))
    model_image = data[0][0]
    cv2_image = np.transpose(model_image.numpy()*255,(1, 2, 0)).astype(np.float32)
    cv2_image = cv2.cvtColor(cv2_image, cv2.COLOR_RGB2BGR).astype(np.float32)

    # add batch dimension
    model_image = torch.unsqueeze(model_image, 0)
    start_time = time.time()
    with torch.no_grad():
      outputs = model(model_image.to(device))
      end_time = time.time()
    # get the current fps
    fps = 1 / (end_time - start_time)
    # add `fps` to `total_fps`
    total_fps += fps
    # increment frame count
    frame_count += 1
    # load all detection to CPU for further operations
    outputs = [{k: v.to('cpu') for k, v in t.items()} for t in outputs]
    # carry further only if there's detected boxes
    if len(outputs[0]['boxes']) != 0:
        boxes = outputs[0]['boxes'].data.numpy()
        scores = outputs[0]['scores'].data.numpy()
        # filter out boxes according to `detection_threshold`
        boxes = boxes[scores >= detection_threshold].astype(np.int32)
        scores = np.round(scores[scores >= detection_threshold],2)
        draw_boxes = boxes.copy()


        # draw the bounding boxes and write the class name on top of it
        for j,box in enumerate(draw_boxes):
            cv2.rectangle(cv2_image,
                          (int(box[0]), int(box[1])),
                          (int(box[2]), int(box[3])),
                          color_inference, 2)
            cv2.putText(img=cv2_image, text="Crater",
                        org=(int(box[0]), int(box[1] - 5)),
                        fontFace=cv2.FONT_HERSHEY_SIMPLEX,fontScale= 0.3,color= color_inference,
                        thickness=1, lineType=cv2.LINE_AA)
            cv2.putText(img=cv2_image, text=str(scores[j]),
                        org=(int(box[0]), int(box[1] + 8)),
                        fontFace=cv2.FONT_HERSHEY_SIMPLEX,fontScale= 0.3,color= color_inference,
                        thickness=1, lineType=cv2.LINE_AA)

        # add boxes for labels
        for box in data[1][0]['boxes']:
            cv2.rectangle(cv2_image,
                          (int(box[0]), int(box[1])),
                          (int(box[2]), int(box[3])),
                          color_label, 2)
            cv2.putText(img=cv2_image, text="Label",
                        org=(int(box[0]), int(box[1] - 5)),
                        fontFace=cv2.FONT_HERSHEY_SIMPLEX,fontScale= 0.3,color= color_label,
                        thickness=1, lineType=cv2.LINE_AA)


        # set size
        plt.figure(figsize=(10,10))
        plt.axis("off")

        # convert color from CV2 BGR back to RGB
        plt_image = cv2.cvtColor(cv2_image/255.0, cv2.COLOR_BGR2RGB)
        plt.imshow(plt_image)
        plt.show()
        cv2.imwrite(f"./results/{image_name}.jpg", cv2_image)
    print(f"Image {i + 1} done...")
```

```
    print('-' * 50)
print('TEST PREDICTIONS COMPLETE')

avg_fps = total_fps / frame_count
print(f"Average FPS: {avg_fps:.3f}")
```




```
Image 1 done...
--------------------------------------------------
```

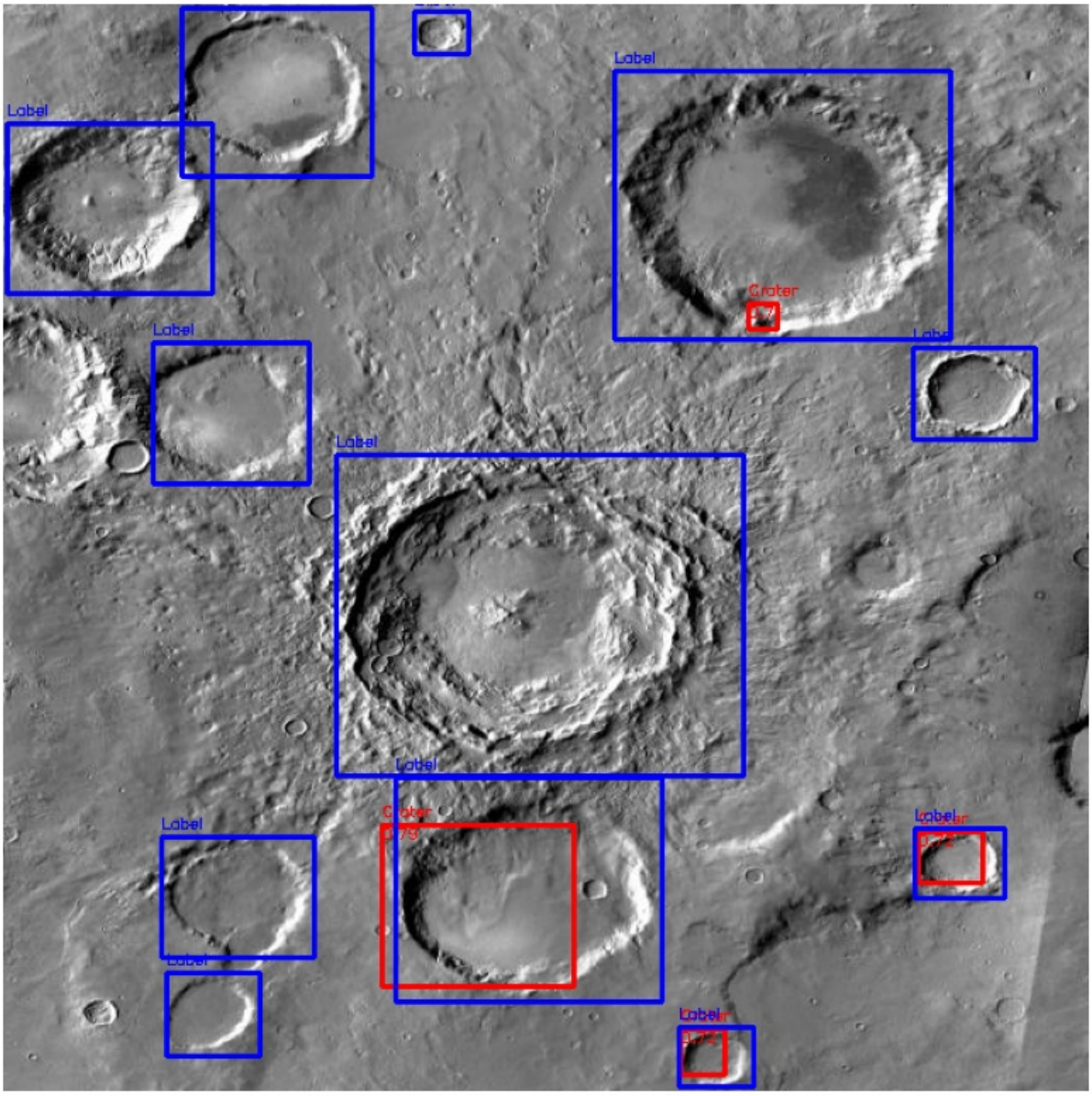

Label
Label
Label
Crater
Label
Label
Label
Crater
Label
Label
Label

Image 2 done...

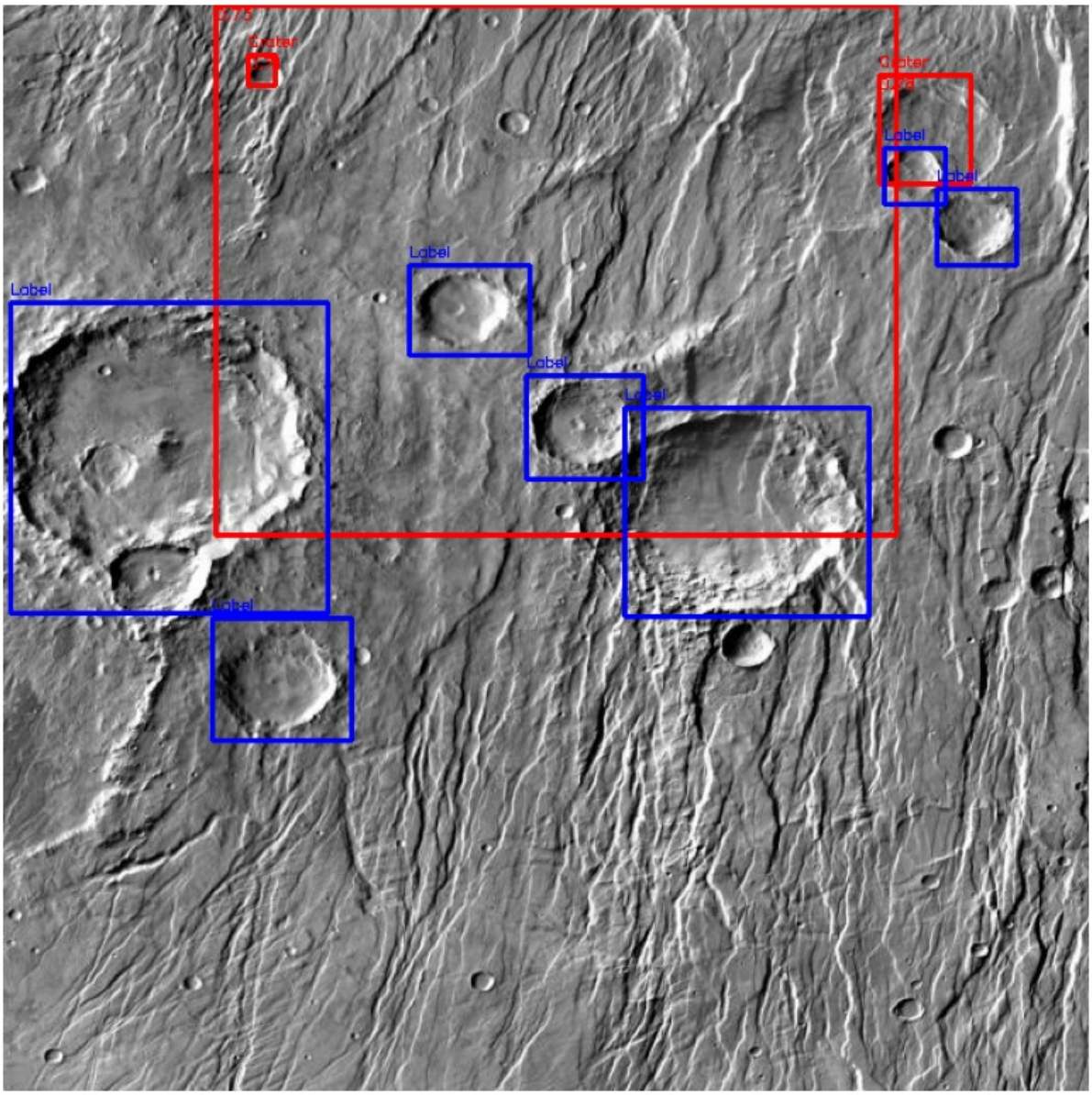


Image 3 done...

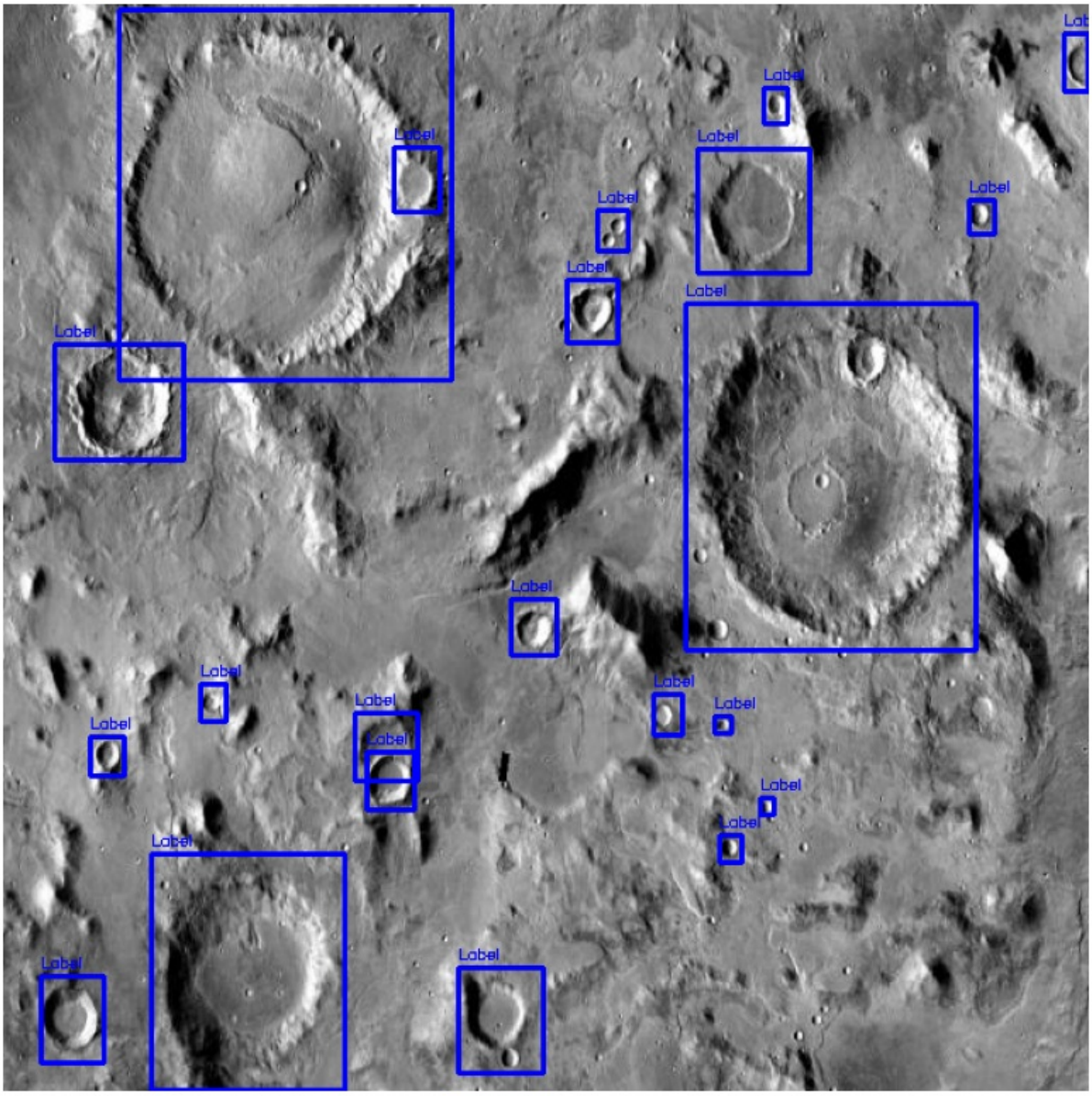
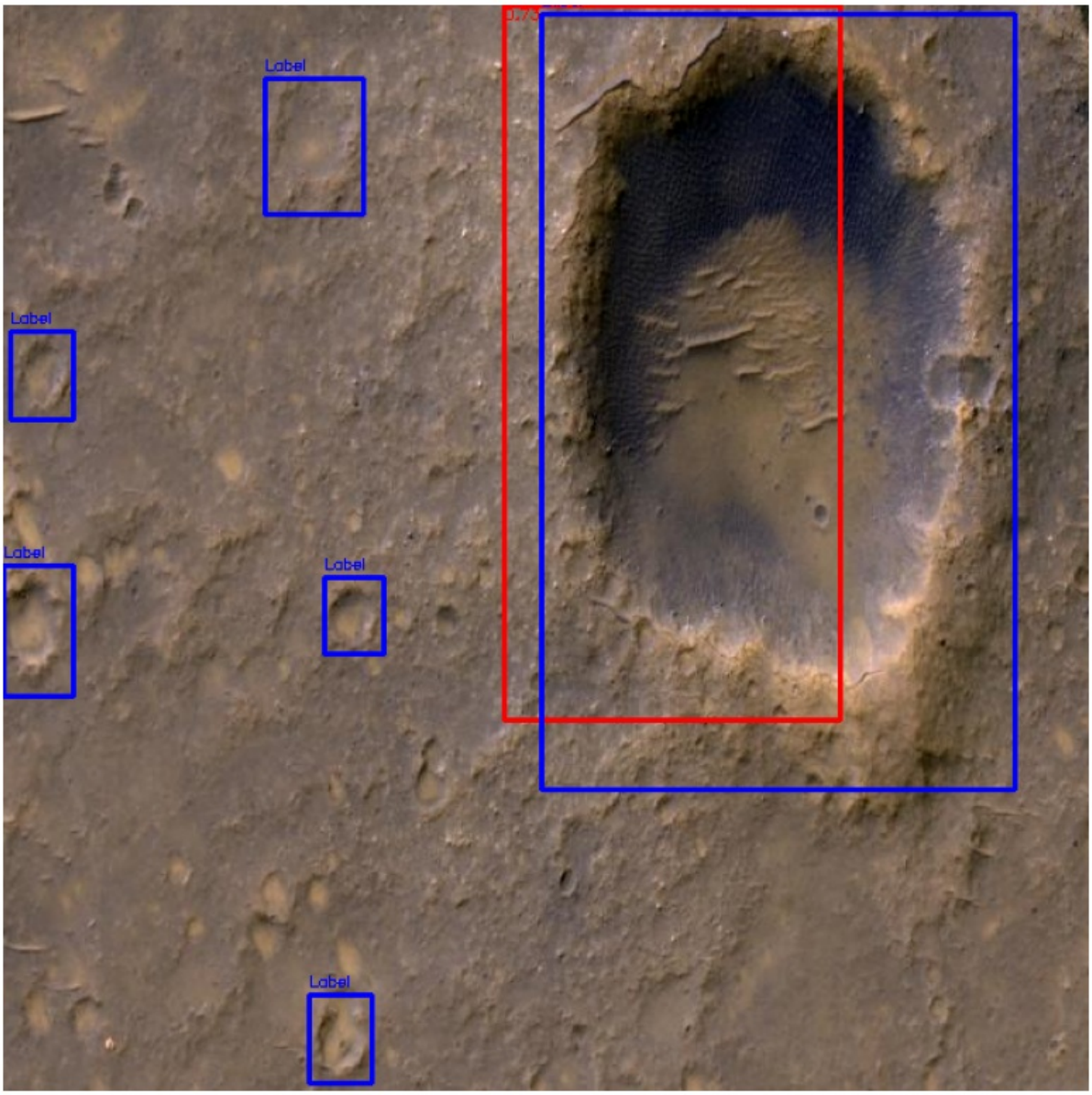

Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label

Image 4 done...




```
Image 5 done...
--------------------------------------------------
```

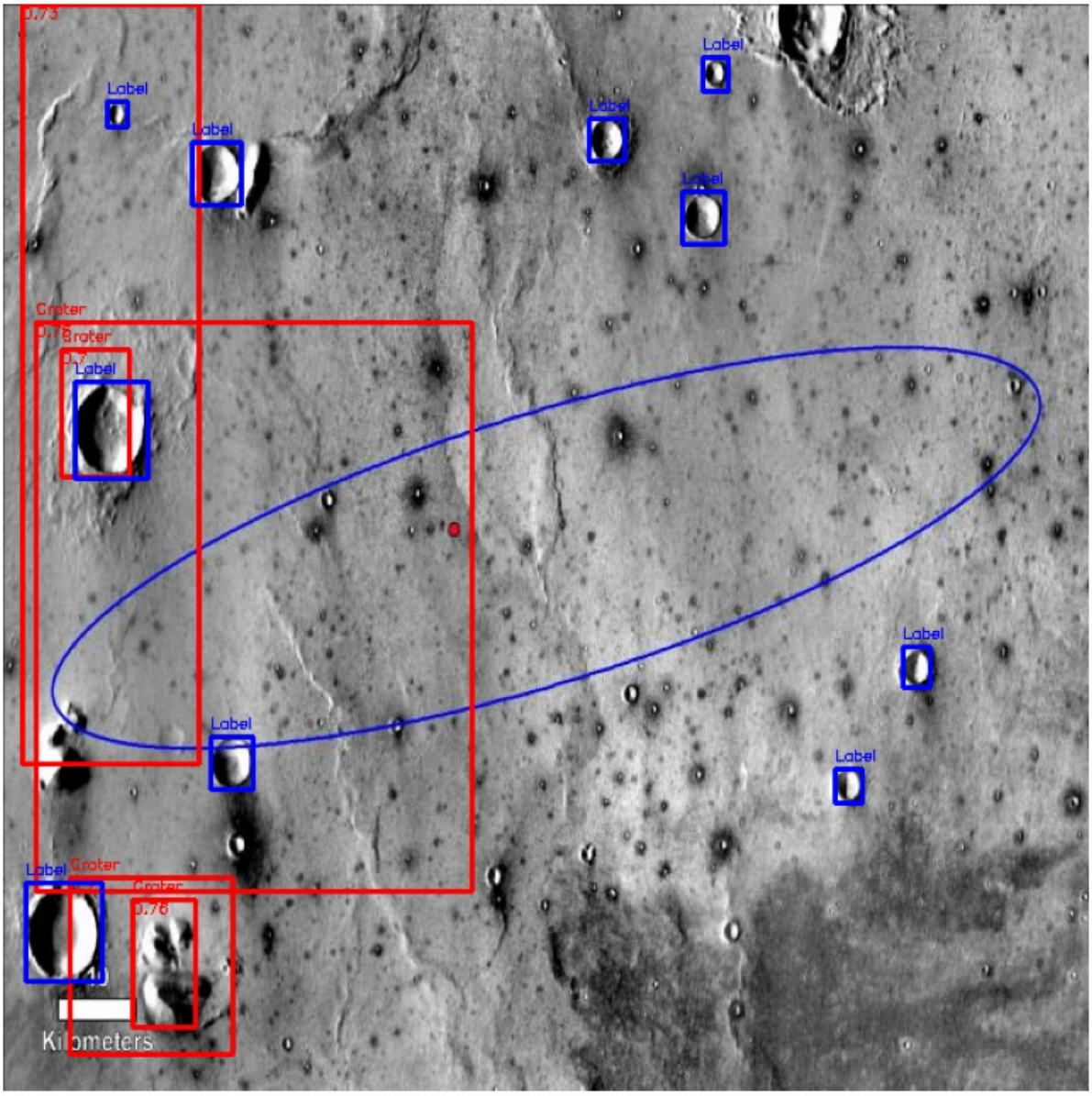

0.73
Label
Label
Label
Label
Label
Crater
Crater
Label
Label
Label
Label
Label
Crater
Crater
0.76
Kilometers

Image 6 done...

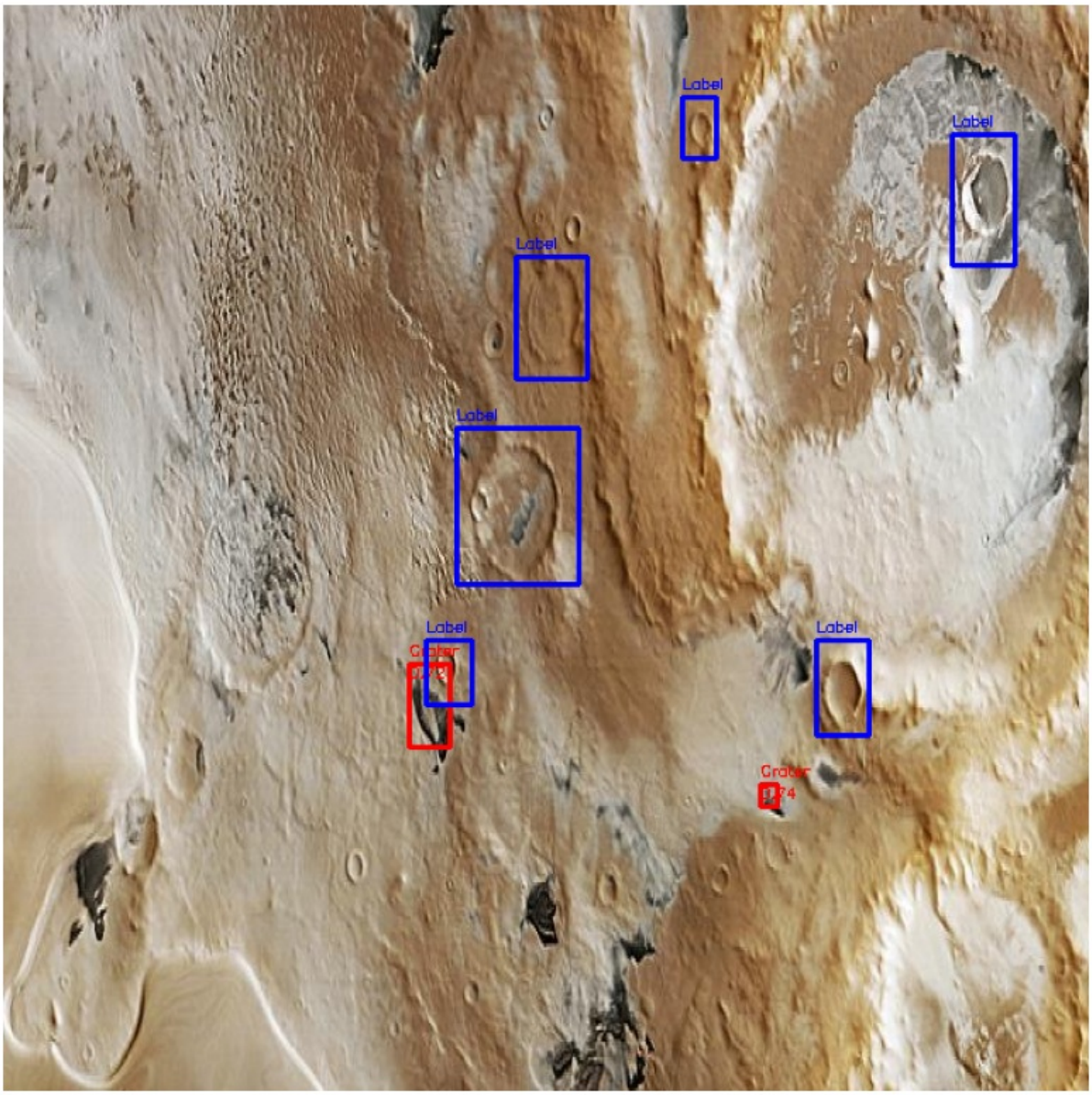


Image 7 done...

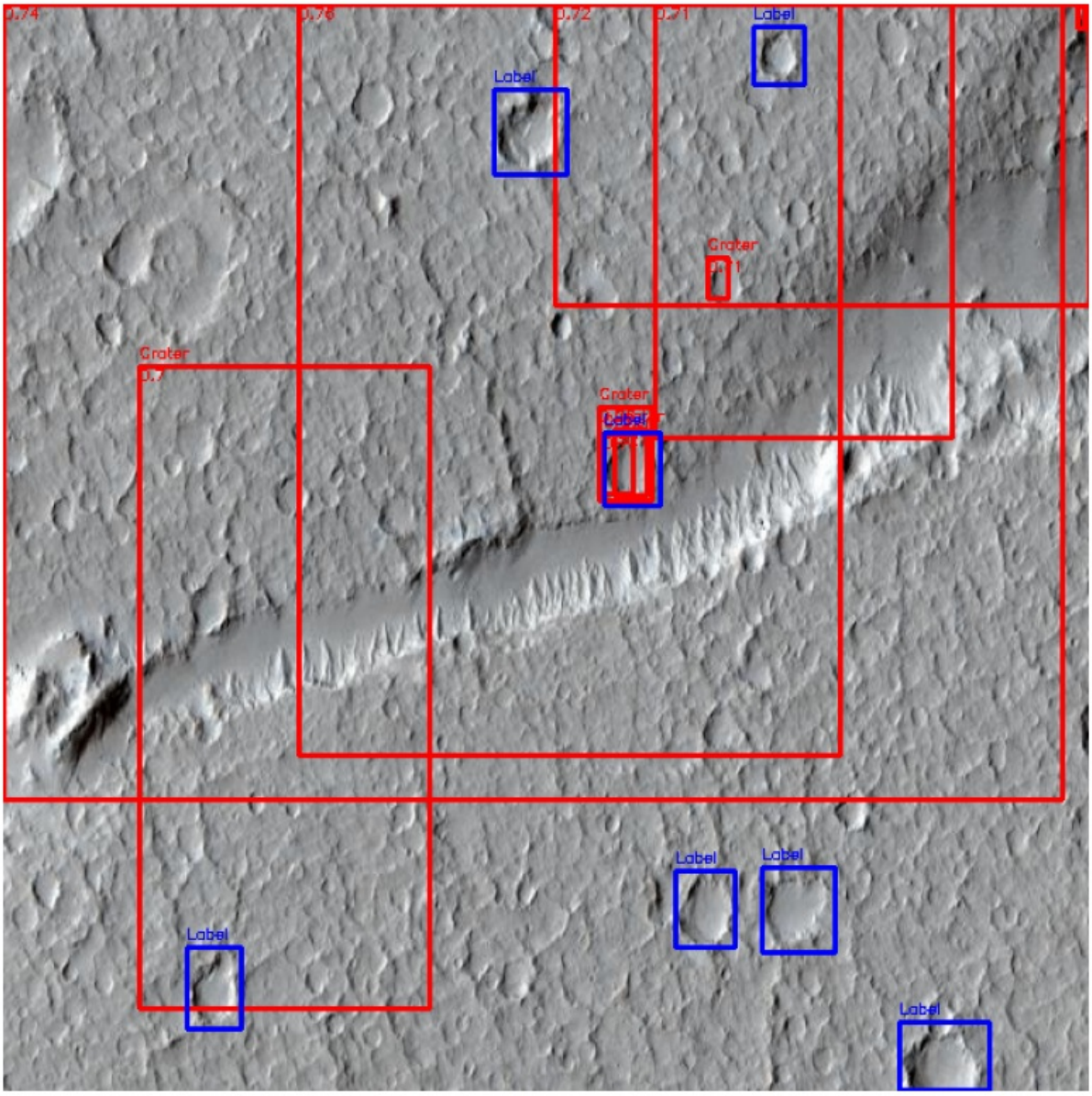

0.74
0.76
0.72
0.71
Label
Label
Crater
Crater
0.7
Crater
Label
Label
Label
Label
Label

Image 8 done...

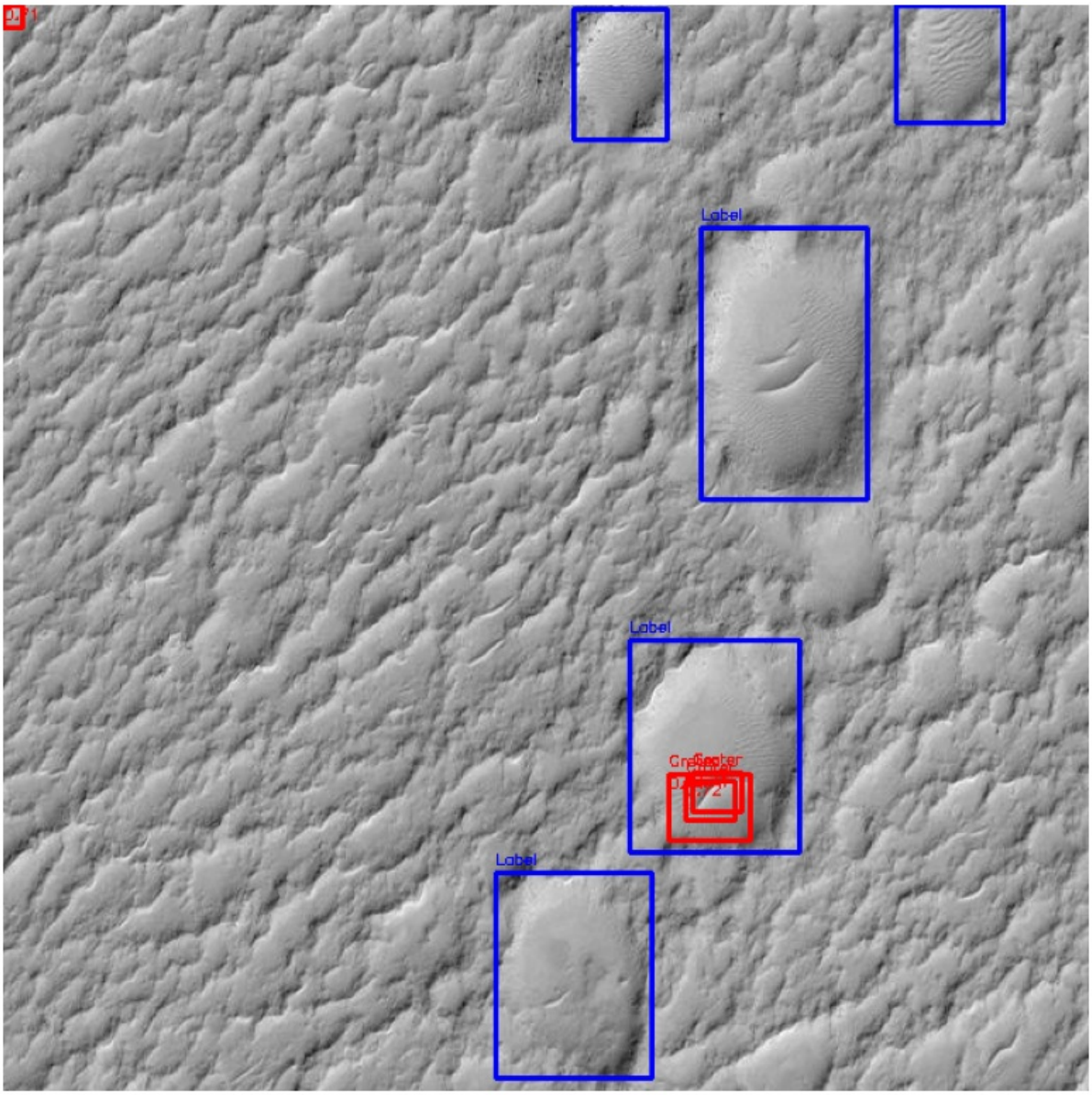
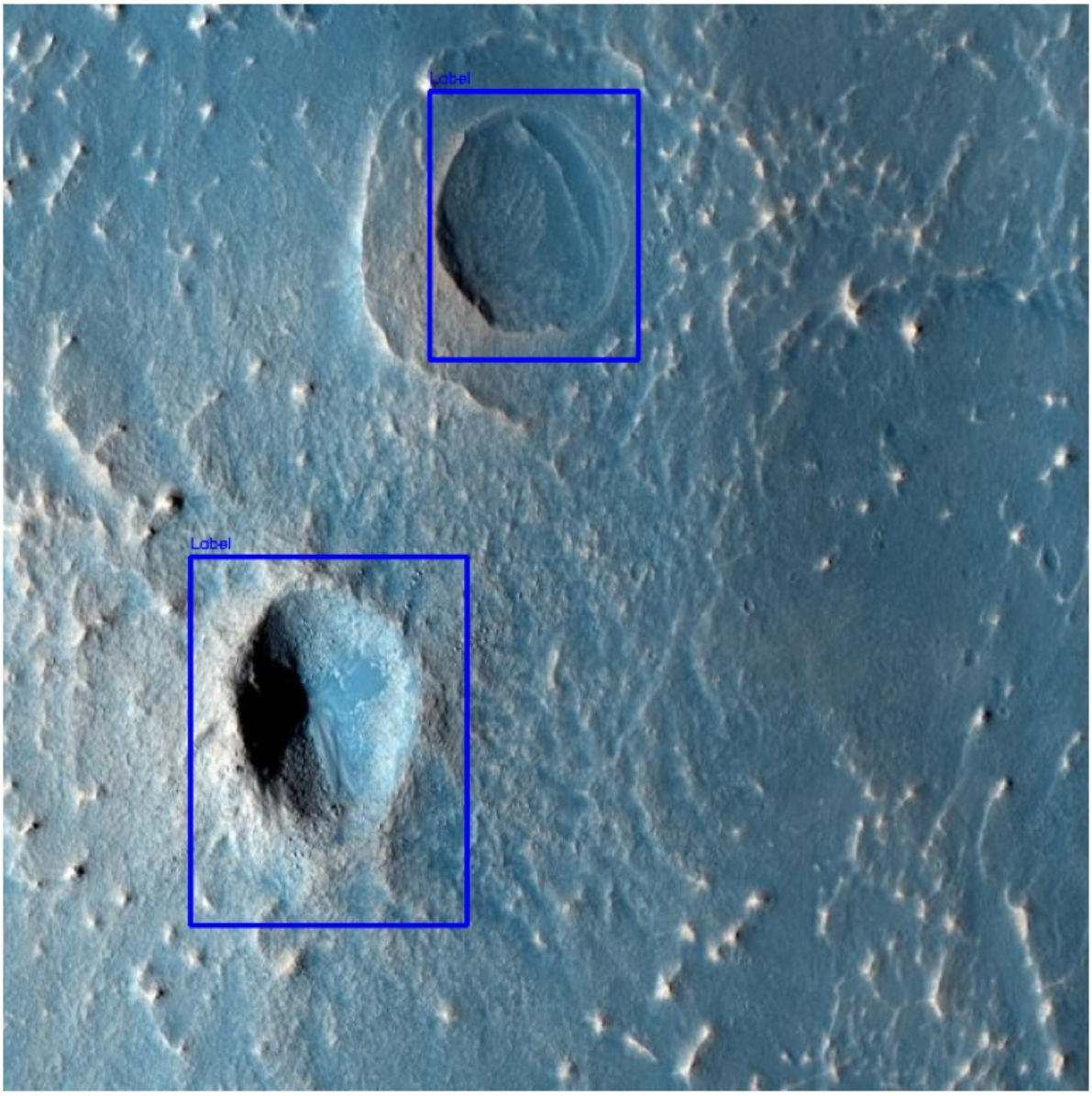


Image 9 done...




```
Image 10 done...
--------------------------------------------------
```

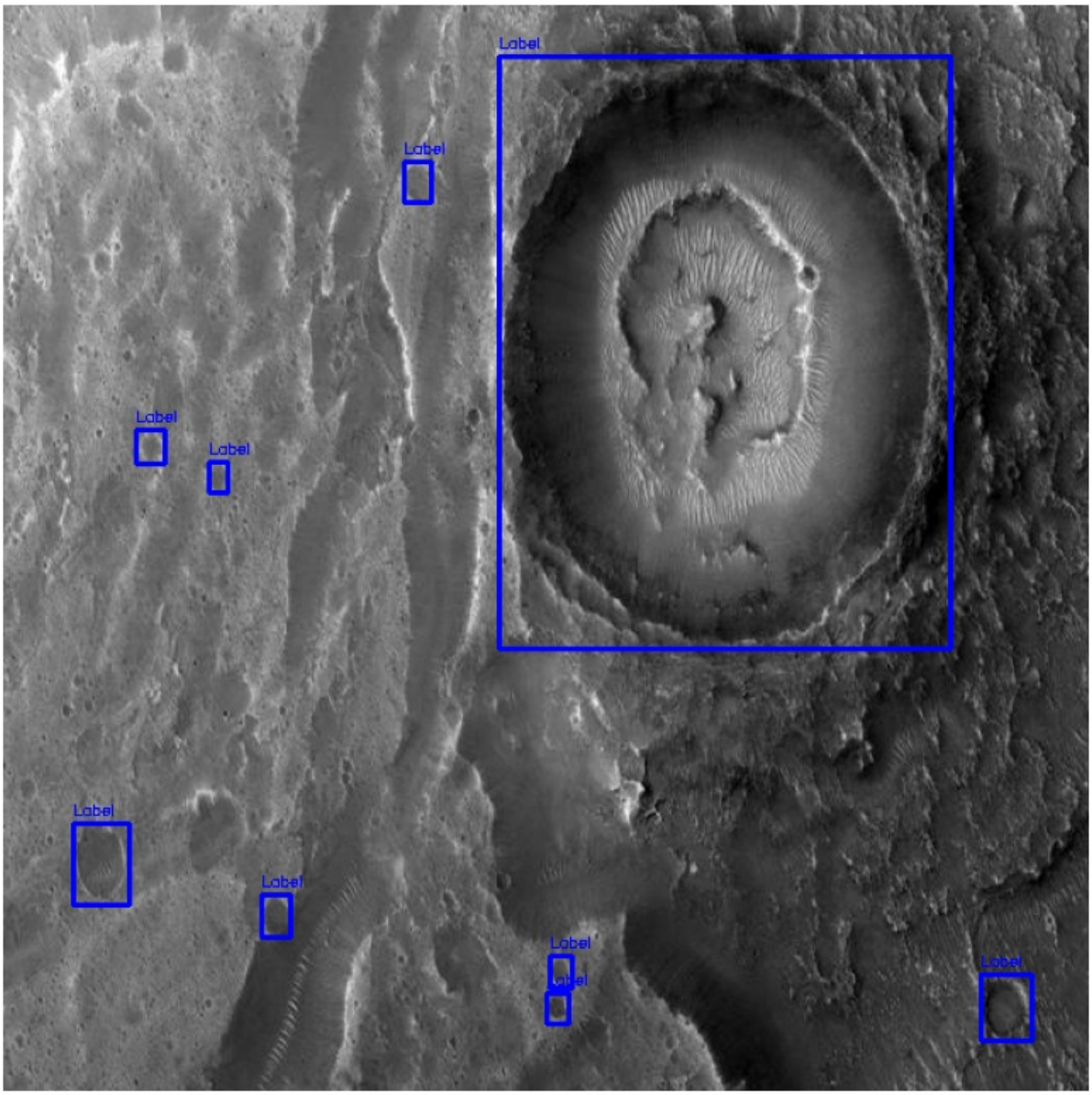
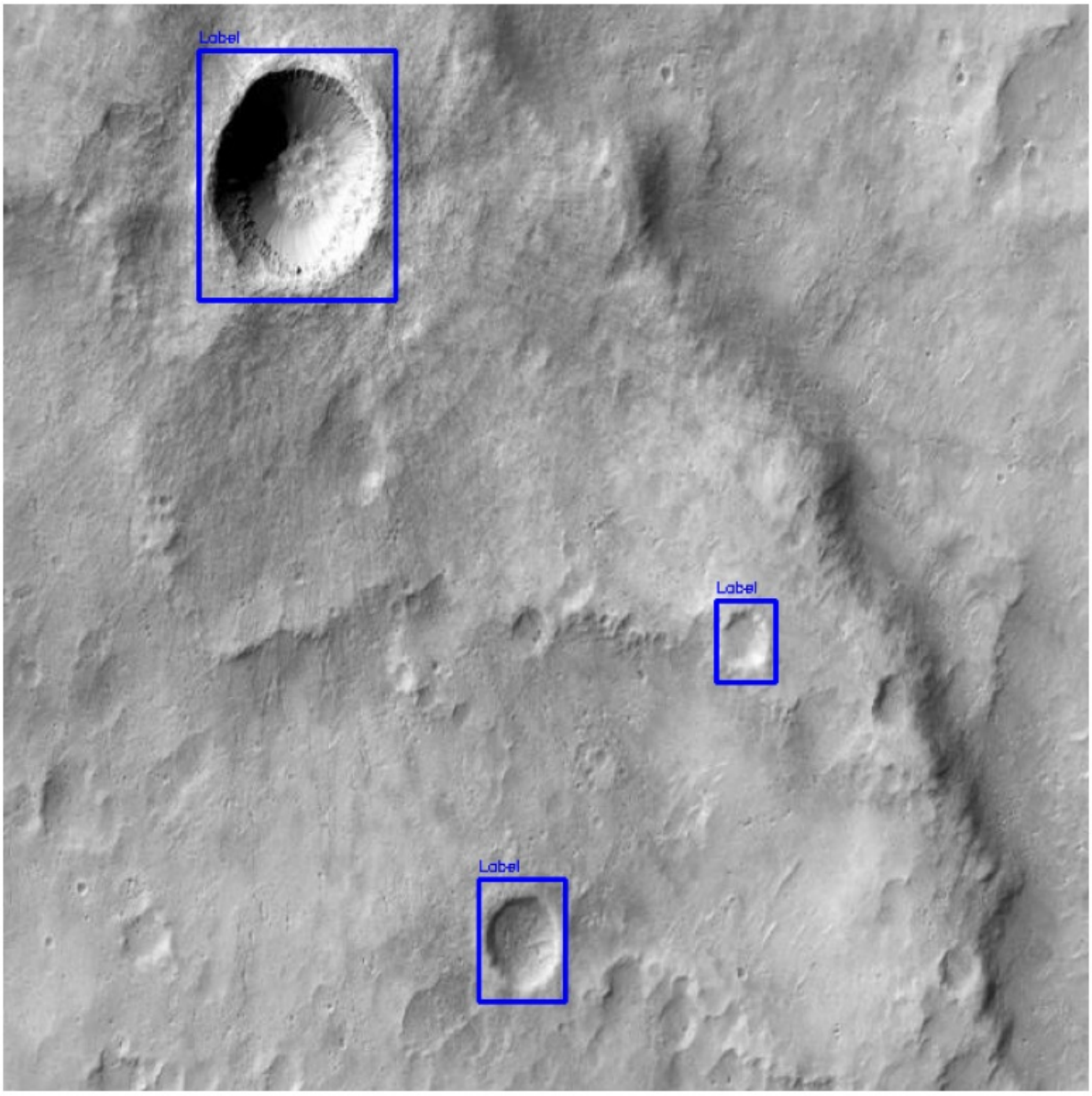


Image 11 done...




```
Image 12 done...
--------------------------------------------------
```

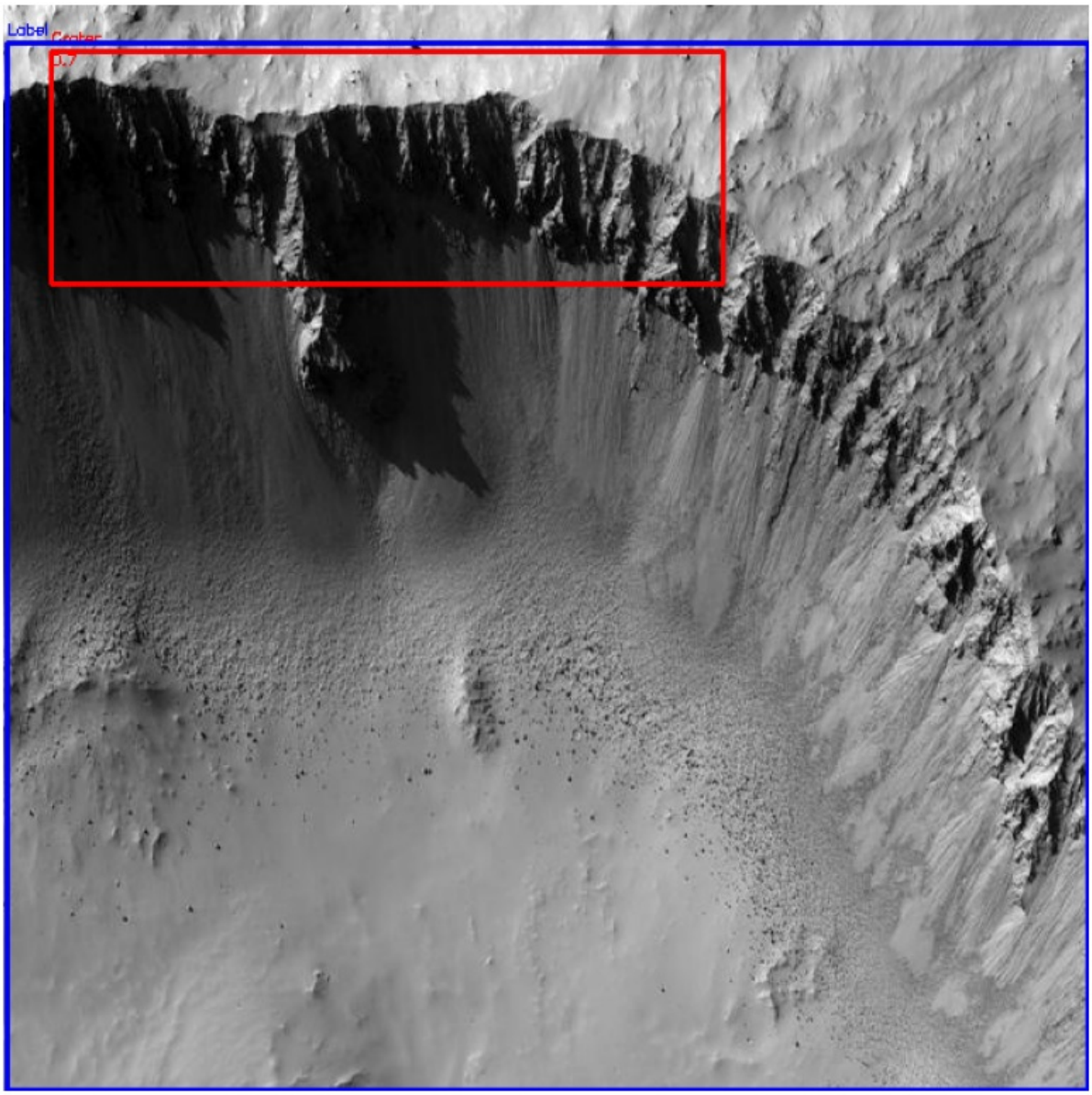


Image 13 done...

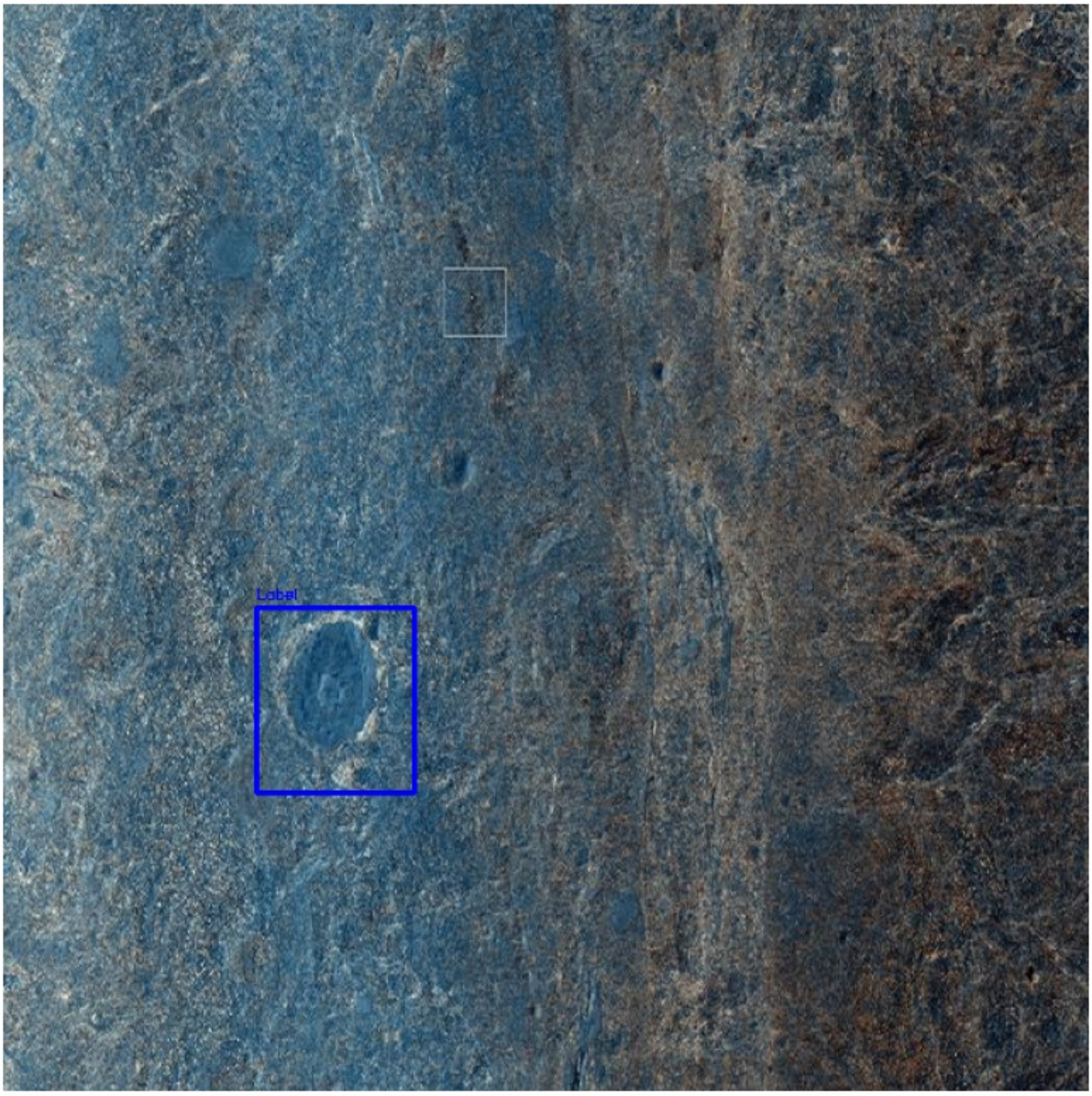


Image 14 done...

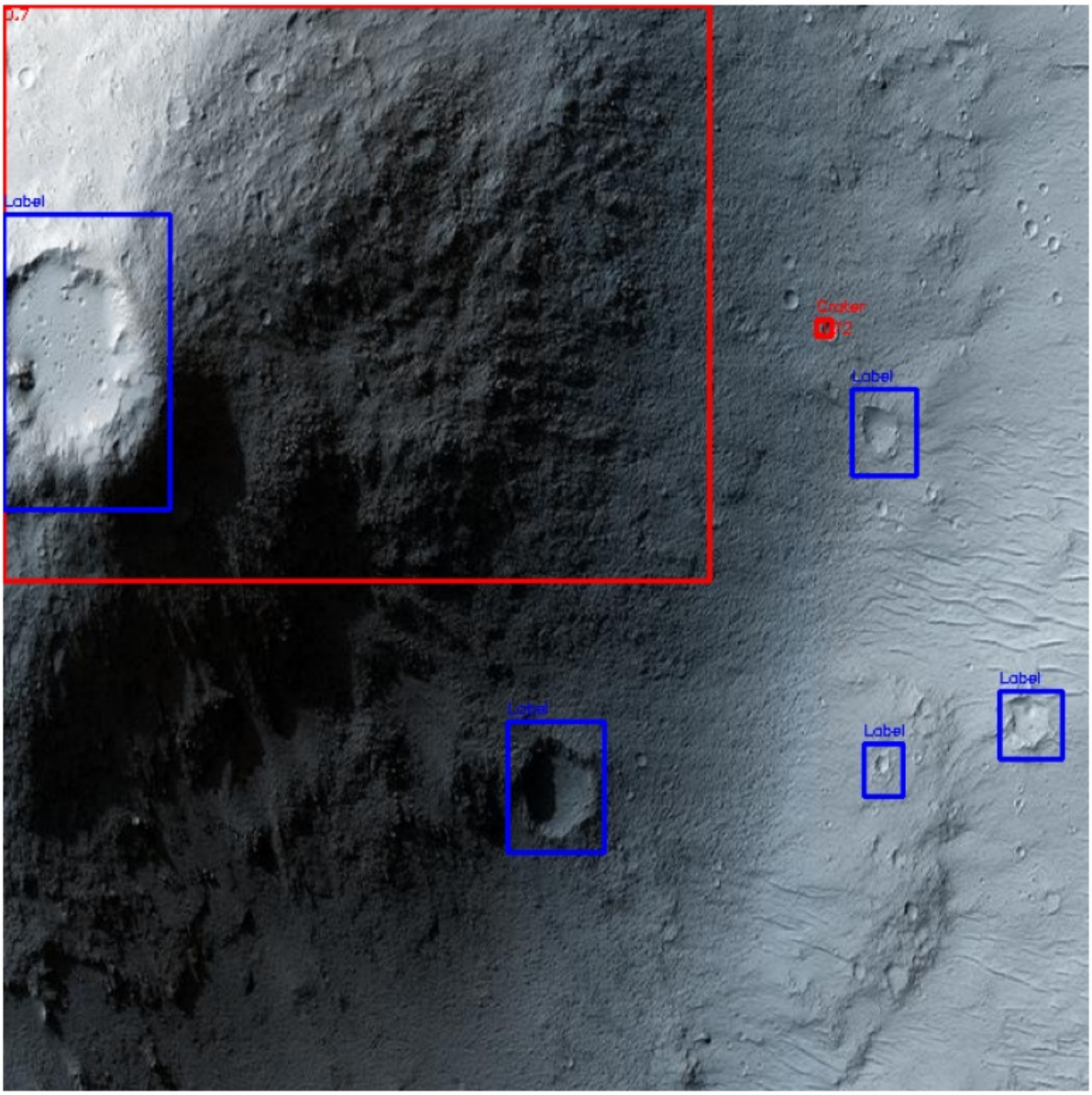


Image 15 done...

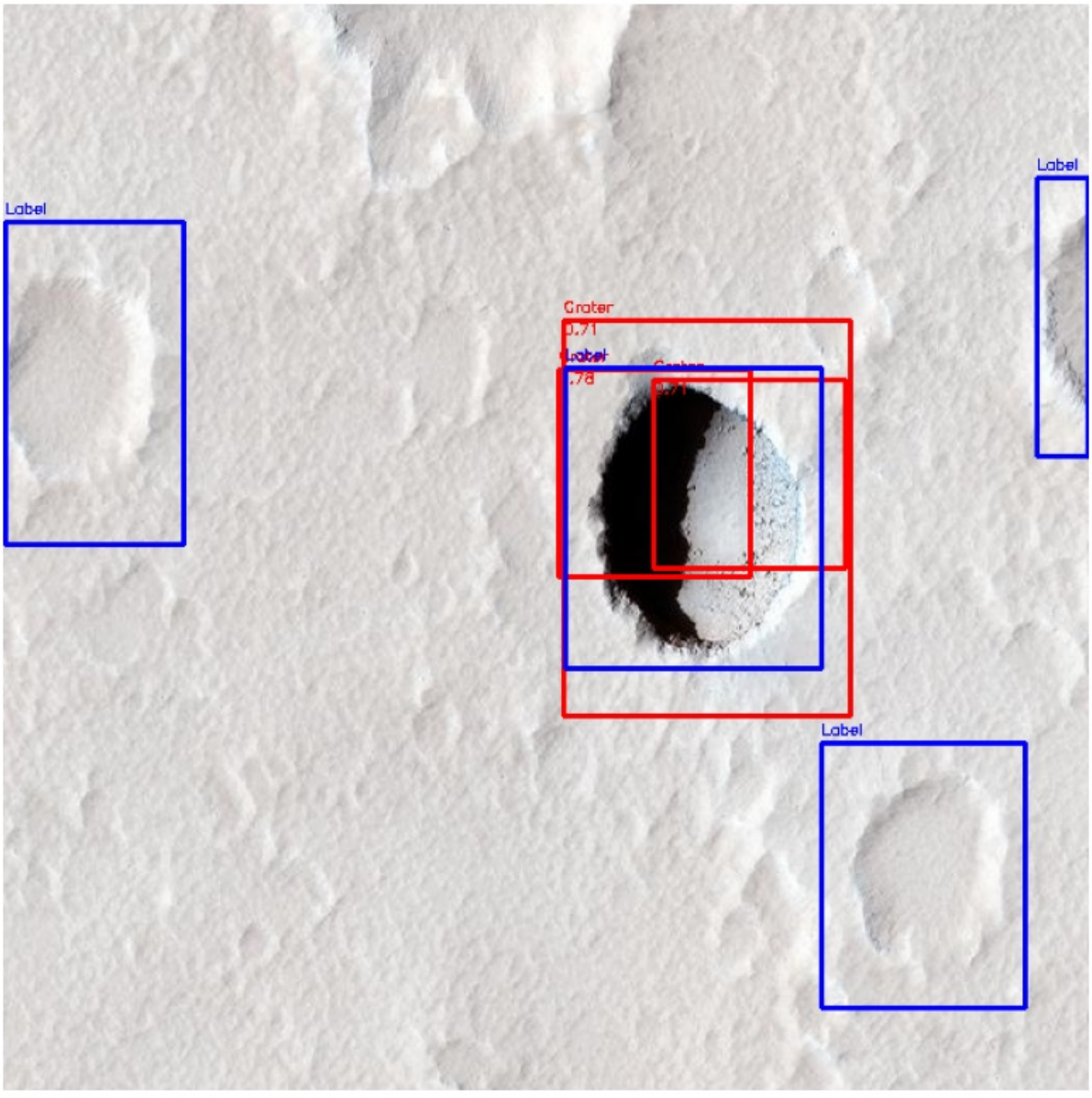


Image 16 done...

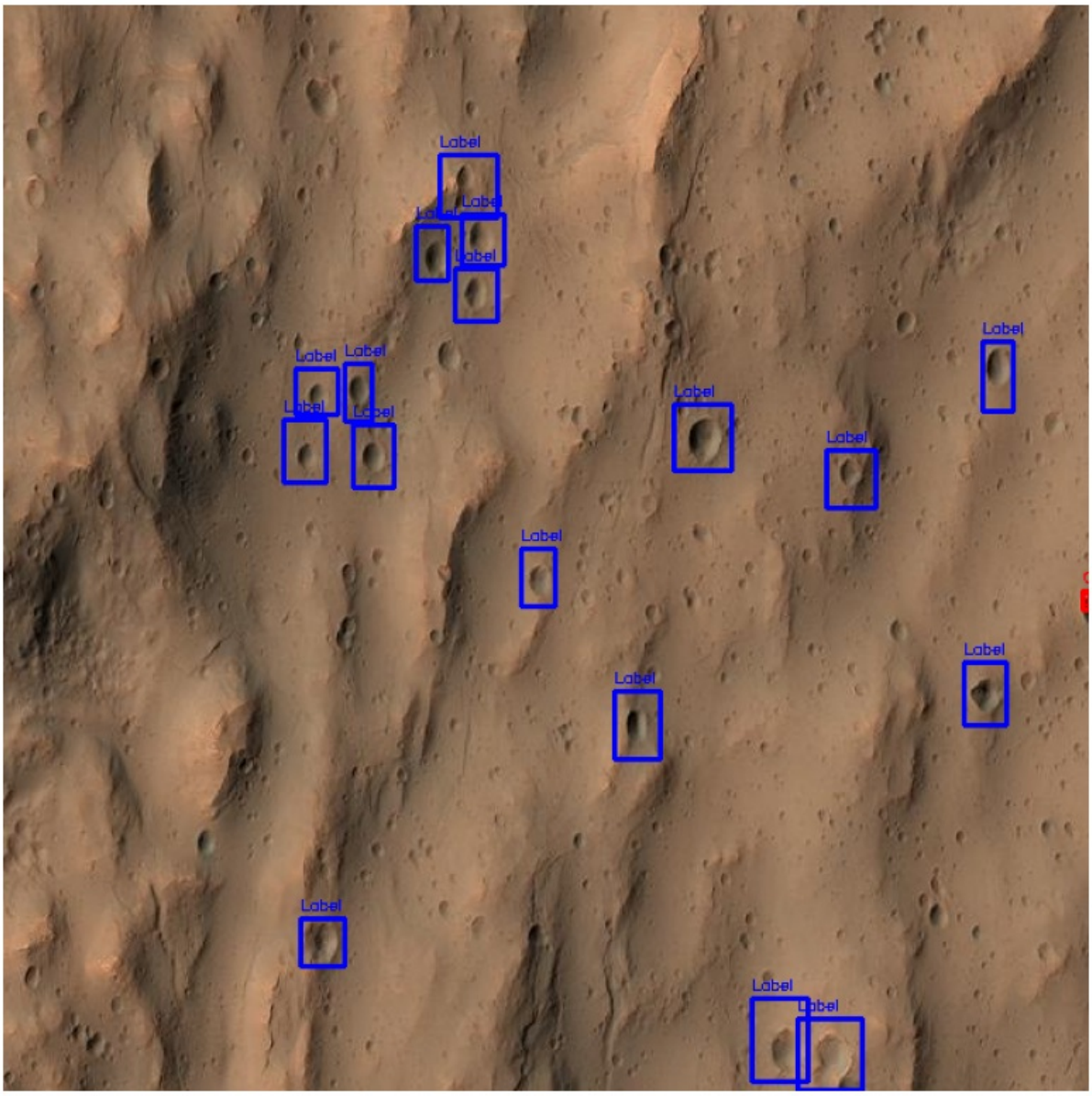
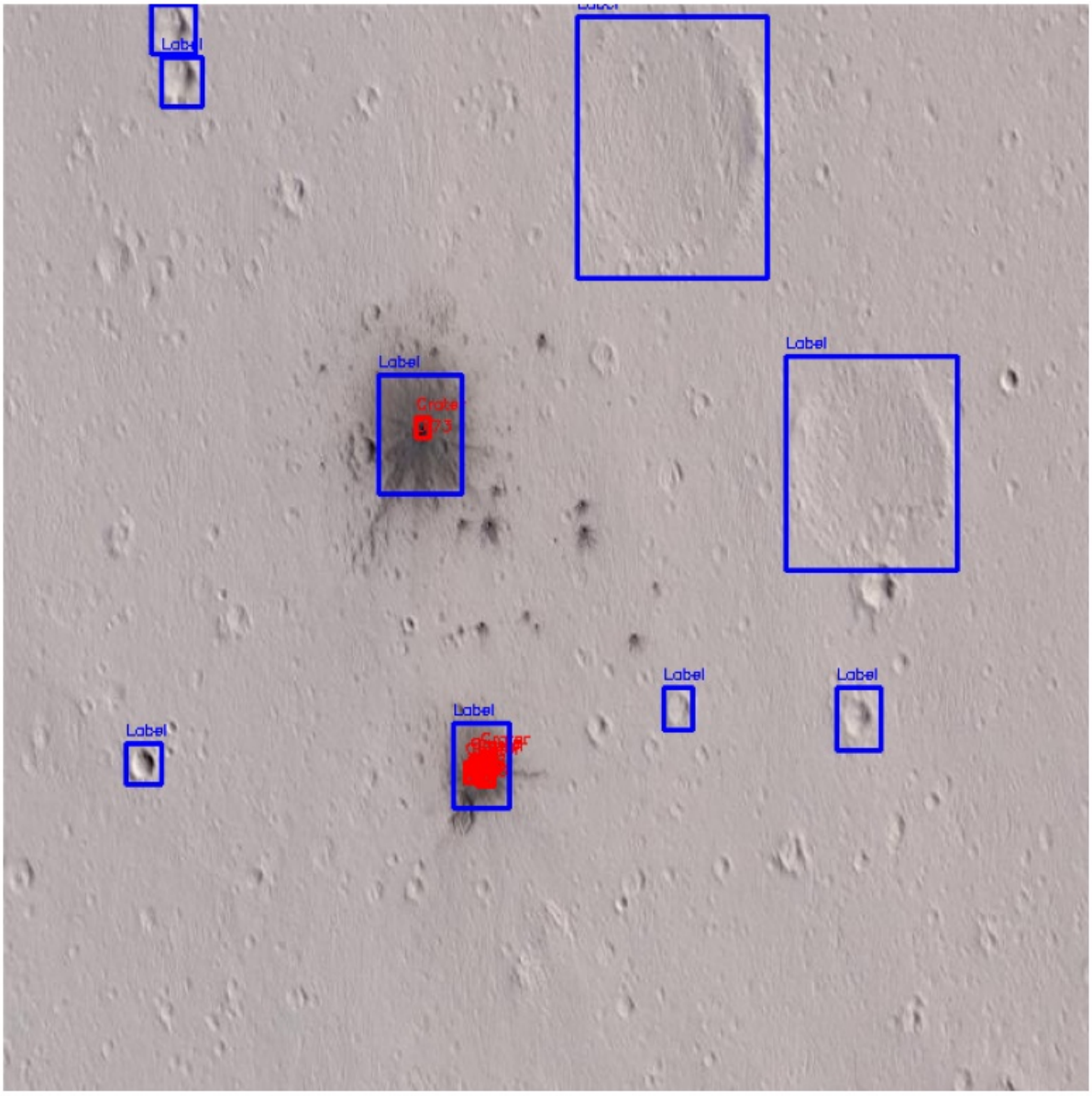
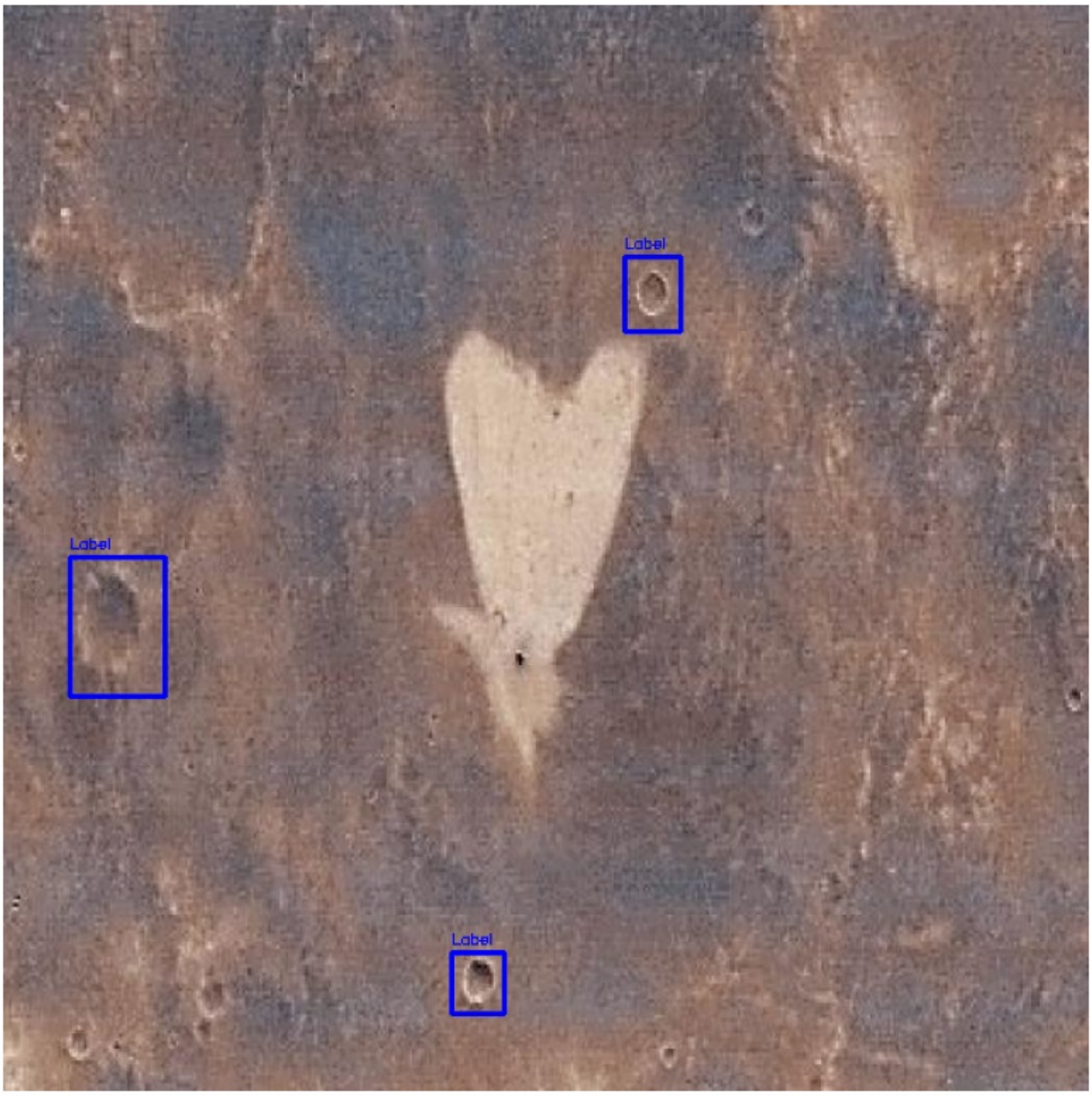


Image 17 done...




```
Image 18 done...
--------------------------------------------------
```




```
Image 19 done...
-------------------------------------------------
TEST PREDICTIONS COMPLETE
Average FPS: 9.393
```

Al principio el código no funcionaba, como no sabíamos qué podía ser, porque los espacios y márgenes estaban bien colocados, le preguntamos al ChatGPT y esto fue lo que dijo que estaba ocurriendo: "El error que estás viendo se debe a que hay un tipo de tensor incompatible entre el tensor de entrada y el tensor de peso. El tensor de entrada está en el dispositivo CUDA (GPU) y el tensor de peso está en la CPU. Para solucionar esto, debes asegurarte de que tanto el tensor de entrada como el tensor de peso estén en el mismo dispositivo. Puedes corregir el error moviendo el modelo y los datos al mismo dispositivo antes de realizar la inferencia." Esto se solucionó entonces agregando el siguiente código:

```
device = torch.device('cuda' if torch.cuda.is_available()else 'cpu')
```

```
model.to(device)
```

```
model_image = model_image.to(device)
```

**También hubo que añadir al principio del código model.eval(), para indicar que estamos haciendo inferencia. Al ejecutar el código nos aparecieron boxes por toda la imagen y no era preciso. Para solucionar esto subimos el número de épocas de 2 a 5, por eso en el código anterior decidimos poner 5 en el número de épocas.

Este fragmento de código se encarga de realizar inferencias en las imágenes de prueba utilizando el modelo previamente entrenado. Aquí se describen las principales acciones que realiza este fragmento:

1. Verifica la disponibilidad de CUDA y establece el dispositivo en función de la disponibilidad.
2. Mueve el modelo y los datos al dispositivo correspondiente.
3. Itera a través del `data_loader_test`, que contiene datos de prueba.

4. Recupera el nombre del archivo de imagen para el nombre del archivo de predicciones.
5. Realiza la inferencia en la imagen del modelo y mide el tiempo de inicio y finalización para calcular los cuadros por segundo.
6. Mueve los resultados de la inferencia al dispositivo de la CPU para operaciones posteriores.
7. Filtra las cajas delimitadoras basadas en un umbral de detección.
8. Dibuja las cajas delimitadoras y las etiquetas correspondientes en la imagen.
9. Muestra la imagen con las cajas delimitadoras y guarda la imagen resultante en el directorio de resultados.
10. Imprime el progreso del procesamiento de la imagen actual.
11. Al finalizar la iteración, se muestra el mensaje "TEST PREDICTIONS COMPLETE".
12. Calcula y muestra el promedio de cuadros por segundo.

No pudimos sacar una conclusión muy acertada de cómo de buena era la detección, ya que unas veces daba unos resultados y otras veces otro. Sin embargo, en todas ellas se conseguían detectar algunos de los cráteres etiquetados. La clave para mejorar la detección era subir el número de épocas y de Kfolds. Nosotros pusimos un número 'bajito' pero considerable para nuestro objetivo y recursos.

2. Modificar el código para comparar algoritmo.

- Cambiar el modelo de detección propuesto por otro de los que se encuentran en `torchvision.models.detection`.
- Cambiar la función nueva `get_model_bbox_alternativo()` que sea igual a `get_model_bbox()` salvo cambiando el modelo a entrenar.
- Entrenar el modelo y hacer predicciones. Dichas predicciones deberán ir pintadas en verde y se representarán de manera conjunta con el ground truth y el resultado del modelo que viene en el notebook original.

Definir una función get_model_bbox_alternativo que utiliza un modelo RetinaNet:

In [ ]:
```
def get_model_bbox_alternativo(num_classes):
    # load an instance segmentation model pre-trained on COCO
    model = torchvision.models.detection.retinanet_resnet50_fpn(pretrained=True)

    # get number of input features for the classifier
    # No hay atributos roi:
    # in_features = model.roi_heads.box_predictor.cls_score.in_features
    # replace the pre-trained head with a new one
    # model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    return model
```

Importante ejecutar el siguiente código para liberar espacio en la GPU, ya que en este punto se suele encontrar ya bastante saturado por la ejecución del código anterior. También para evitar que variables sueltas del código anterior nos afecten en nuestro nuevo modelo de detección.

In [ ]:
```
torch.cuda.empty_cache()
```

K-FOLD CROSS VALIDATION:

In [ ]:
```
# train on the GPU or on the CPU, if a GPU is not available
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
k_folds = 7
num_epochs = 5

# our dataset has two classes only - background and crater
num_classes = 2
# use our dataset and defined transformations
dataset = CraterDataset('/content/craters/train', get_transform(train=True))
dataset_val = CraterDataset('/content/craters/train', get_transform(train=False))

# Define the K-fold Cross Validator
kfold = KFold(n_splits=k_folds, shuffle=True)

# Start print
print('--------------------------------')

# K-fold Cross Validation model evaluation
for fold, (train_ids, val_ids) in enumerate(kfold.split(dataset)):
    print(f'FOLD {fold}')
    print('--------------------------------')

    dataset_subset = torch.utils.data.Subset(dataset, list(train_ids))
    dataset_val_subset = torch.utils.data.Subset(dataset_val, list(val_ids))

    # get the model using our helper function
    model = get_model_bbox_alternativo(num_classes)


    # move model to the right device
    model.to(device)
```

```
    # construct an optimizer
    params = [p for p in model.parameters() if p.requires_grad]
    optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9, weight_decay=0)

    # and a learning rate scheduler
    lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)

    # let's train!
    for epoch in range(num_epochs):
        # train for one epoch, printing every 50 iterations
        train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=50)
        # update the learning rate
        lr_scheduler.step()
```

```
--------------------------------
FOLD 0
--------------------------------
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretraine
d' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a w
eight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current beha
vior is equivalent to passing `weights=RetinaNet_ResNet50_FPN_Weights.COCO_V1`. You can also use `weights=Retin
aNet_ResNet50_FPN_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
Epoch: [0]  [ 0/13]  eta: 0:00:43  lr: 0.000421  loss: 1.7827 (1.7827)  classification: 1.4669 (1.4669)  bbox_r
egression: 0.3158 (0.3158)  time: 3.3760  data: 1.2385  max mem: 7367
Epoch: [0]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 1.2421 (1.2495)  classification: 0.8229 (0.8754)  bbox_r
egression: 0.3798 (0.3742)  time: 1.8675  data: 0.1447  max mem: 7367
Epoch: [0] Total time: 0:00:24 (1.8728 s / it)
Epoch: [1]  [ 0/13]  eta: 0:00:28  lr: 0.005000  loss: 1.1102 (1.1102)  classification: 0.7975 (0.7975)  bbox_r
egression: 0.3126 (0.3126)  time: 2.1835  data: 0.4187  max mem: 7367
Epoch: [1]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.8921 (0.9498)  classification: 0.6229 (0.6326)  bbox_r
egression: 0.3126 (0.3172)  time: 1.7340  data: 0.0641  max mem: 7367
Epoch: [1] Total time: 0:00:22 (1.7394 s / it)
Epoch: [2]  [ 0/13]  eta: 0:00:27  lr: 0.005000  loss: 0.8032 (0.8032)  classification: 0.5753 (0.5753)  bbox_r
egression: 0.2279 (0.2279)  time: 2.1355  data: 0.3776  max mem: 7367
Epoch: [2]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.5741 (0.5893)  classification: 0.3385 (0.3451)  bbox_r
egression: 0.2376 (0.2442)  time: 1.7028  data: 0.0584  max mem: 7367
Epoch: [2] Total time: 0:00:22 (1.7085 s / it)
FOLD 1
--------------------------------
Epoch: [0]  [ 0/13]  eta: 0:00:28  lr: 0.000421  loss: 1.8773 (1.8773)  classification: 1.5742 (1.5742)  bbox_r
egression: 0.3031 (0.3031)  time: 2.1673  data: 0.4076  max mem: 7367
Epoch: [0]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 1.0875 (1.2161)  classification: 0.6419 (0.8577)  bbox_r
egression: 0.3530 (0.3584)  time: 1.7129  data: 0.0639  max mem: 7367
Epoch: [0] Total time: 0:00:22 (1.7184 s / it)
Epoch: [1]  [ 0/13]  eta: 0:00:28  lr: 0.005000  loss: 0.7869 (0.7869)  classification: 0.5110 (0.5110)  bbox_r
egression: 0.2759 (0.2759)  time: 2.1834  data: 0.4103  max mem: 7367
Epoch: [1]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.7562 (0.7463)  classification: 0.5108 (0.4733)  bbox_r
egression: 0.2718 (0.2729)  time: 1.7434  data: 0.0700  max mem: 7367
Epoch: [1] Total time: 0:00:22 (1.7562 s / it)
Epoch: [2]  [ 0/13]  eta: 0:00:29  lr: 0.005000  loss: 0.6031 (0.6031)  classification: 0.3656 (0.3656)  bbox_r
egression: 0.2375 (0.2375)  time: 2.2582  data: 0.4714  max mem: 7367
Epoch: [2]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.5721 (0.5854)  classification: 0.3177 (0.3376)  bbox_r
egression: 0.2375 (0.2478)  time: 1.7871  data: 0.0905  max mem: 7367
Epoch: [2] Total time: 0:00:23 (1.7979 s / it)
```

ENTRENAMIENTO:

In [ ]:

```
num_epochs = 5

# our dataset has two classes only - background and crater
num_classes = 2
# use our dataset and defined transformations
dataset = CraterDataset('/content/craters/train', get_transform(train=True))
dataset_test = CraterDataset('/content/craters/test', get_transform(train=False))

# define training and validation data loaders
data_loader = torch.utils.data.DataLoader(
        dataset, batch_size=8, shuffle=True, num_workers=2,
    collate_fn=utils.collate_fn)

data_loader_test = torch.utils.data.DataLoader(
    dataset_test, batch_size=1, shuffle=False, num_workers=2,
    collate_fn=utils.collate_fn)

# get the model using our helper function
model = get_model_bbox_alternativo(num_classes)

'''
Use this to reset all trainable weights
model.apply(reset_weights)
'''

# move model to the right device
model.to(device)
```

```
# construct an optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005,  # Feel free to play with values
                            momentum=0.9, weight_decay=0)

# Defining learning rate scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                               step_size=20,
                                               gamma=0.2)


result_mAP = []
best_epoch = None

# Let's train!
for epoch in range(num_epochs):


    # train for one epoch, printing every 10 iterations
    train_one_epoch(model, optimizer, data_loader, device, epoch, print_freq=50)
    # update the learning rate
    lr_scheduler.step()

# Saving the last model
save_path = os.path.join(f'Crater_noaug_sgd_2batch-lastepoch{num_epochs-1}.pth')
torch.save(model.state_dict(), save_path)
print(f'model from last epoch(no.{num_epochs-1}) saved')
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretraine
d' is deprecated since 0.13 and may be removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a w
eight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current beha
vior is equivalent to passing `weights=RetinaNet_ResNet50_FPN_Weights.COCO_V1`. You can also use `weights=Retin
aNet_ResNet50_FPN_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
Epoch: [0]  [ 0/13]  eta: 0:00:31  lr: 0.000421  loss: 2.0497 (2.0497)  classification: 1.6200 (1.6200)  bbox_r
egression: 0.4297 (0.4297)  time: 2.4193  data: 0.5923  max mem: 7367
Epoch: [0]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 1.2861 (1.3469)  classification: 0.8943 (0.9668)  bbox_r
egression: 0.3688 (0.3801)  time: 1.7745  data: 0.0756  max mem: 7367
Epoch: [0] Total time: 0:00:23 (1.7846 s / it)
Epoch: [1]  [ 0/13]  eta: 0:00:33  lr: 0.005000  loss: 0.9292 (0.9292)  classification: 0.6711 (0.6711)  bbox_r
egression: 0.2580 (0.2580)  time: 2.6043  data: 0.7565  max mem: 7367
Epoch: [1]  [12/13]  eta: 0:00:01  lr: 0.005000  loss: 0.9569 (1.1988)  classification: 0.6711 (0.8899)  bbox_r
egression: 0.3057 (0.3090)  time: 1.7696  data: 0.0886  max mem: 7367
Epoch: [1] Total time: 0:00:23 (1.7753 s / it)
model from last epoch(no.1) saved
```

CARACTERÍSTICAS DE LAS VISUALIZACIONES (color...):

In [ ]:
```
# Define colors for bounding boxes
color_inference = np.array([0.0,255.0,0.0])
color_label = np.array([255.0,0.0,0.0])

# Score value thershold for displaying predictions
detection_threshold = 0.7
# to count the total number of images iterated through
frame_count = 0
# to keep adding the FPS for each image
total_fps = 0

!mkdir ./results
```

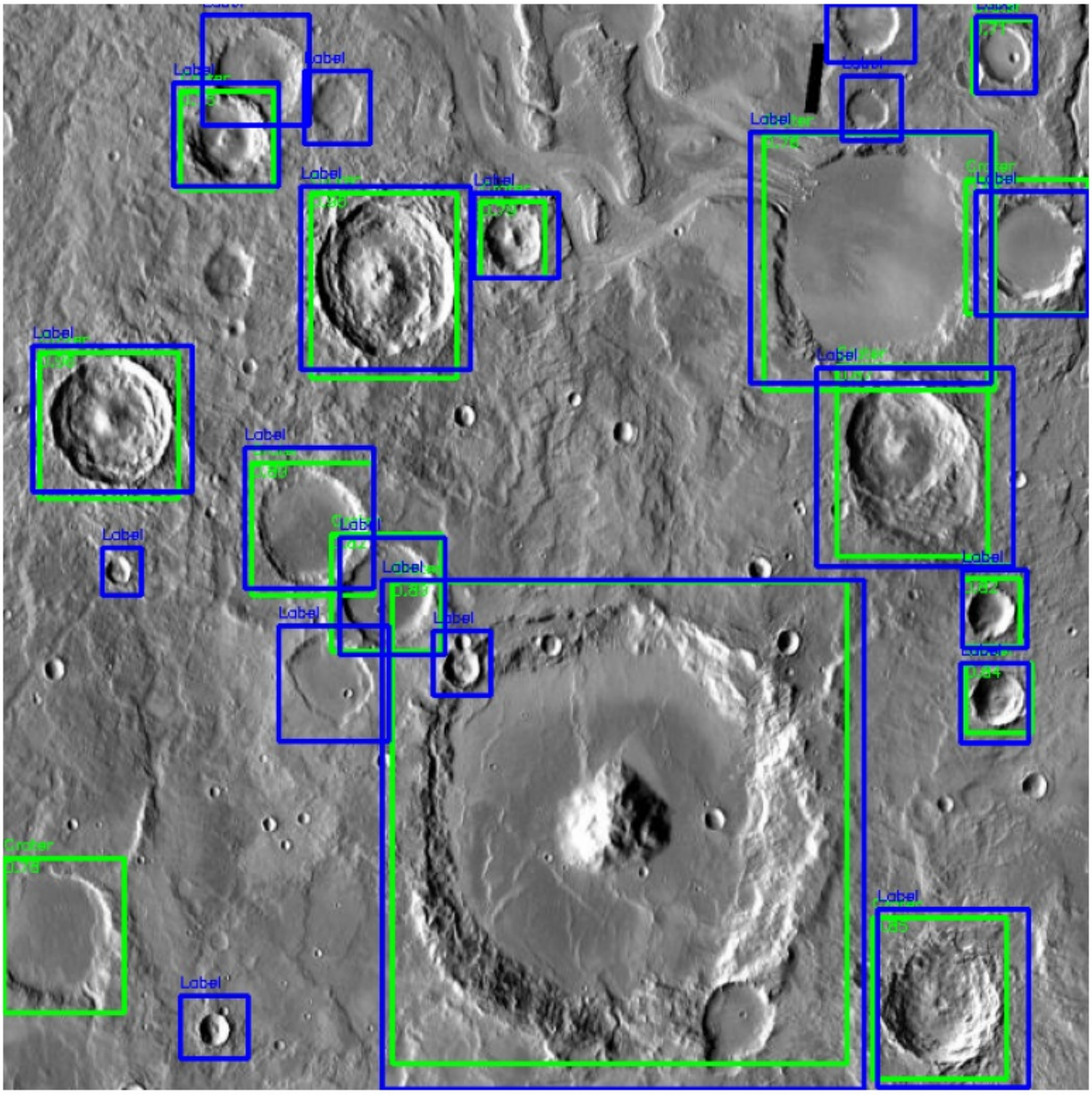
```
mkdir: cannot create directory ‘./results’: File exists
```

Queremos que el color de las predicciones sea verde, para ello cambiamos color_inference = np.array([0.0,0.0,255.0]) a color_inference = np.array([0.0,255.0,0.0])

INFERENCIAS Y PREDICCIONES DEL MODELO ENTRENADO

In [ ]:
```
model.eval()
# Verificar la disponibilidad de CUDA y configurar el dispositivo
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Mover el modelo y los datos al dispositivo correspondiente
model.to(device)
model_image = model.to(device)

for i,data in enumerate(data_loader_test):
    # get the image file name for predictions file name
    image_name = 'image no:' + str(int(data[1][0]['image_id']))
    model_image = data[0][0]
    cv2_image = np.transpose(model_image.numpy()*255,(1, 2, 0)).astype(np.float32)
    cv2_image = cv2.cvtColor(cv2_image, cv2.COLOR_RGB2BGR).astype(np.float32)

    # add batch dimension
```

```
    model_image = torch.unsqueeze(model_image, 0)
    start_time = time.time()
    with torch.no_grad():
      outputs = model(model_image.to(device))
      end_time = time.time()
    # get the current fps
    fps = 1 / (end_time - start_time)
    # add `fps` to `total_fps`
    total_fps += fps
    # increment frame count
    frame_count += 1
    # load all detection to CPU for further operations
    outputs = [{k: v.to('cpu') for k, v in t.items()} for t in outputs]
    # carry further only if there's detected boxes
    if len(outputs[0]['boxes']) != 0:
        boxes = outputs[0]['boxes'].data.numpy()
        scores = outputs[0]['scores'].data.numpy()
        # filter out boxes according to `detection_threshold`
        boxes = boxes[scores >= detection_threshold].astype(np.int32)
        scores = np.round(scores[scores >= detection_threshold],2)
        draw_boxes = boxes.copy()


        # draw the bounding boxes and write the class name on top of it
        for j,box in enumerate(draw_boxes):
            cv2.rectangle(cv2_image,
                          (int(box[0]), int(box[1])),
                          (int(box[2]), int(box[3])),
                          color_inference, 2)
            cv2.putText(img=cv2_image, text="Crater",
                        org=(int(box[0]), int(box[1] - 5)),
                        fontFace=cv2.FONT_HERSHEY_SIMPLEX,fontScale= 0.3,color= color_inference,
                        thickness=1, lineType=cv2.LINE_AA)
            cv2.putText(img=cv2_image, text=str(scores[j]),
                        org=(int(box[0]), int(box[1] + 8)),
                        fontFace=cv2.FONT_HERSHEY_SIMPLEX,fontScale= 0.3,color= color_inference,
                        thickness=1, lineType=cv2.LINE_AA)

        # add boxes for labels
        for box in data[1][0]['boxes']:
            cv2.rectangle(cv2_image,
                          (int(box[0]), int(box[1])),
                          (int(box[2]), int(box[3])),
                          color_label, 2)
            cv2.putText(img=cv2_image, text="Label",
                        org=(int(box[0]), int(box[1] - 5)),
                        fontFace=cv2.FONT_HERSHEY_SIMPLEX,fontScale= 0.3,color= color_label,
                        thickness=1, lineType=cv2.LINE_AA)


        # set size
        plt.figure(figsize=(10,10))
        plt.axis("off")

        # convert color from CV2 BGR back to RGB
        plt_image = cv2.cvtColor(cv2_image/255.0, cv2.COLOR_BGR2RGB)
        plt.imshow(plt_image)
        plt.show()
        cv2.imwrite(f"./results/{image_name}.jpg", cv2_image)
    print(f"Image {i + 1} done...")
    print('-' * 50)
print('TEST PREDICTIONS COMPLETE')

avg_fps = total_fps / frame_count
print(f"Average FPS: {avg_fps:.3f}")
```




```
Image 1 done...
--------------------------------------------------
```

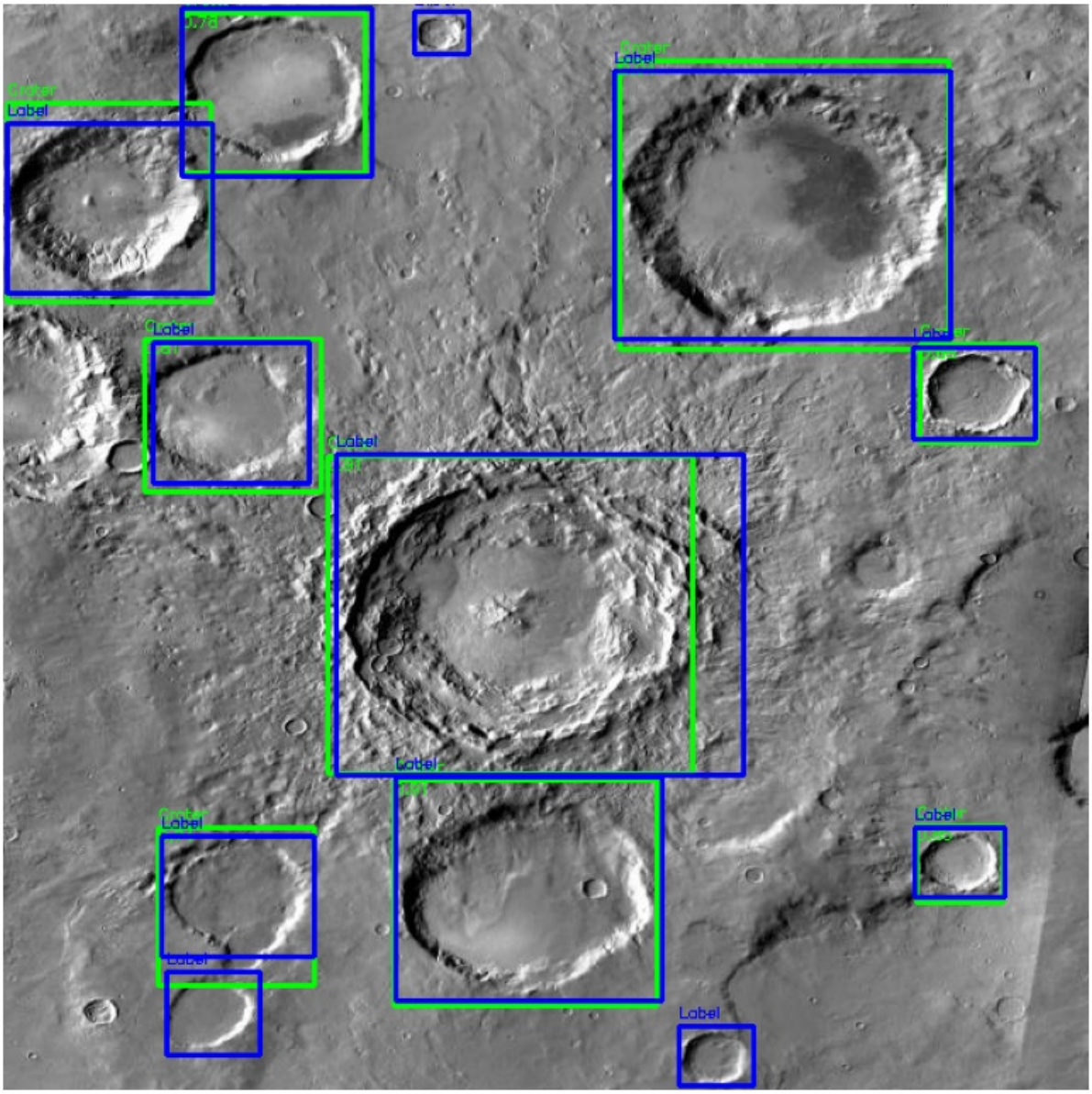


Image 2 done...

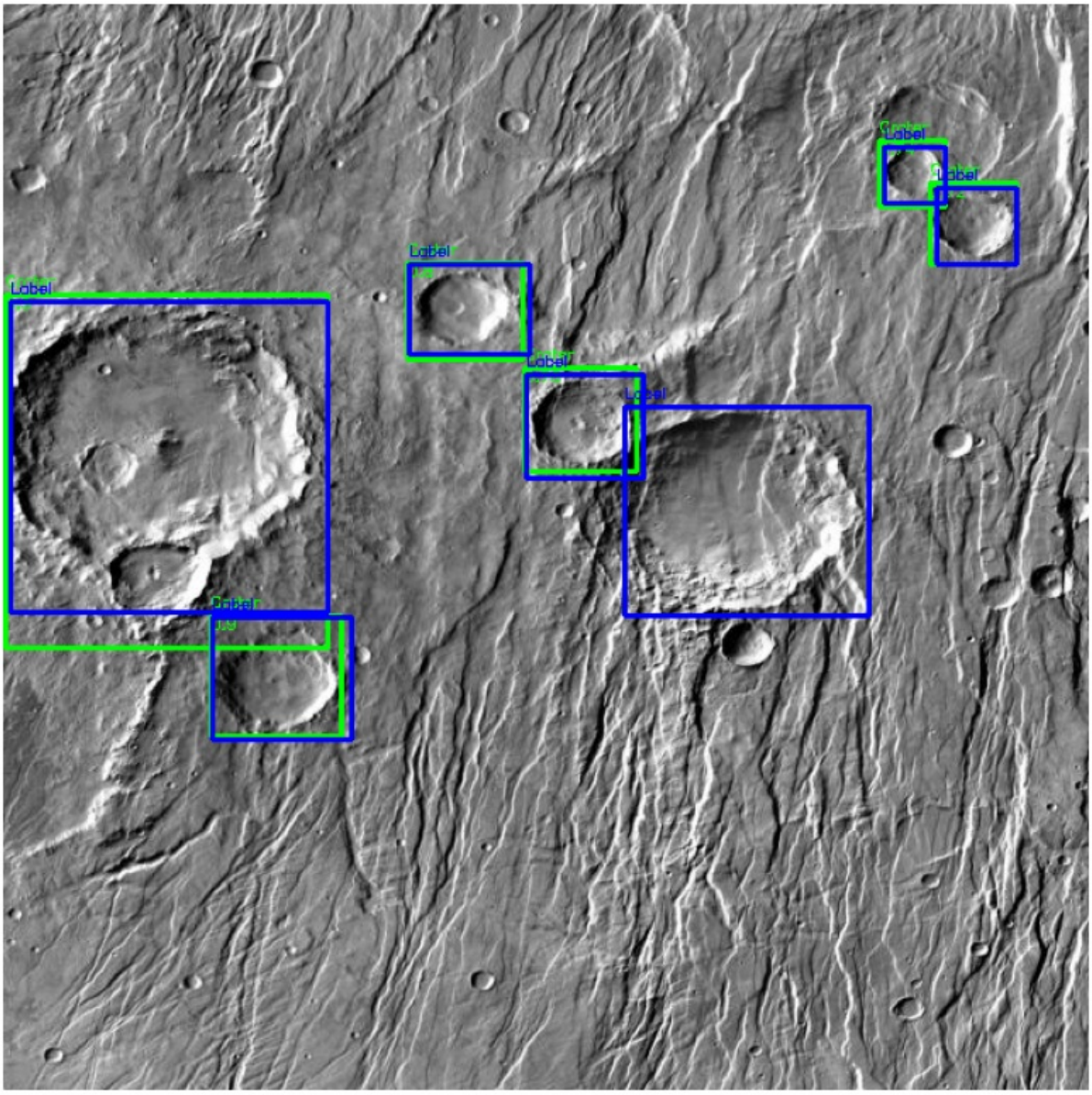


Image 3 done...

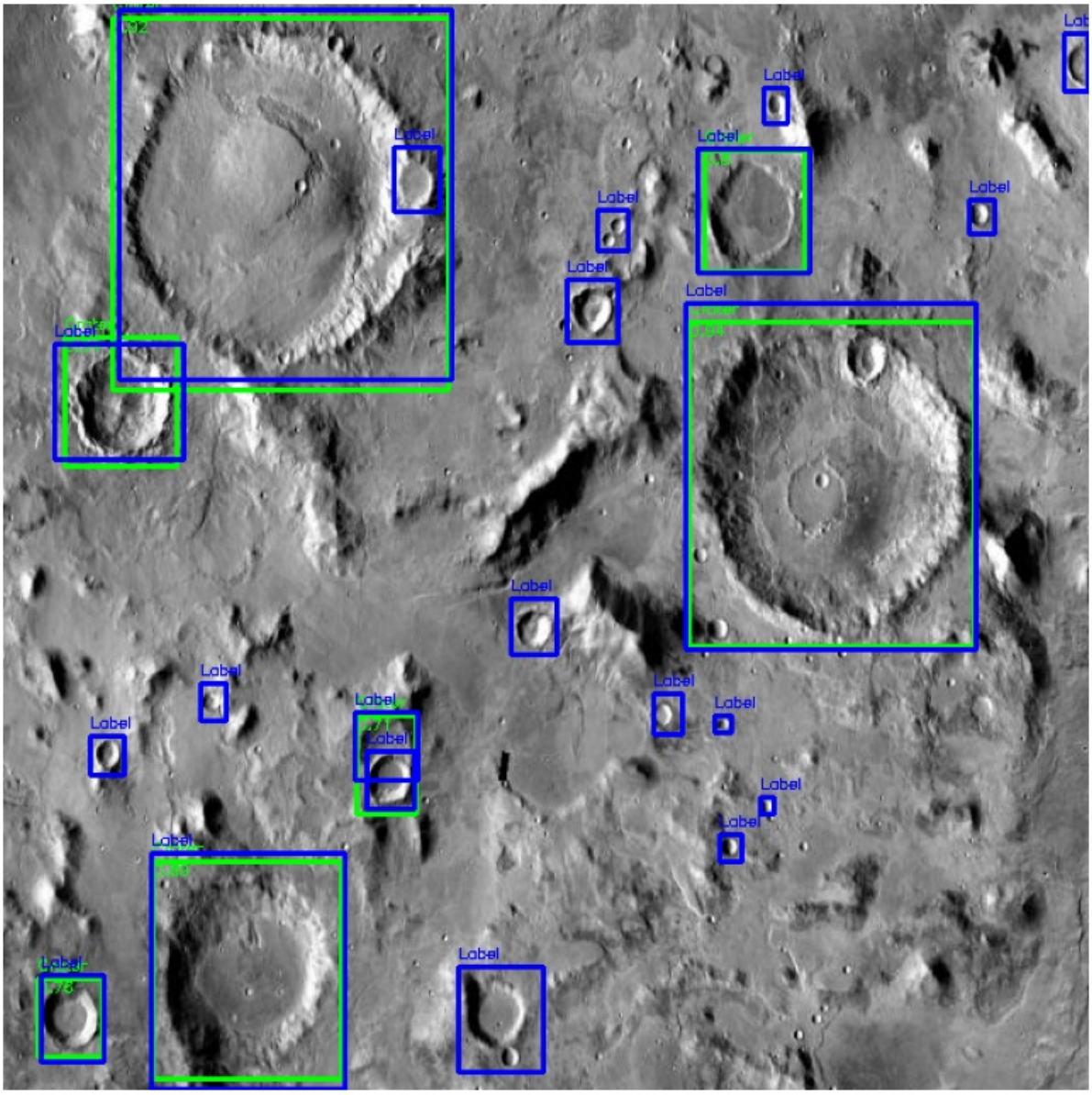

Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label
Label

Image 4 done...

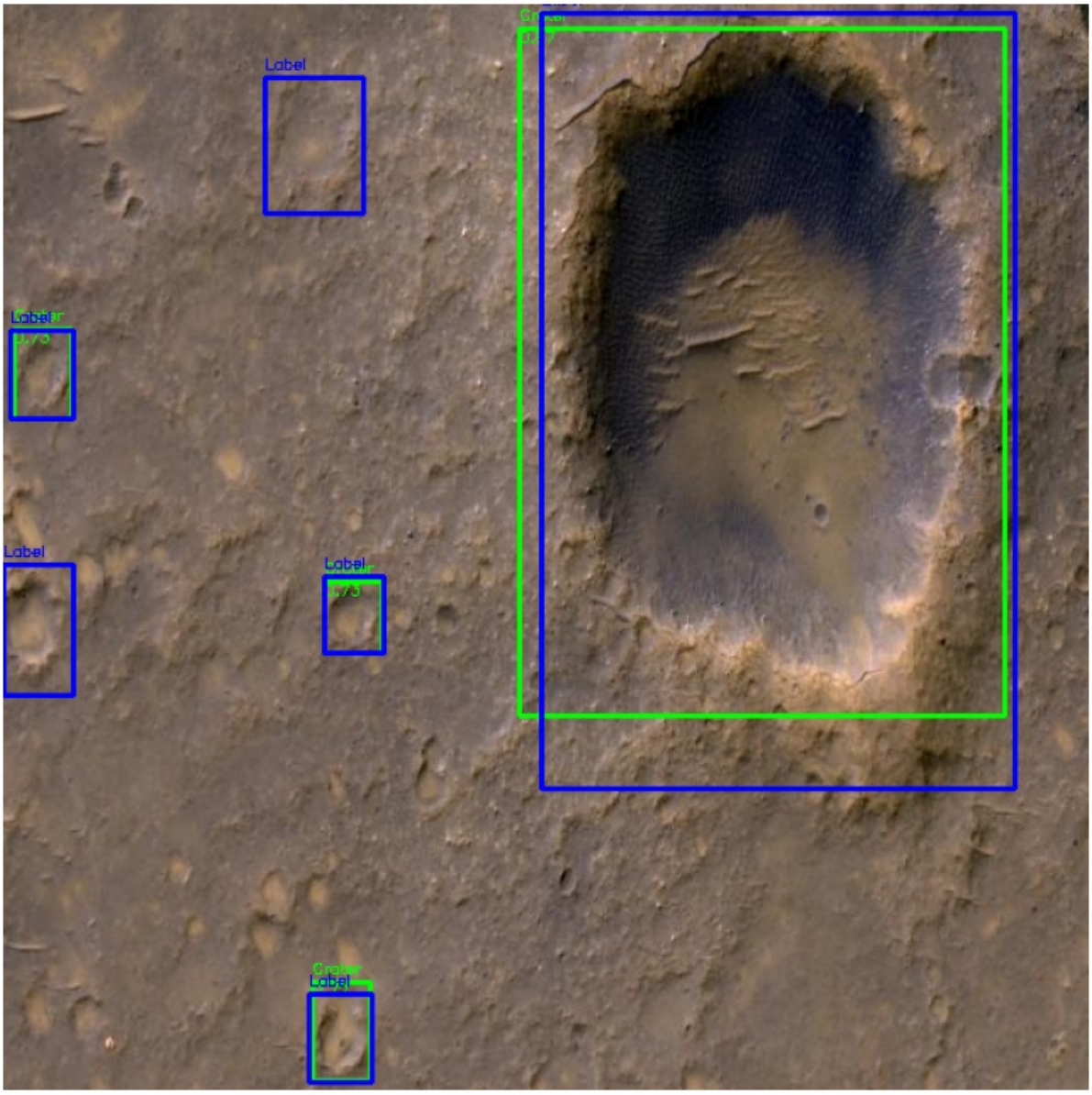


Image 5 done...

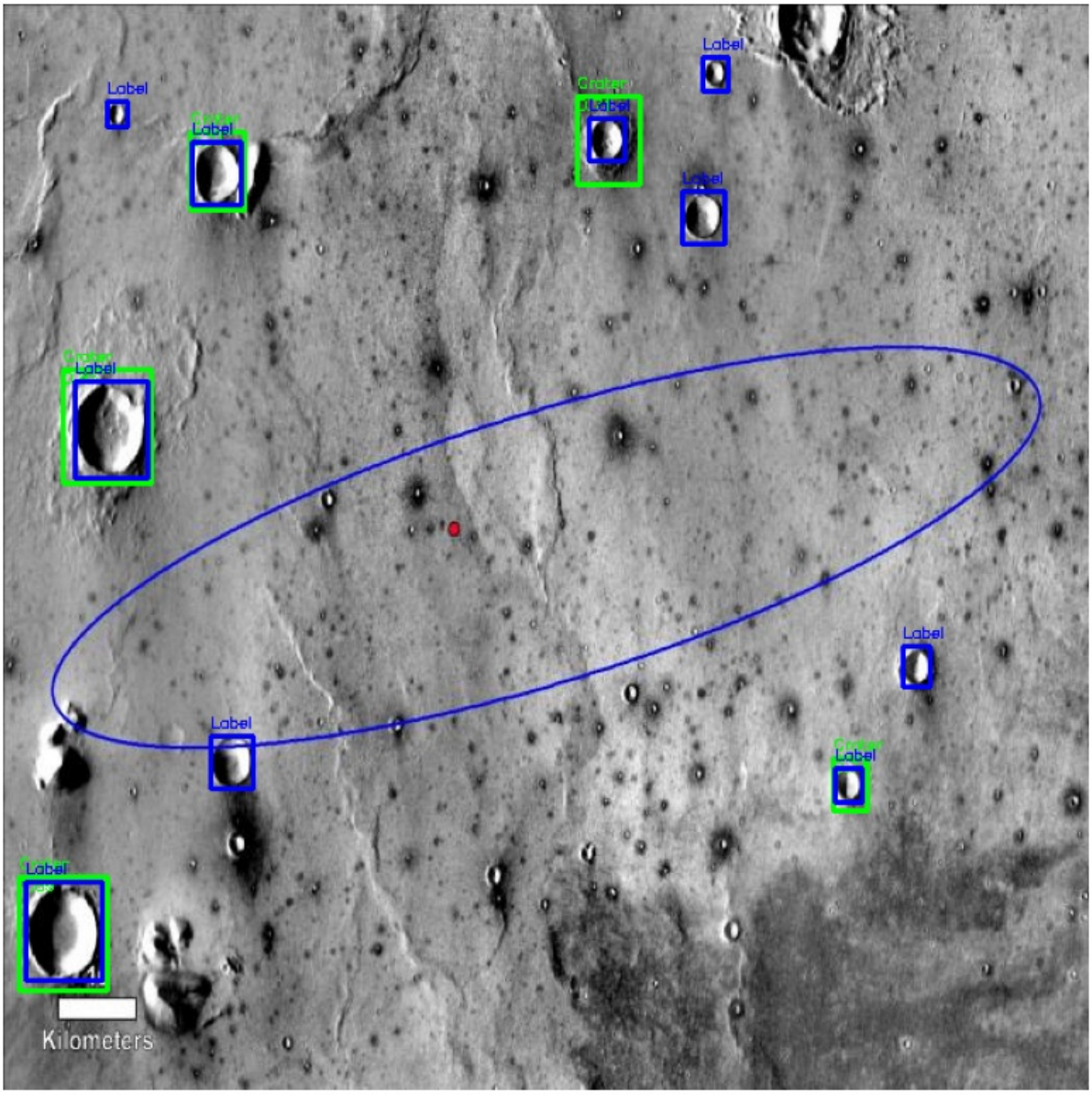
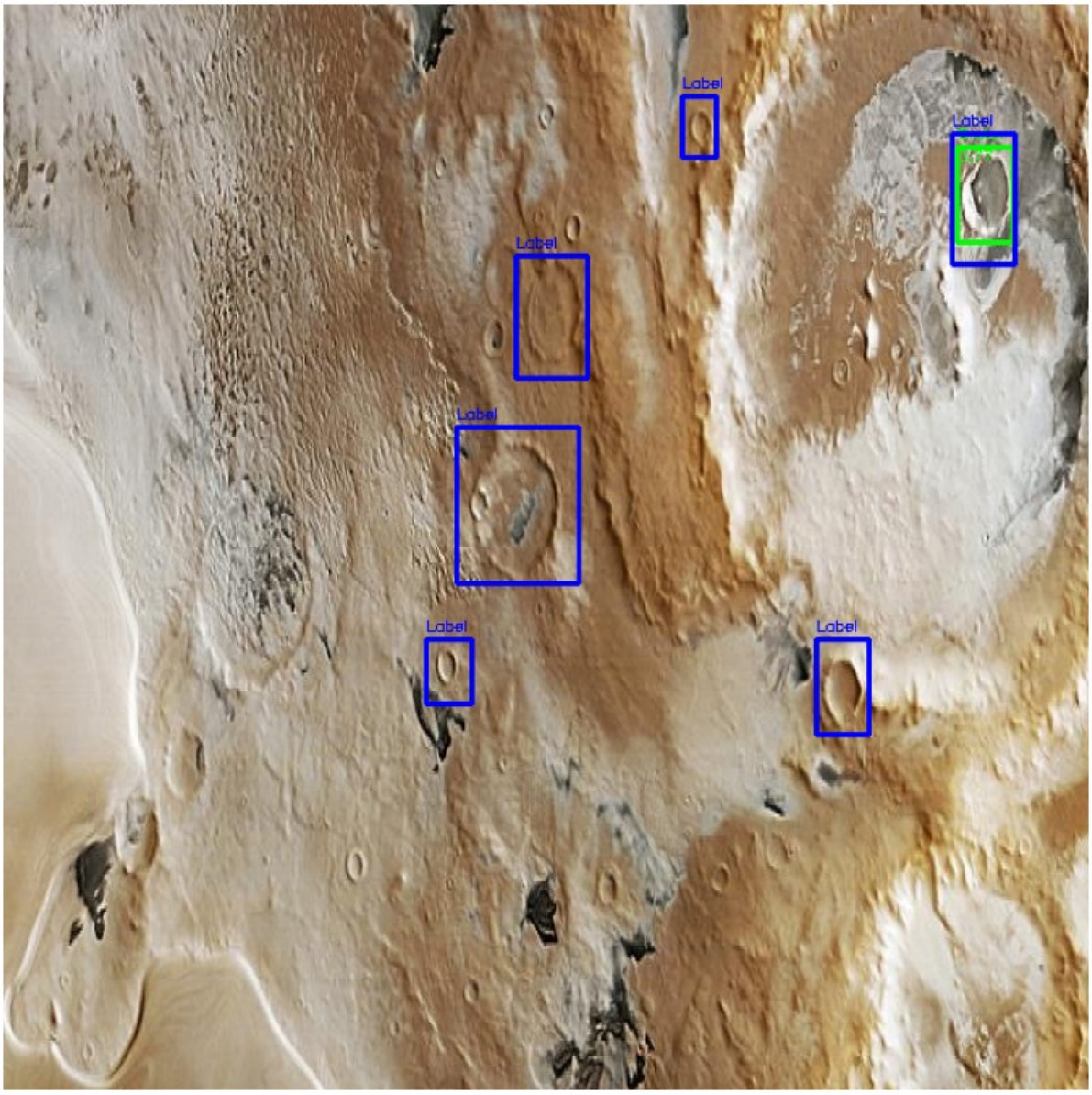

Label
Label
Crater
Label
Label
Label
Crater
Label
Label
Label
Crater
Label
Label
Crater
Label
Kilometers

Image 6 done...




```
Image 7 done...
--------------------------------------------------
```

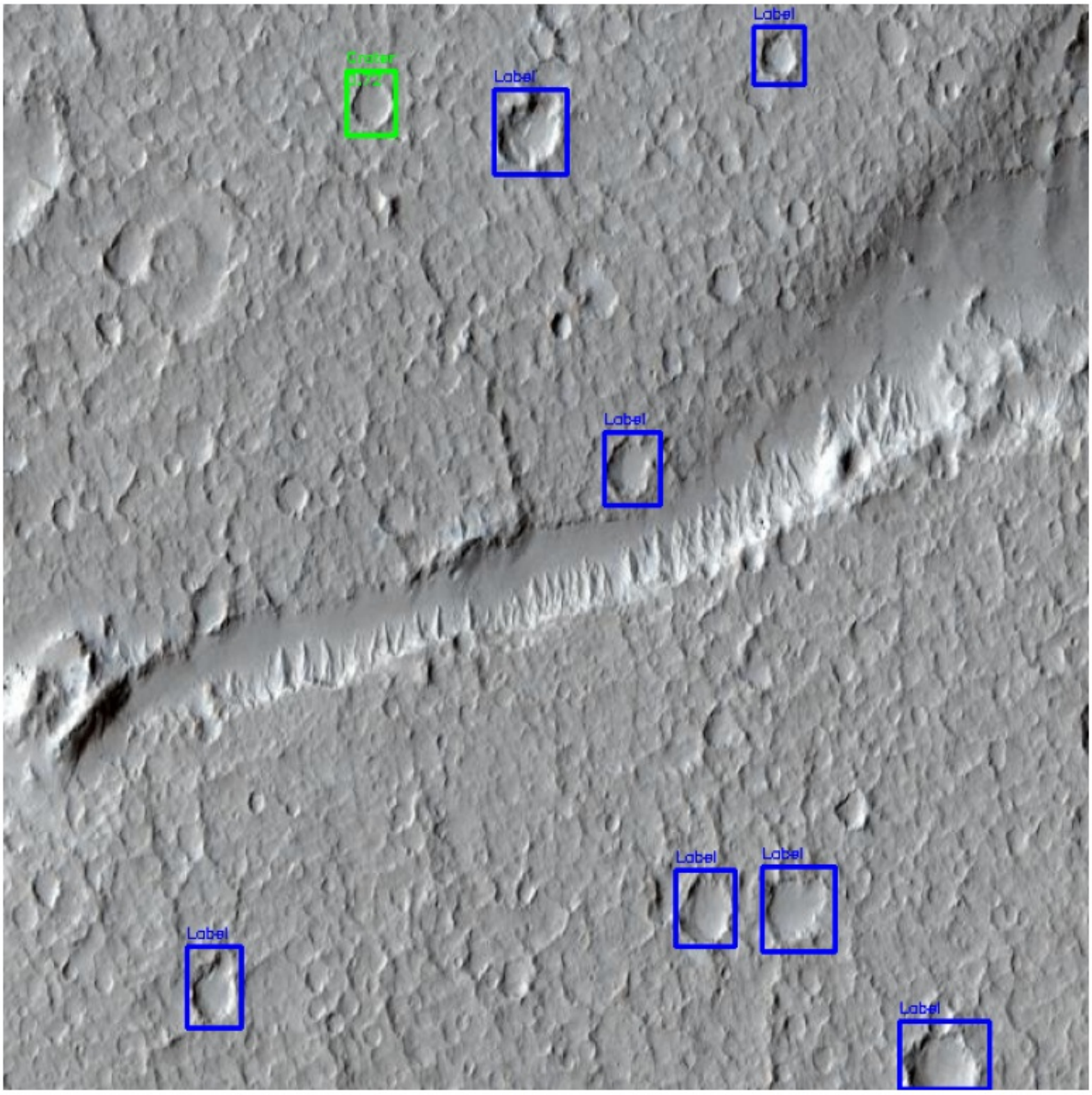


Image 8 done...

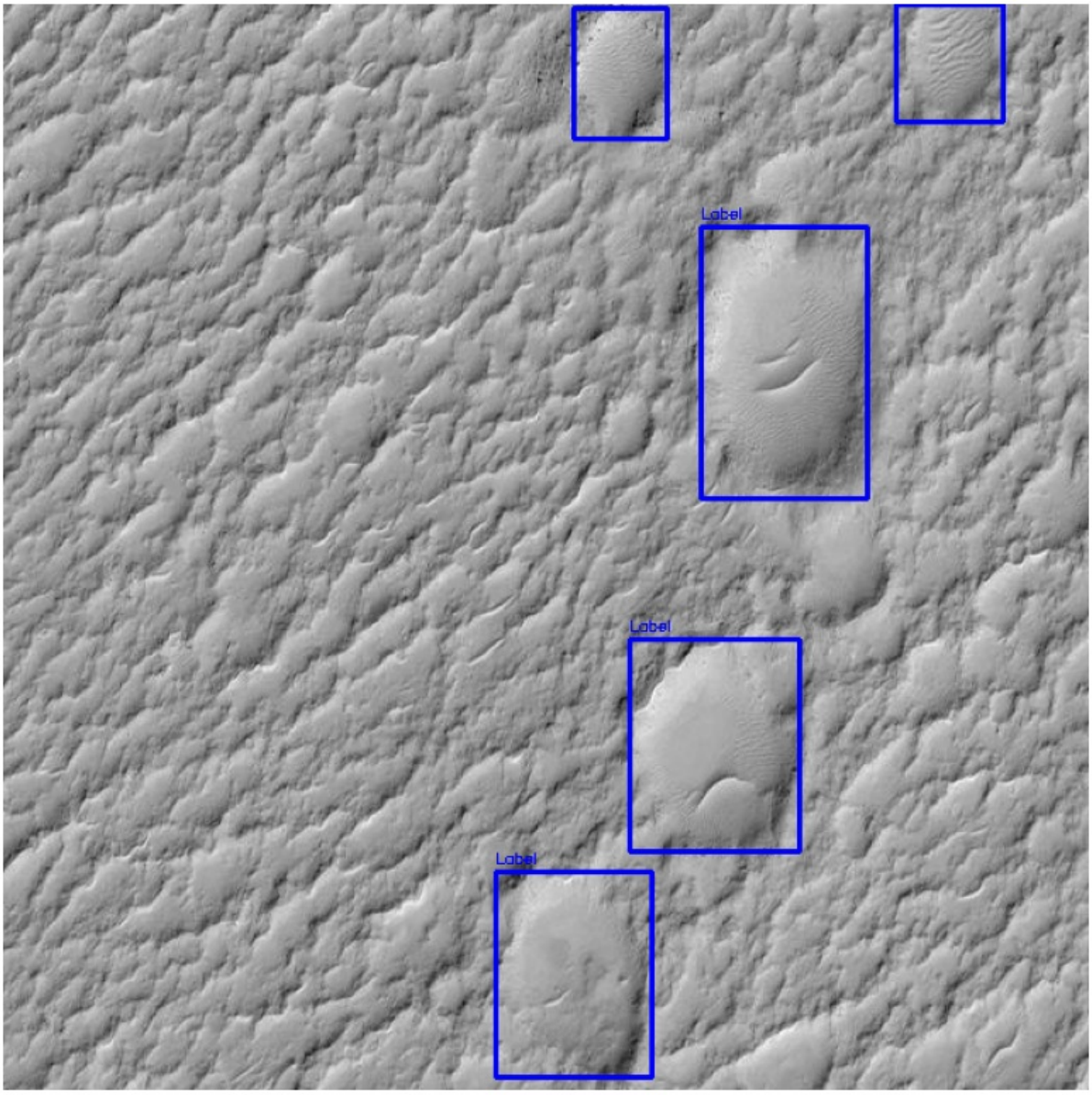


Image 9 done...

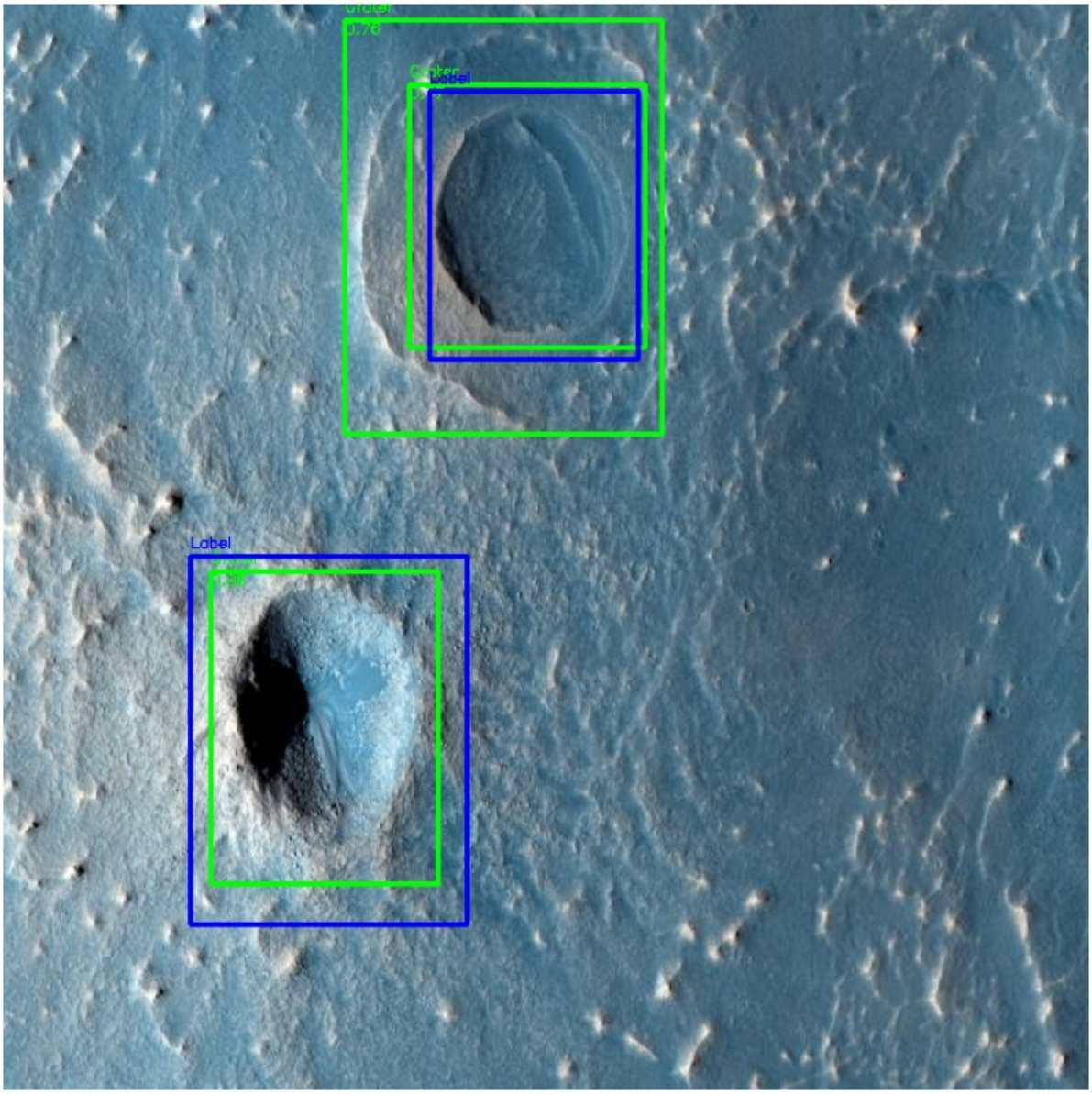


Image 10 done...

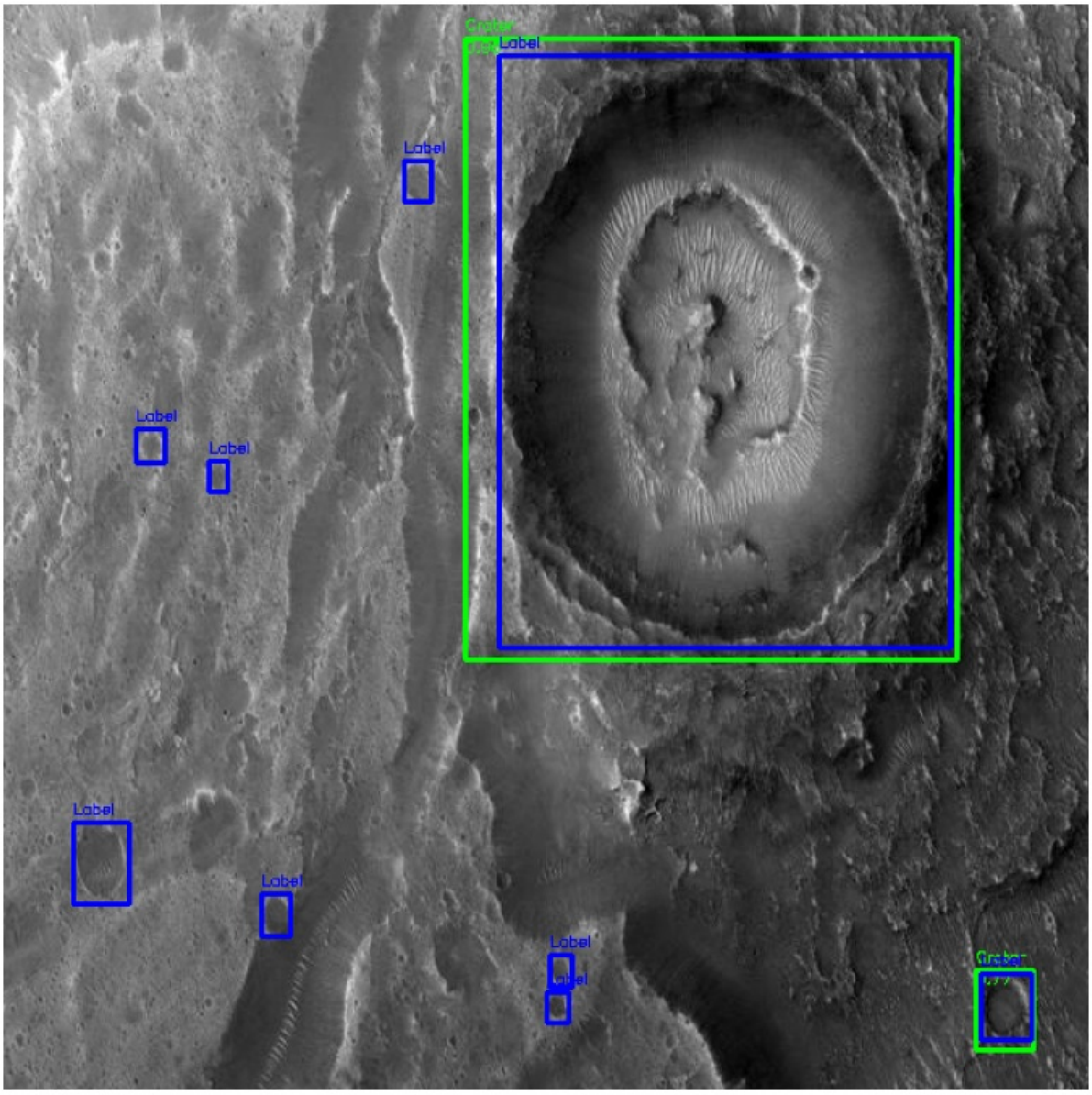


Image 11 done...

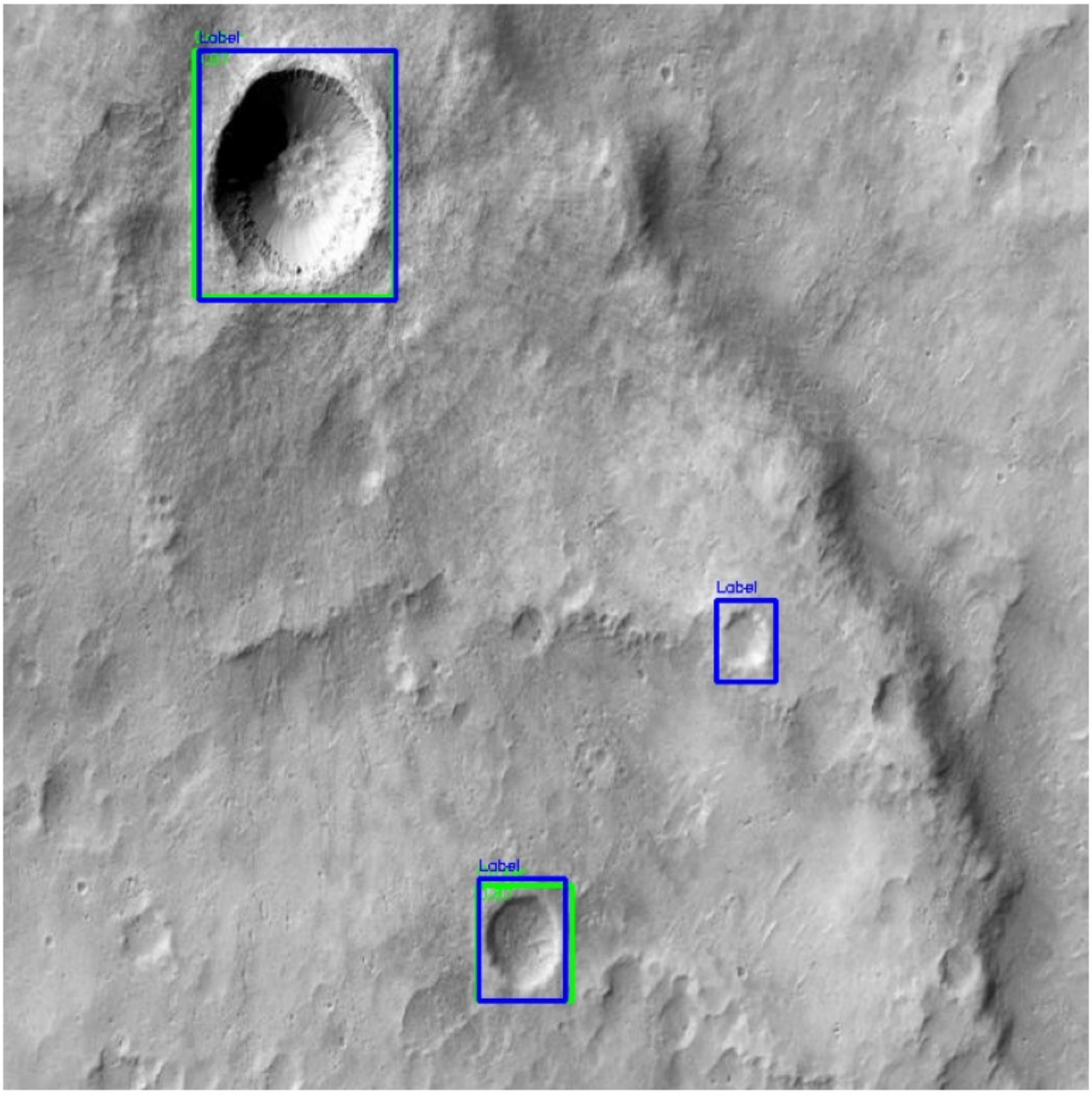


Image 12 done...

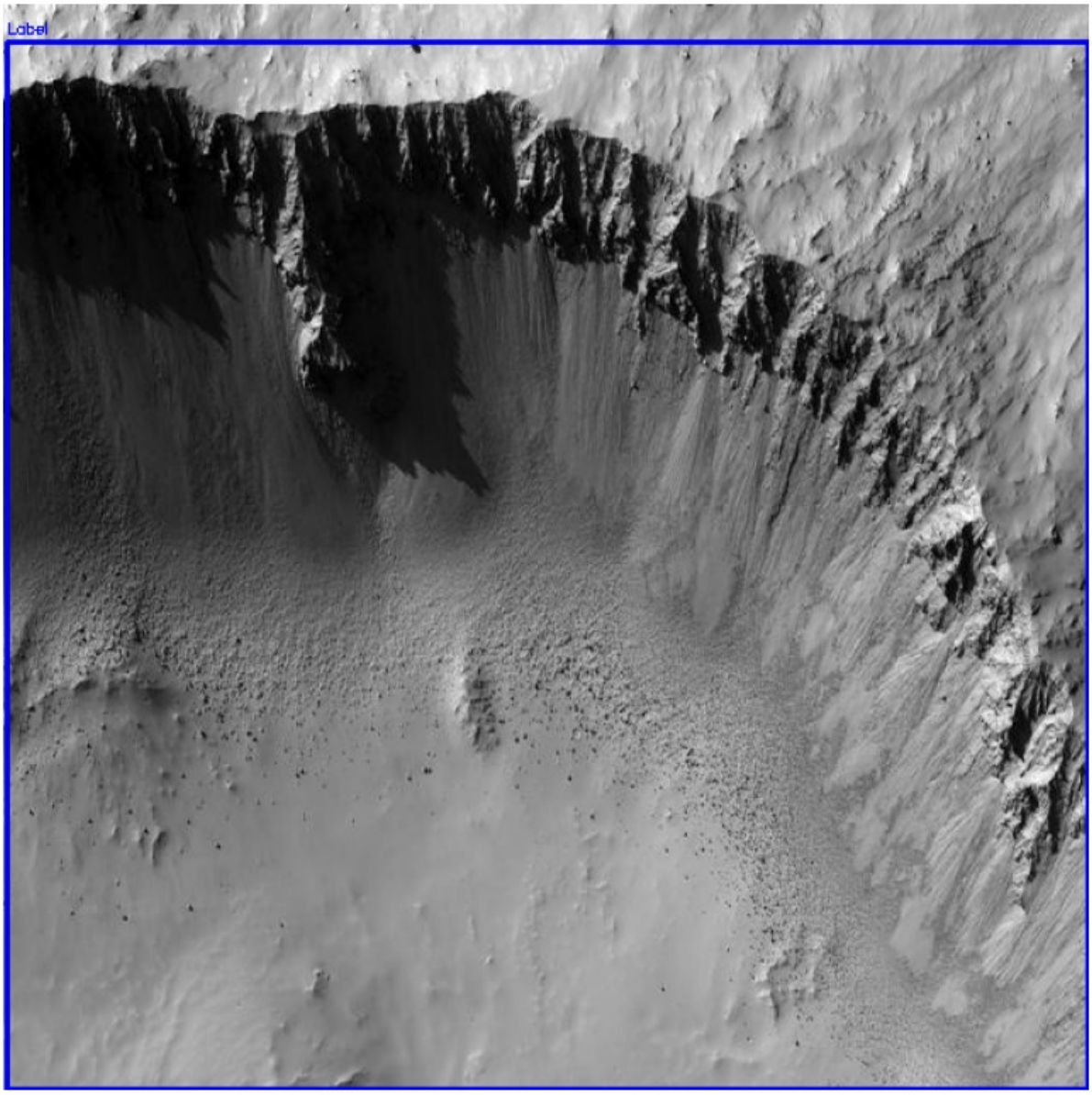


Image 13 done...

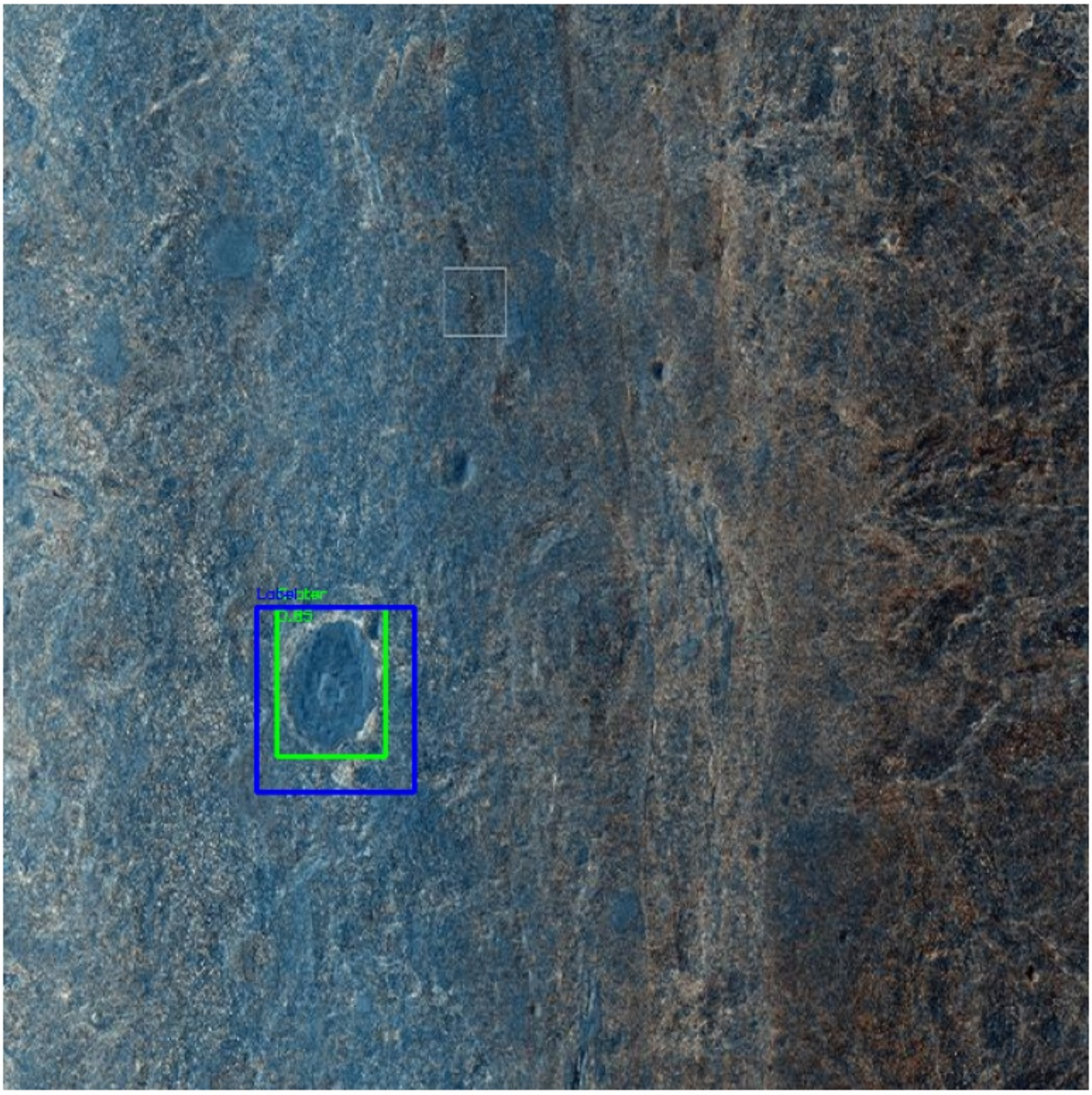


Image 14 done...

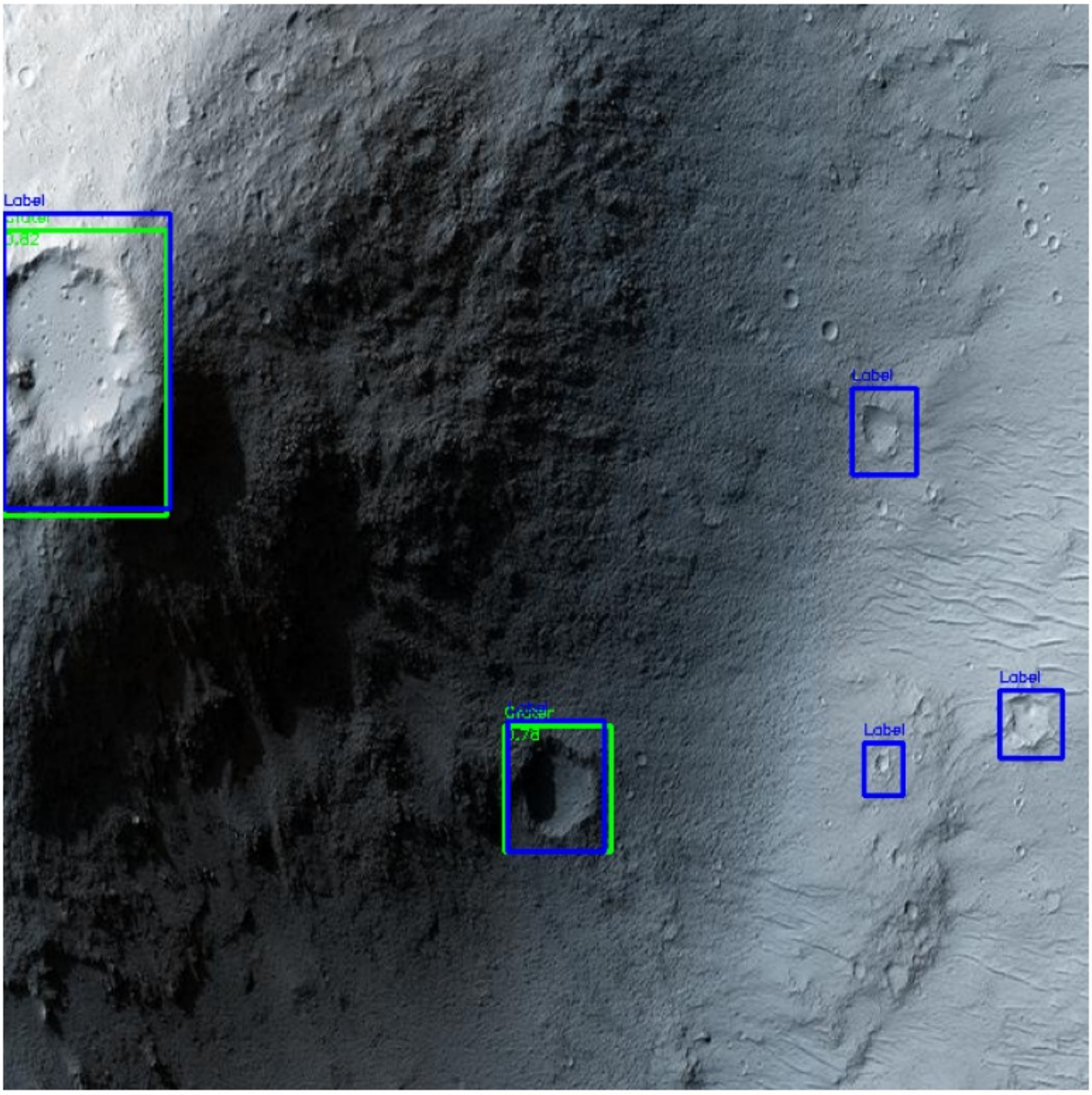


Image 15 done...

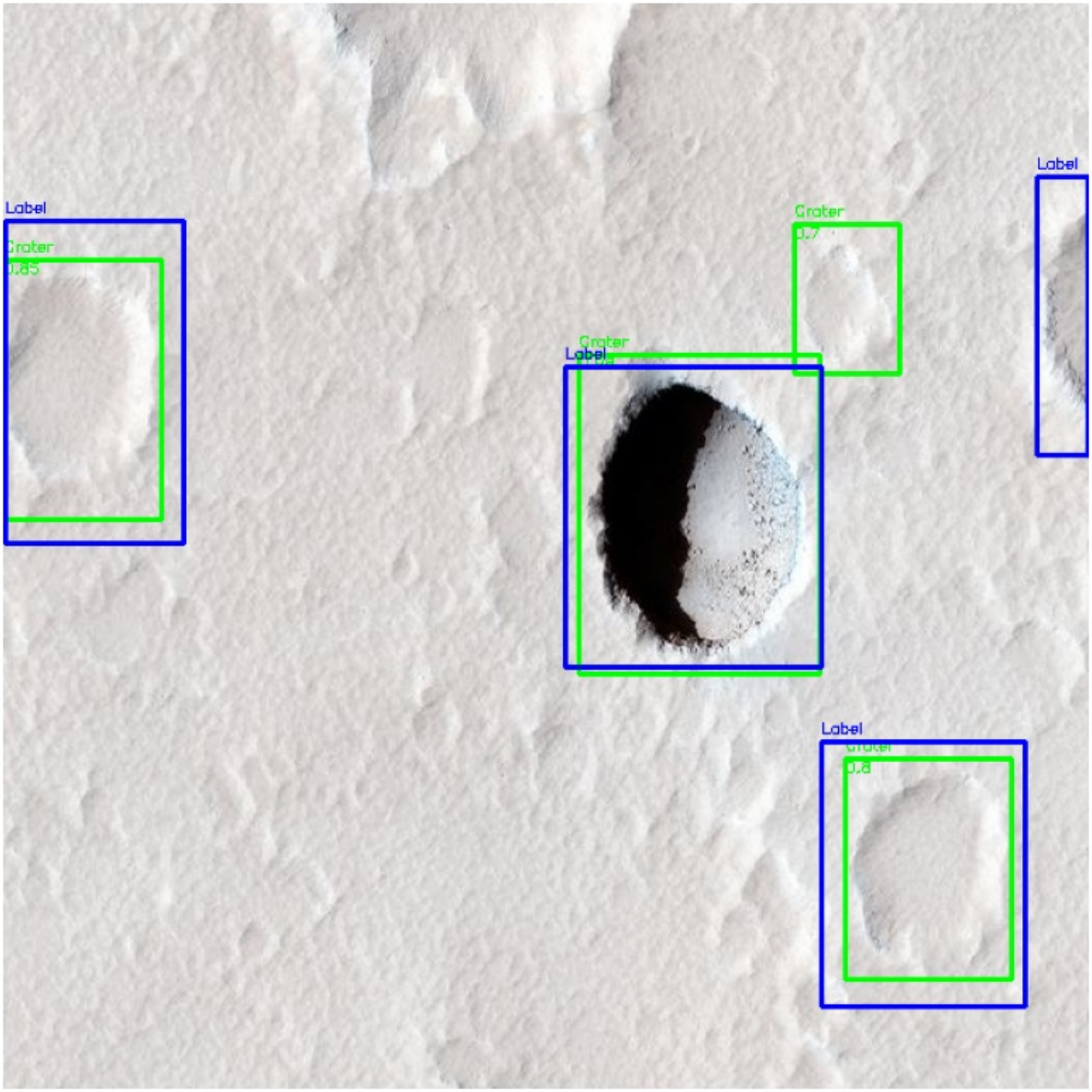


Image 16 done...

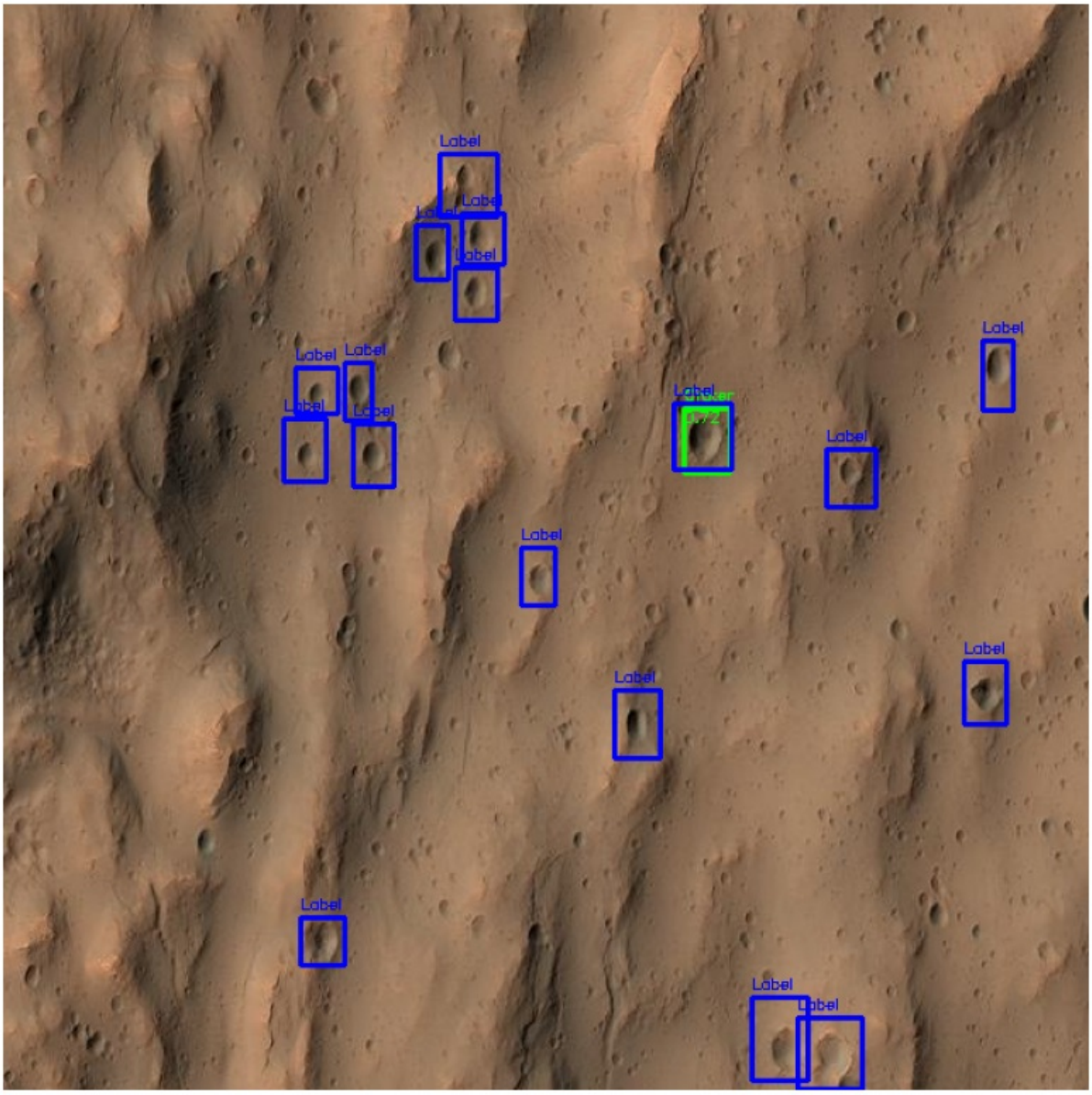


Image 17 done...

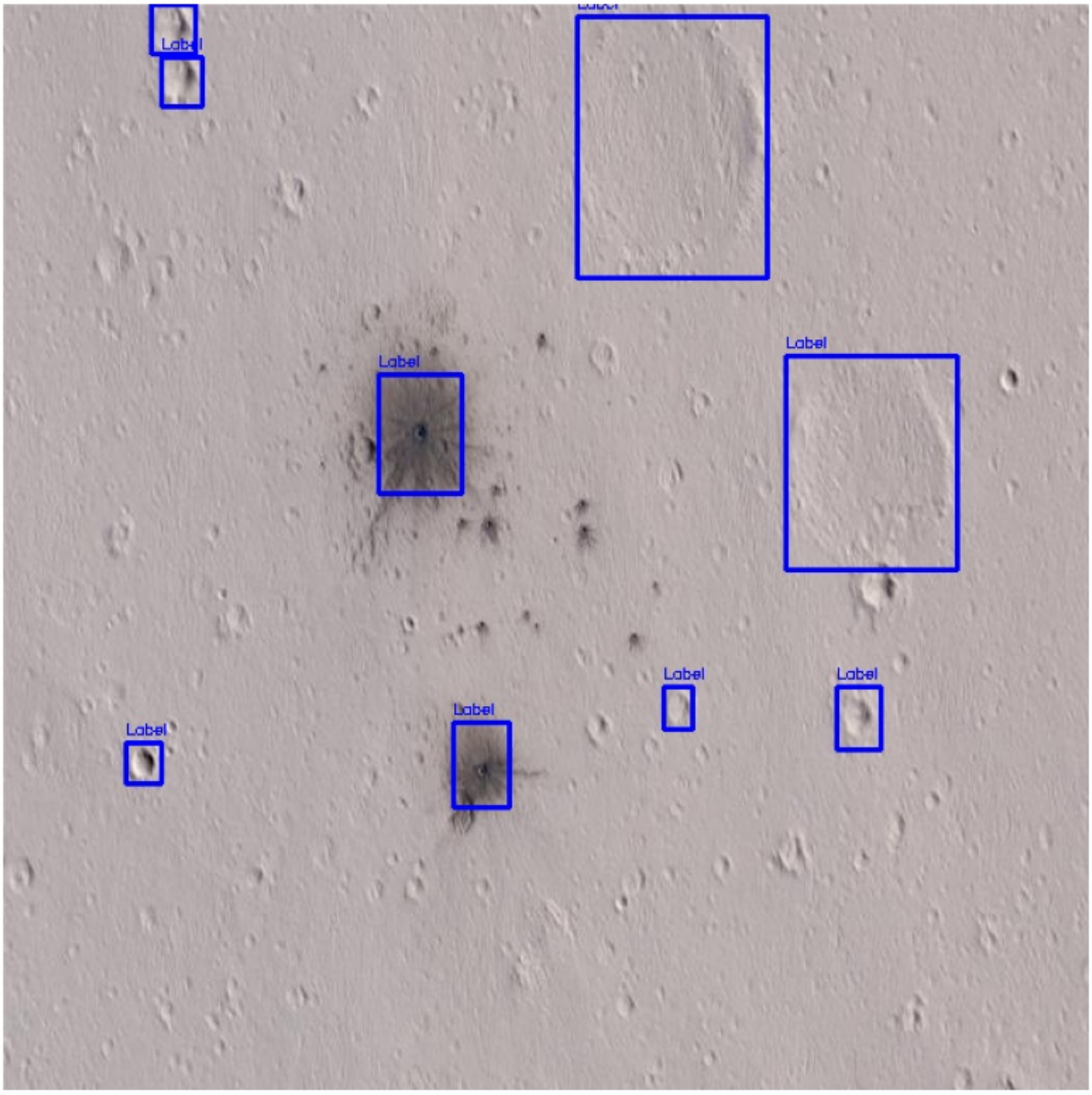
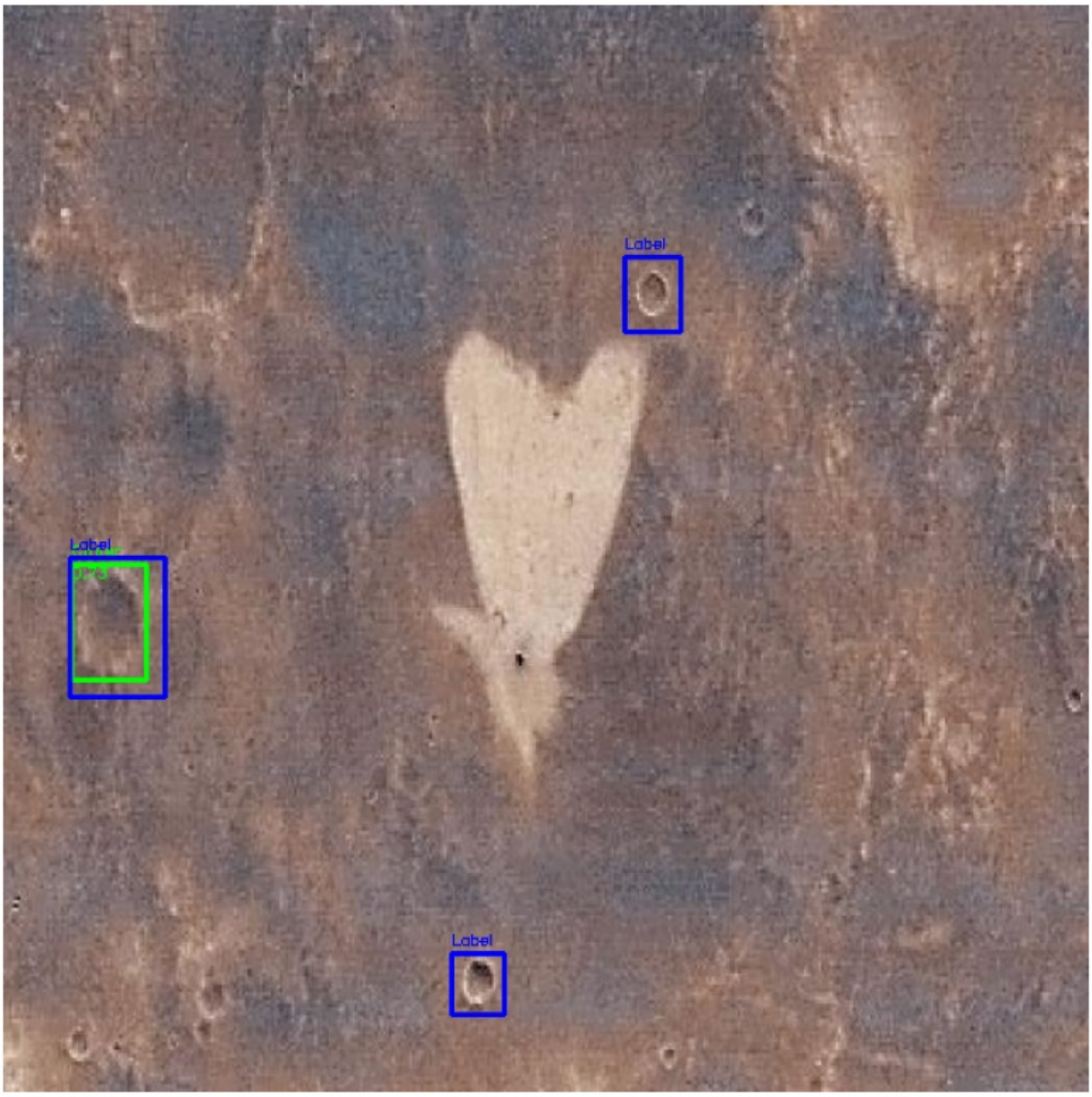


Image 18 done...




```
Image 19 done...
------------------------------------------------
TEST PREDICTIONS COMPLETE
Average FPS: 10.203
```

Para cuando nosotras ejecutamos el código, pudimos ver que este modelo hace una mejor detección que el anterior