

# MMDetection



# MMDetection

OpenMMLab

*OpenMMLab* è una organizzazione, un ecosistema, con lo scopo di permettere la ricerca e lo sviluppo nell'ambito dell'intelligenza artificiale e dell'apprendimento automatico.



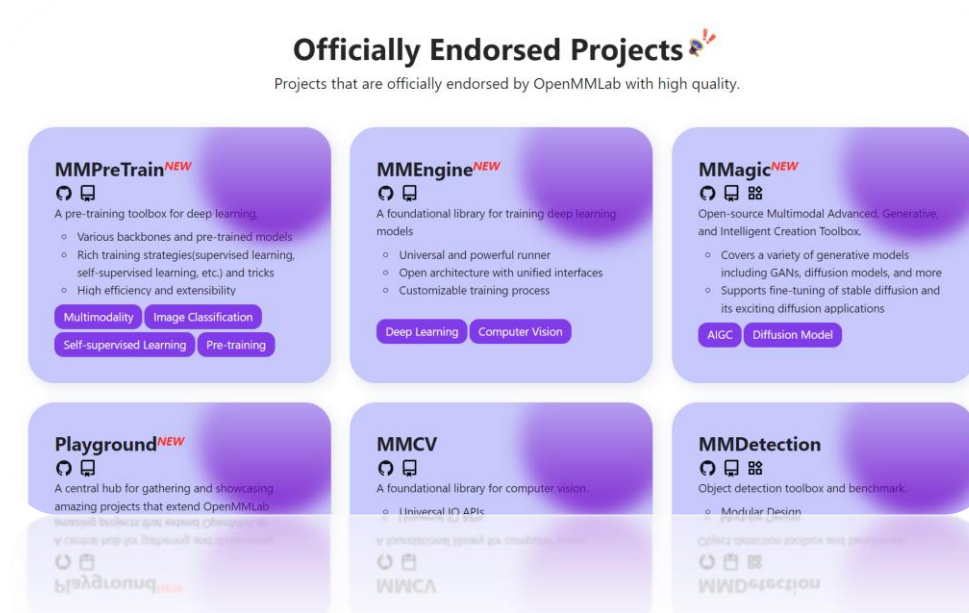
In particolare:

- È open source.
- È un ecosistema modulare, costituito di molteplici sotto-moduli.
- Ogni modulo è specifico per la risoluzione di un task o per agevolare l'automazione e il dialogo fra altri moduli.

# MMDetection

## OpenMMLab

Dal sito web è possibile esplorare la [Codebase](#), ossia la raccolta dei moduli che compongono *OpenMMLab*.



E per ogni modulo, accedere a risorse quali *GitHub*, documentazione e playground...

Di seguito, sarà presentato uno dei moduli più utilizzati: *mmdetection*.

# MMDetection

## Premessa

MMDetection è un framework di rilevamento oggetti, parte dell'ecosistema di moduli della famiglia di **OpenMMLab**.



Come altri in **OpenMm**:

- È basato su **PyTorch**.
- È modulare nelle implementazioni e flessibile nelle configurazioni.
- Facilita la sperimentazione di vecchi e nuovi algoritmi, tecniche di addestramento e creazione di modelli.
- Dà rapido accesso a numerosi algoritmi di rilevamento di oggetti:  
*Faster R-CNN, Mask R-CNN, RetinaNet...*

# MMDetection

## Premessa

Di seguito i riferimenti alle principali fonti di informazione legate al framework:

- Il repository **GitHub**.
- La documentazione.



È presente, inoltre, un riferimento al classico «*Get Started*»

[GET STARTED – MMDetection](#)

Per maggiore chiarezza, però, l'installazione verrà seguita *passo dopo passo* nelle prossime slide.

# MMDetection

## Installazione: conda

L'assunzione su cui si basano i prossimi passi di installazione è una:

- Nel pc, è installato il manager di ambienti virtuali **conda**.



Al termine dell'installazione di *mmdetection*, il risultato sarà, infatti, la presenza di un nuovo ambiente virtuale.

Il processo di addestramento, valutazione, e rilascio di modelli sarà gestito:

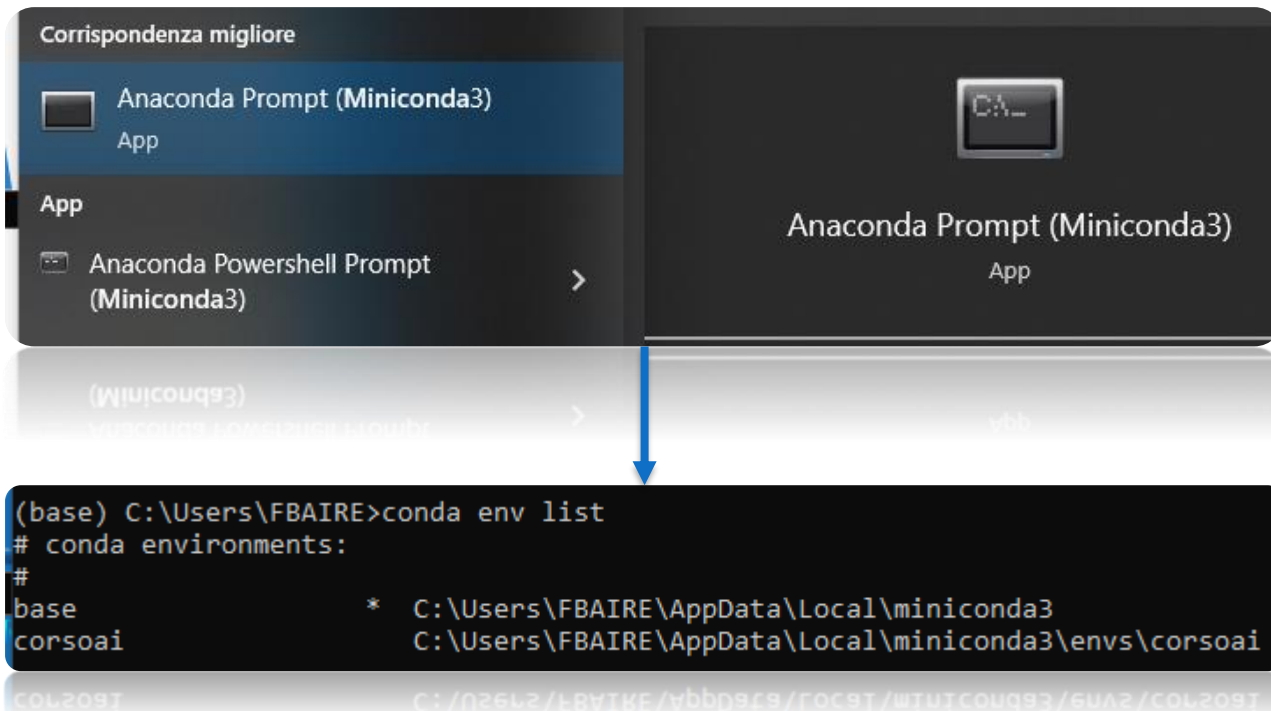
- Eseguendo script Python dall'ambiente conda attivato.
- Da terminale in visual studio code collegato all'ambiente.

# MMDetection

## Installazione: conda

Apriamo il terminale miniconda per verificare gli ambienti attualmente virtuali attualmente presenti nel sistema:

*conda env list*



The image shows a Windows Start menu search for 'Anaconda Prompt (Miniconda3)'. The search results list two applications: 'Anaconda Prompt (Miniconda3)' and 'Anaconda Powershell Prompt (Miniconda3)'. A blue arrow points from the first application to a terminal window below. The terminal window shows the command 'conda env list' being executed in a Miniconda3 environment. The output lists two environments: 'base' and 'corsoai', both located in the local Miniconda3 directory.

```
(base) C:\Users\FBAIRE>conda env list
# conda environments:
#
base          * C:\Users\FBAIRE\AppData\Local\miniconda3
corsoai       C:\Users\FBAIRE\AppData\Local\miniconda3\envs\corsoai
```

# MMDetection

## Installazione: root

Tramite il terminale *conda* ancora attivo, posizionarsi in un punto del sistema nel quale si vorrà creare la cartella di installazione del framework per poi procedere.

In questa guida, la cartella *root* dell'installazione sarà:

*c:\code\myGIT*

Spostarsi quindi nella *root*:

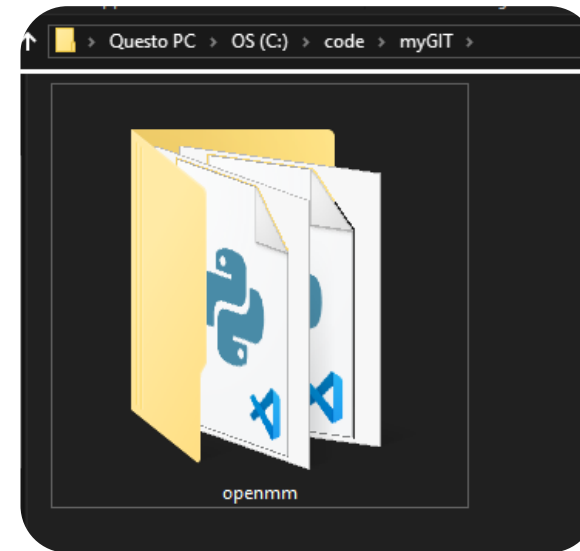
*cd <root>*

Creare la cartella di destinazione «*openmm*»:

*mkdir openmm*

Spostarti nella cartella di destinazione:

*cd openmm*





# MMDetection

## Installazione: ambiente

Da terminale *conda*, creare un nuovo ambiente virtuale di nome *openmm* e con l'installazione immediata del pacchetto python alla versione 3.11.

*conda create --name openmm python=3.11*

Alla richiesta di conferma di installazione dell'ambiente, confermare con 'y' e premendo «invio».

È possibile verificare la creazione dell'ambiente:

*conda env list*

```
(base) c:\code\myGIT\openmm>conda create --name openmm python=3.11
Retrieving notices: ...working... done
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\FBAIRE\AppData\Local\miniconda3\envs\openmm

  added / updated specs:
    - python=3.11
```

```
The following NEW packages will be INSTALLED:
python=3.11
```

```
(base) C:\Users\FBAIRE>conda env list
# conda environments:
#
base          * C:\Users\FBAIRE\AppData\Local\miniconda3
corsoai       C:\Users\FBAIRE\AppData\Local\miniconda3\envs\corsoai
openmm        C:\Users\FBAIRE\AppData\Local\miniconda3\envs\openmm
```



# MMDetection

## Installazione: git

Entrare nell'ambiente appena creato:

*conda activate openmm*

Procedere ad installare il primo pacchetto *git*:

*conda install git*

Alla richiesta di conferma, confermare con 'y' e premendo «invio».

```
(openmm) c:\code\myGIT\openmm>conda install git
Channels:
- conda-forge
- defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\Users\FBAIRE\AppData\Local\miniconda3\envs\openmm

added / updated specs:
- git
```

The following packages will be downloaded:

# MMDetection

## Installazione: pytorch

Procedere ad installare i pacchetti tramite i due canali conda dedicati a nvidia e pytorch:

- *pytorch*
- *torchvision*
- *pytorch-cuda*

*conda install pytorch==2.1.2 torchvision==0.16.2 pytorch-cuda=12.1 -c pytorch -c nvidia*

Alla richiesta di conferma, confermare con 'y' e premendo «invio».

Al momento della stesura di questa guida, le versioni compatibili sono:

- Pytorch → 2.1.2
- Torchvision → 0.16.2

Con pytorch-cuda si installano i pacchetti necessari all'utilizzo di gpu nvidia.

*Per una installazione cpu, sostituire pytorch-cuda=XY.Z con cpuonly*



# MMDetection

## Installazione: mim

Installare tramite *pip* il pacchetto *openmim*:

```
pip install -U openmm
```

Tramite questo pacchetto, sarà possibile eseguire installazioni mirate dei pacchetti e dei moduli dell'ecosistema di *openmm*.

Rif: [openmim](#)

A seguito dell'installazione, sarà possibile verificare quale di questi pacchetti sono stati installati attualmente nell'ambiente:

```
mim list
```

Inizialmente, non ve ne sarà alcuno.

```
(openmm) c:\code\myGIT\openmm>mim list
Package      Version      Source
-----
```

# MMDetection

## Installazione: mmengine

Procedere all'installazione di *mmengine* tramite il manager di pacchetti dedicato ad *openmm*:

```
mim install "mmengine>=0.7.0"
```

Il seguente pacchetto rappresenta il «motore» dell'ecosistema *openmm* al di sopra del quale vanno ad agganciarsi i moduli specifici legati ai task e tramite il quale si possono effettuare gli addestramenti dei modelli.

Rif: [mmengine](#)

```
(openmm) c:\code\myGIT\openmm>mim install "mmengine>=0.7.0"
Looking in links: https://download.openmmlab.com/mmcv/dist/cu121/torch2.1.0/index.html
Collecting mmengine>=0.7.0
  Downloading mmengine-0.10.4-py3-none-any.whl.metadata (20 kB)
Collecting addict (from mmengine>=0.7.0)
```

```
(openmm) C:\Users\FBAIRE>mim list
Package      Version      Source
-----
mmengine     0.10.4       https://github.com/open-mmlab/mengine
```

# MMDetection

## Installazione: mmcv

Procedere all'installazione di *mmcv* tramite il manager di pacchetti dedicato ad *openmm*:

*mim install "mmcv>=2.0.0rc4, <2.2.0"*

Tramite il pacchetto *mmcv* si ha accesso a funzionalità quali pre-processamento di video e immagini, annotazione e visualizzazione dati...oltre che l'accesso a modelli di rete CNN e operazioni specifiche su tensori.

Rif: [mmcv](#)

```
(openmm) c:\code\myGIT\openmm>mim install "mmcv>=2.0.0rc4, <2.2.0"
Looking in links: https://download.openmmlab.com/mmcv/dist/cu121/torch2.1.0/index.html
Collecting mmcv>=2.0.0rc4
  Downloading https://download.openmmlab.com/mmcv/dist/cu121/torch2.1.0/mmcv-2.2.0-cp311-cp311-win_amd64.whl (35.4 MB)
    35.4/35.4 MB 13.6 MB/s eta 0:00:00
Requirement already satisfied: addict in c:\users\fbairre\appdata\local\miniconda3\envs\openmm\lib\site-packages (from mmcv>=2.0.0rc4)
```

```
(openmm) C:\Users\FBAIRE>mim list
```

Package	Version	Source
mmcv	2.1.0	https://github.com/open-mmlab/mmcv
mmengine	0.10.4	https://github.com/open-mmlab/mengine

# MMDetection

## Installazione: mmdetection

Il pacchetto *mmdetection* sarà direttamente «clonato» tramite *git* precedentemente installato:

```
git clone -b main https://github.com/open-mmlab/mmdetection.git
```

```
(openmm) c:\code\myGIT\openmm>git clone -b main https://github.com/open-mmlab/mmdetection.git
Cloning into 'mmdetection'...
remote: Enumerating objects: 38019, done.
remote: Total 38019 (delta 0), reused 0 (delta 0), pack-reused 38019
Receiving objects: 100% (38019/38019), 63.17 MiB | 4.02 MiB/s, done.
Resolving deltas: 100% (26236/26236), done.
Updating files: 100% (2443/2443), done.
```

A seguito del download dei sorgenti nella cartella openmm precedentemente creata, si potrà procedere ad una installazione locale del pacchetto.

Rif: [mmdetection](#)

# MMDetection

## Installazione: mmpretrain

Il pacchetto *mmpretrain* sarà direttamente «clonato» tramite *git* precedentemente installato:

*git clone -b main https://github.com/open-mmlab/mmpretrain.git*

```
(openmm) c:\code\myGIT\openmm>git clone -b main https://github.com/open-mmlab/mmpretrain.git
Cloning into 'mmpretrain'...
remote: Enumerating objects: 17495, done.
remote: Counting objects: 100% (153/153), done.
remote: Compressing objects: 100% (99/99), done.
remote: Total 17495 (delta 63), reused 113 (delta 52), pack-reused 17342
Receiving objects: 100% (17495/17495), 13.82 MiB | 3.53 MiB/s, done.
Resolving deltas: 100% (12150/12150), done.
```

A seguito del download dei sorgenti nella cartella openmm precedentemente creata, si potrà procedere ad una installazione locale del pacchetto.

Rif: [mmpretrain](#)



# MMDetection

## Installazione: mmdeploy

Il pacchetto *mmdeploy* sarà direttamente «clonato» tramite *git* precedentemente installato:

```
git clone -b main https://github.com/open-mmlab/mmdploy.git
```

```
(openmm) c:\code\myGIT\openmm>git clone -b main https://github.com/open-mmlab/mmdploy.git
Cloning into 'mmdploy'...
remote: Enumerating objects: 25290, done.
remote: Counting objects: 100% (875/875), done.
remote: Compressing objects: 100% (501/501), done.
remote: Total 25290 (delta 440), reused 661 (delta 322), pack-reused 24415
Receiving objects: 100% (25290/25290), 13.57 MiB | 4.69 MiB/s, done.
Resolving deltas: 100% (16099/16099), done.
Updating files: 100% (1888/1888), done.
```

```
Updating files: 100% (1888/1888), done.
Resolving deltas: 100% (16099/16099), done.
```

A seguito del download dei sorgenti nella cartella openmm precedentemente creata, si potrà procedere ad una installazione locale del pacchetto.

Rif: [mmdeploy](#)

# MMDetection

## Installazione: mmdetection

Procedere all'installazione locale del pacchetto *mmdetection* accedendo alla cartella dei sorgenti scaricata:

*cd mmdetection*

E installando il pacchetto nell'ambiente virtuale, *openmm*, attivo:

*pip install -e .*

```
(openmm) c:\code\myGIT\openmm>cd mmdetection

(openmm) c:\code\myGIT\openmm\mmdetection>pip install -e .
Obtaining file:///C:/code/myGIT/openmm/mmdetection
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in c:\users\fbaire\appdata\local\miniconda3
Requirement already satisfied: numpy in c:\users\fbaire\appdata\local\miniconda3\envs
Collecting pycocotools (from mmdet==3.3.0)
  Using cached pycocotools-2.0.7-cp311-cp311-win_amd64.whl.metadata (1.1 kB)
Collecting scipy (from mmdet==3.3.0)
  Downloading scipy-1.13.0-cp311-cp311-win_amd64.whl.metadata (60 kB)
    ----- 60.6/60.6 kB 535.5 kB/s eta 0:00:00
Collecting shapely (from mmdet==3.3.0)
Collecting sparse (from mmdet==3.3.0)
    ----- 60.6/60.6 kB 535.5 kB/s eta 0:00:00
Downloading sparse-0.15.2-cp311-cp311-win_amd64.whl (60.6 kB)
```

# MMDetection

## Installazione: mmpretrain

Procedere all'installazione locale del pacchetto *mmpretrain* accedendo alla cartella dei sorgenti scaricata:

*cd .. → cd mmpretrain*

E installando il pacchetto nell'ambiente virtuale, *openmm*, attivo:

*pip install -e .*

```
(openmm) c:\code\myGIT\openmm>cd mmpretrain  
  
(openmm) c:\code\myGIT\openmm\mmpretrain>pip install -e .  
Obtaining file:///C:/code/myGIT/openmm/mmpretrain  
  Preparing metadata (setup.py) ... done  
Collecting einops (from mmpretrain==1.2.0)  
  Downloading einops-0.8.0-py3-none-any.whl.metadata (12 kB)  
Requirement already satisfied: importlib-metadata in c:\users\fbair\appdata\local\mini  
Collecting mat4py (from mmpretrain==1.2.0)  
  Downloading mat4py-0.6.0-py2.py3-none-any.whl.metadata (2.8 kB)  
Requirement already satisfied: matplotlib in c:\users\fbair\appdata\local\miniconda3\en  
Collecting modelindex (from mmpretrain==1.2.0)  
  Downloading modelindex-0.0.2-py3-none-any.whl.metadata (756 bytes)  
  Downloading modelindex-0.0.2-py3-none-any.whl (1.2 kB)  
Collecting modelindex (from mmpretrain==1.2.0)  
  Downloading modelindex-0.0.2-py3-none-any.whl (1.2 kB)
```

# MMDetection

## Installazione: mmdeploy

Del pacchetto *mmdeploy*, sono stati scaricati i sorgenti tramite il pacchetto *git*. In questo modo sarà possibile fare riferimenti «locali» ai file, quali quelli di configurazione, che permettono il rilascio dei modelli.

Procediamo poi ad installare il pacchetto nell'ambiente *openmm*:

*pip install mmdeploy==1.3.1*

```
(openmm) c:\code\myGIT\openmm\mmtrain>pip install mmdeploy==1.3.1
Collecting mmdeploy==1.3.1
  Downloading mmdeploy-1.3.1-py3-none-win_amd64.whl.metadata (19 kB)
Collecting aenum (from mmdeploy==1.3.1)
  Downloading aenum-3.1.15-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: grpcio in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (1.48.0)
Requirement already satisfied: matplotlib in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (3.5.3)
Requirement already satisfied: mmengine in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (0.6.0)
Collecting multiprocess (from mmdeploy==1.3.1)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: numpy in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (1.24.3)
Requirement already satisfied: packaging in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (23.1)
Requirement already satisfied: pyparsing in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (3.1.0)
Requirement already satisfied: python-dateutil in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (2.8.2)
Requirement already satisfied: six in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (1.16.0)
Requirement already satisfied: tqdm in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (4.64.1)
Requirement already satisfied: typing-extensions in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (4.5.0)
Requirement already satisfied: wheel in c:\users\fbairre\appdata\local\miniconda2\envs\openmm\lib\site-packages (0.40.0)
Installing collected packages: aenum, multiprocess, mmdeploy
Successfully installed aenum-3.1.15 mmdeploy-1.3.1 multiprocess-0.70.16
```

# MMDetection

## Installazione: mmdeploy-runtime

La stessa procedura, per il pacchetto *mmdeploy-runtime*:

*pip install mmdeploy-runtime==1.3.1*

```
(openmm) c:\code\myGIT\openmm\mmtrain>pip install mmdeploy-runtime==1.3.1
Collecting mmdeploy-runtime==1.3.1
  Downloading mmdeploy_runtime-1.3.1-cp311-none-win_amd64.whl.metadata (310 bytes)
  Downloading mmdeploy_runtime-1.3.1-cp311-none-win_amd64.whl (13.9 MB)
    ----- 13.9/13.9 MB 5.0 MB/s eta 0:00:00
Installing collected packages: mmdeploy-runtime
Successfully installed mmdeploy-runtime-1.3.1
```

Successfully installed mmdeploy-runtime-1.3.1

# MMDetection

## Installazione: conclusione

Al termine della procedura, riassumendo:

- Si avrà l'ambiente virtuale *openmm*.
- Sarà installato parte dell'ecosistema *openmm*:
  - *mmcv*.
  - *mmdet*.
  - *mmengine*.
  - *mmpretrain*.
- Saranno presenti, accessibili e modificabili i file sorgenti di questi pacchetti.

Per verificare:

*mim list*

# MMDetection

## Installazione: conclusione

```
(openmm) C:\Users\FBAIRE>mim list
Package      Version      Source
-----
mmcv         2.1.0        https://github.com/open-mmlab/mmcv
mmdet        3.3.0        c:\code\mygit\openmm\mmdetection
mmengine     0.10.4       https://github.com/open-mmlab/mengine
mmpretrain   1.2.0        c:\code\mygit\openmm\mmpretrain
```

### ✓ OPENMM

- > mmdeploy
- > mmdetection
- > mmpretrain

Proviamo?

