

OneAI 文件 (/s/user-guide)

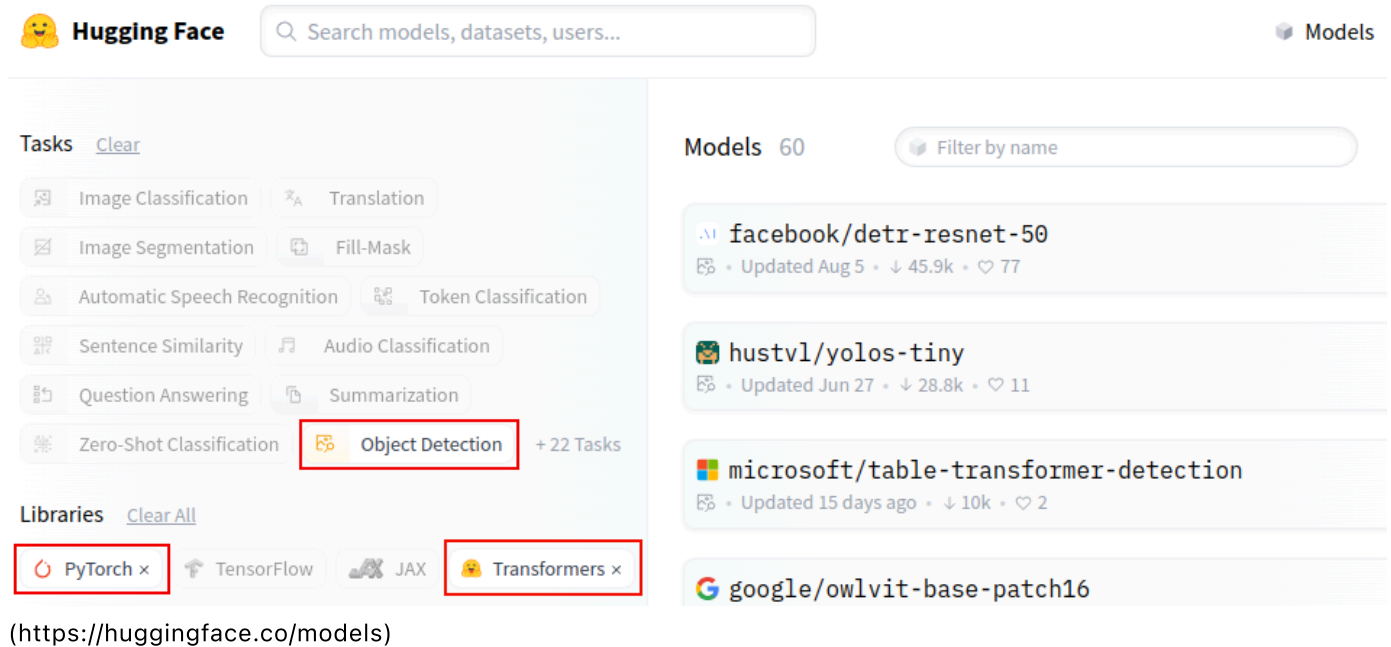
AI Maker 案例教學 - Hugging Face 物件偵測應用

- AI Maker 案例教學 - Hugging Face 物件偵測應用
 - 0. 前言
 - 1. 準備資料集
 - 1.1 資料格式說明
 - 1.2 建立儲存體
 - 1.3 下載資料集
 - 2. 訓練物件偵測任務模型
 - 2.1 建立訓練任務
 - 2.2 啟動訓練任務
 - 2.3 檢視訓練結果
 - 2.4 註冊模型
 - 3. 建立推論服務
 - 3.1 建立推論任務
 - 3.2 進行推論

0. 前言

Hugging Face (<https://huggingface.co/>) 是公開的 AI community，支援主流的深度學習框架，並提供大量的預訓練模型、資料集及開發工具，包括自然語言處理、語音辨識及影像辨識等多種 AI 應用。

透過 OneAI 提供的 **huggingface** 容器映像檔，和 AI Maker 基於 PyTorch 與 transformers 框架所整合的 **任務** 範本可協助您運用 OneAI 的運算資源，加速 AI 應用的開發。



(圖片來源：Hugging Face) (<https://huggingface.co/models>)

在本範例中，我們將使用 AI Maker 針對 Hugging Face 物件偵測應用所提供的範本，逐步建立一個物件偵測應用，此範本中定義了訓練任務和推論任務中所需環境變數、映像檔、程式等設定，您只須準備訓練用的資料集或推論用的模型，即可快速執行訓練與推論任務。

主要步驟如下：

- **資料集準備**

在此階段，我們會從 **Hugging Face Datasets** (<https://huggingface.co/datasets>) 下載公開的資料集，並整理成符合此模型訓練的資料格式。

- **訓練模型**

在此階段，我們將配置訓練任務，以進行模型的訓練與擬合，並將訓練好的模型儲存。

- **建立推論服務**

在此階段，我們會將儲存下來的模型部署到服務中，以執行推論。

提示：參考資訊

- Hugging Face 官方網站 (<https://huggingface.co/>)
- Hugging Face - CPPE-5 Dataset (<https://huggingface.co/datasets/cppe-5>)

1. 準備資料集

本章節將透過程式來下載 Hugging Face Datasets 所提供的 **CPPE-5 公開資料集** (<https://huggingface.co/datasets/cppe-5>)，如果您想要使用自有的資料集，只需要符合本章節中的 **訓練資料格式**，再上傳至 **儲存服務** 的儲存體即可。

1.1 資料格式說明

本範例是偵測醫護場景中相關的醫療個人防護設備，分為以下幾類：

category_id	name
0	coveralls
1	mask
2	face shield
3	gloves
4	goggles

其中物件標註格式採用類 COOC format，必填屬性為：

- image_id：物件對應的 image id，不過在本範例中為一張圖片對應多個物件，故無利用到該屬性。
- area：矩形物件的面積大小，即長 x 寬。
- bbox：矩形物件的四點坐標，分別為 xmin、ymin、xmax、ymax。
- category_id：物件所屬的類別編號。

訓練資料支援 **JSON** 格式，為兩欄式的資料呈現。

欄位	說明	範例
path	圖片檔的相對路徑，相對於 JSON 檔案	train/000001.jpg
annotations	物件標示內容	[{"image_id": 1, "area": 3796, "bbox": [302.0, 109.0, 73.0, 52.0], "category_id": 4}]

JSON 格式範例

```
{"path": "train/000001.jpg", "annotations": [{"image_id": 15, "area": 3796, "bbo
...
```

1.2 建立儲存體

從 OneAI 服務列表選單選擇「儲存服務」，進入儲存服務管理頁面，接著點擊「+ 建立」，新增一個名為 **hf-cppe5** 的儲存體，此儲存體稍後會用來存放資料集。

1.3 下載資料集

本章節將使用 OneAI 的 筆記本服務 從 **Hugging Face Datasets**

(<https://huggingface.co/docs/datasets/index>) 下載 **CPPE-5 (醫療個人防護裝備資料集)**

(<https://huggingface.co/datasets/cppe-5>) 的 train 和 test 兩組資料集當成訓練和驗證資料。

1.3.1 建立筆記本服務

從 OneAI 服務列表選單選擇「**筆記本服務**」，進入筆記本服務管理頁面，接著點擊「**+ 建立**」。

筆記本服務的建立資訊如下，更多資訊請參閱 **筆記本服務 (/s/notebook)** 文件。

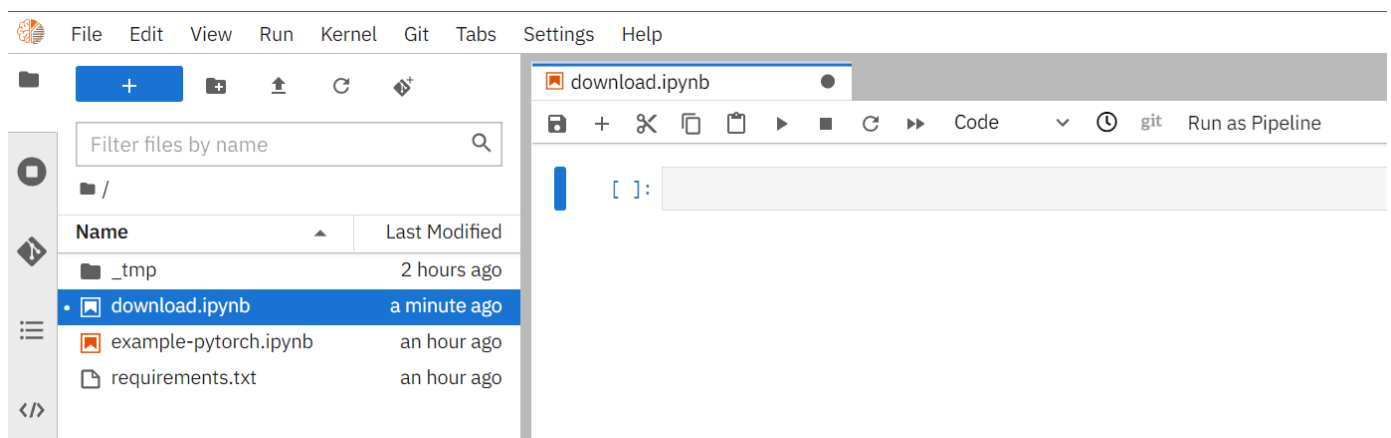
- 基本資訊
 - 名稱：**hf-demo**，名稱不可重複，請自行命名
 - 開發框架：**PyTorch-21.02-py3**
- 硬體設定：選擇最小的運算資源即可
- 儲存設定：掛載訓練資料的儲存體
 - 掛載路徑：**/workspace**
 - 儲存：**/hf-cppe5**

筆記本服務建立完成後，進入筆記本服務詳細資料頁面，在 **連線** 區塊，點擊 JupyterLab 右側的「**啟動**」。



1.3.2 下載並處理資料

開啟 JupyterLab 後，建立一個名為 **download.ipynb** 的 Notebook。



接著，複製並執行下列的程式碼來下載資料集。

```
1 # 安裝 Datasets & image library
2 !pip install datasets==2.5.0
3 !pip install Pillow==9.0.0
4
5 # 下載 Hugging Face 資料集
6 # 請確保儲存空間中沒有以該變數為名稱的資料夾存在，以免發生錯誤
7 folder = "cppe-5"
8 from datasets import load_dataset
9 cppe5 = load_dataset(folder)
10
11 print(cppe5)
12 print(cppe5["train"].features)
13 print(cppe5["train"][0])
14
15 # 儲存成對應格式
16 import json
17 import os
18 from PIL import Image
19
20 def save2Json(dataset, folder, outputFile):
21     dirname = os.path.dirname(outputFile)
22     os.makedirs(dirname, exist_ok=True)
23     with open(outputFile, 'w+', encoding='utf-8') as f:
24         num = 1
25         for record in dataset:
26             data = {}
27             data["annotations"] = []
28             objs = record["objects"]
29             size = len(objs["id"])
30             for i in range(size):
31                 obj = {}
32                 obj["image_id"] = record["image_id"]
33                 obj["area"] = objs["area"][i]
34                 obj["bbox"] = objs["bbox"][i]
35                 obj["category_id"] = objs["category"][i]
36                 data["annotations"].append(obj)
37                 data["path"] = "{}/{:06d}.{}".format(folder, num, record['
38 filename = "{}/{}/{:06d}.{}".format(dirname, folder, num, record['
39 os.makedirs(os.path.dirname(filename), exist_ok=True)
40 record["image"].save(filename)
41 f.write(json.dumps(data, ensure_ascii=False) + "\n")
42 num += 1
43
44 save2Json(cppe5["train"], 'train', 'cppe5/train.json')
45 save2Json(cppe5["test"], 'val', 'cppe5/val.json')
```

執行完畢後會產生 cppe5 資料夾並且包含 train.json 與 val.json 兩個檔案，目錄結構如下：

```
├── cppe5
│   ├── train
│   │   ├── 000001.JPEG
│   │   ├── 000002.JPEG
│   │   └── 000003.JPEG
│   └── ...
│       ├── train.json
│       ├── val
│       │   ├── 000001.JPEG
│       │   ├── 000002.JPEG
│       │   └── 000003.JPEG
│       └── ...
└── val.json
```

2. 訓練物件偵測任務模型

完成 資料集的準備 後，就可以使用這些資料，來訓練與擬合我們的物件偵測任務模型。

2.1 建立訓練任務

從 OneAI 服務列表選擇「**AI Maker**」，再點擊「**訓練任務**」，進入訓練任務管理頁面。AI Maker 提供 **Smart ML 訓練任務** 與 **一般訓練任務** 兩種訓練方法，訓練方法不同所須設定的參數也有所不同。

訓練任務管理 [深入了解](#)

SMART ML 訓練任務

一般訓練任務

- **一般訓練任務**

根據您所給定的訓練參數，執行一次性的訓練。

- **Smart ML 訓練任務**

可自動調整超參數，能夠有效地將計算資源用於多個模型訓練，節省您在分析和調整模型訓練參數上的時間和成本。

在此範例中我們選用 **一般訓練任務** 來建立一個新的訓練任務。訓練任務的建立步驟如下，詳細說明可參考 **AI Maker > 訓練任務** (<https://ai-maker.twcc.ai/s/hf-object-detection>)。

1. 基本資訊

AI Maker 為物件偵測訓練提供 **huggingface-object-detection** 範本，在輸入名稱與描述後，您可以選擇系統所提供的 **huggingface-object-detection** 範本，自動帶出公用映像檔 **huggingface:v1** 及後續步驟的各項參數設定。



提示： 名稱不可重複，請自行命名。

建立訓練任務

1

2

3

4

5

基本資訊

硬體設定

儲存設定

變數設定

檢閱 + 建立

MLflow Experiment ⓘ

選擇

名稱 *

hf-demo

描述

Hugging Face Demo

選擇方法 *

選擇

一般訓練任務

選擇範本

選擇

huggingface-object-detection

映像檔來源 *

選擇

huggingface

映像檔標籤 *

v1

2. 硬體設定

參考目前的可用配額與訓練程式的需求，從列表中選出合適的硬體資源，選擇包含 **GPU** 的硬體選項可加速運算。

3. 儲存設定

這個階段是將我們存放訓練資料的儲存體掛載到訓練環境中。掛載路徑與環境變數的宣告在範本中已經設定完成，這邊只要選擇在 **建立儲存體** 步驟所建立的儲存體名稱即可。

建立訓練任務

✓

✓

3

4

5

基本資訊硬體設定儲存設定變數設定檢閱 + 建立

輸入來源 *

名稱

掛載路徑

儲存

INPUT

/datasets

hf-cppe5 ▾

🗑

新增

輸出位置 *

名稱	掛載路徑	儲存
OUTPUT	/output	system

4. 變數設定

當在填寫基本資訊，選擇套用 **huggingface-object-detection** 的範本，會自動帶入基本的變數與命令，變數設定值可依照開發需求來進行調整或新增，範本 **huggingface-object-detection** 所提供的參數如下描述。

變數名稱	預設值	說明
task_type	object-detection	Hugging Face 任務類型
training_file	/datasets /cppe5 /train.json	訓練集描述檔，掛載儲存體
validation_file	/datasets /cppe5 /val.json	驗證集描述檔，如果沒有特別指定 validation_size 從訓練集
validation_size	0.2	沒有指定驗證集才會有作用 依設定比例來隨機產生驗證集 外也可以指定要多少筆（整數）
pretrained_model	facebook/detr-resnet-50 (https://huggingface.co/facebook/detr-resnet-50)	AI 任務對應的預訓練模型， Models (https://huggingface.co) 也可以使用置放在儲存體之 在建立訓練任務時新增輸入：
pretrained_tokenizer		沒有設定會參照 pretrained_model 除指定 Hugging Face Models (https://huggingface.co/models) 也可以使用置放在儲存體之 資料夾的絕對路徑
model_config		沒有設定會參照 pretrained_model 除指定 Hugging Face Models (https://huggingface.co/models) 也可以使用置放在儲存體含 config.json 資料夾的絕對
from_scratch	False	如果設定 True 會重新訓練整個模型而不使
only_top_layer	False	如果設定 True 則模型中的 只調整 top layer
score_thresholds	0.7	分數門檻值
iou_thresholds	0.5	IoU (intersection over union) 預測物件區域與真實物件區

 **提示：預訓練模型**
pretrained_model 必須要是基於 PyTorch 以及 Transformers 框架並符合該任務類別 (object-detection) 的模型才能採用。

進階的訓練參數設定可參考 **Hugging Face - TrainingAuguments**
(https://huggingface.co/docs/transformers/v4.21.1/en/main_classes/trainer#transformers.TrainingArguments)
官方文件說明，以下說明幾項常用的參數。

變數名稱	預設值	說明
num_train_epochs	30	全部資料集訓練的次數，若有設定 max_steps (https://huggingface.co/docs/transformers/v4.21.3/en/main_classes/trainer#transformers.TrainingArguments) 參數，則 epochs 數目會被覆蓋掉
learning_rate	0.00002	學習率
per_device_train_batch_size per_device_eval_batch_size		
auto_find_batch_size	True	啟用後會自動找尋合適 batch_size 來避免 OOM
load_best_model_at_end	False	訓練後是否要儲存最佳模型
evaluation_strategy save_strategy	epoch epoch	這兩項參數的值必須同時為 epoch 或是 steps
save_total_limit	2	必須大於或等於 2，儲存 checkpoints 個數
...	...	其他變數則參照原有 函式庫 (https://huggingface.co/docs/transformers/v4.21.3/en/main_classes/trainer#transformers.TrainingArguments)

💡 提示：目前不支援的項目

變數名稱	預設值	說明
use_iepx	False	目前不支援 Intel® Extension
tf32	False	目前不支援 tf32 相關設置
bf16 bf16_full_eval	False False	目前不支援 bf16 相關設置
xpu_backend		目前不支援分散式訓練 (mpi/ccl)
tpu_num_cores		目前不支援 TPU
sharded_ddp	False	目前不支援 FairScale (https://github.com/facebookresearch/fairscale)
fsdp fsdp_min_num_params	False 0	目前不支援 FSDP (https://pytorch.org/docs/stable/fsdp.html)
deepspeed		目前不支援 DeepSpeed (https://github.com/microsoft/deepspeed)

5. 環境變數與超參數

根據 **建立訓練任務** 時所選擇的訓練方法不同，**Smart ML 訓練任務** 與 **一般訓練任務** 的變數設定會稍有不同。

欄位名稱	說明
環境變數	輸入環境變數的名稱及數值。 這邊的環境變數除了包含訓練執行的相關設定外， 也包括了訓練網路所需的參數設定
超參數*	(Smart ML 訓練任務) 這是告訴任務，有哪些參數需要進行嘗試。 每個參數在設定時，須包含參數的名稱、類型及數值（或數值範圍）， 選擇類型後（整數、小數和陣列），請依提示輸入相對的數值格式
目標參數*	(Smart ML 訓練任務) 在使用 Bayesian 或 TPE 演算法時，會基於 目標參數 的結果來反覆調校出合適參數，作為下次訓練任務的基準。 訓練結束會回傳一值做為最終結果，這邊為該值設定名稱及目標方向。 例如：若回傳的數值為準確率，則可命名為 accuracy， 並設定其目標方向為最大值；若回傳的值為錯誤率，則命名為 error， 其方向為最小值。 根據該任務類型所提供的 metrics 為 mAP，其方向為 最大值
命令	輸入欲執行的命令或程式名稱。 根據此映像檔所提供的指令為： <code>python3.8 /usr/src/app/training.py</code>
任務次數*	(Smart ML 訓練任務) 即訓練次數設定，讓訓練任務執行多次， 以找到更好的參數組合

* **超參數** 與 **任務次數** 為選擇 **Smart ML 訓練任務** 時才需設定的參數。


其中，**環境變數** 與 **超參數** 可以互相移動。若您想固定該參數，則可將該參數從超參數區域中移除，新增至環境變數區域，並給定固定值；反之，若想將該參數加入嘗試，則將它從環境變數中移除，加入至下方的超參數區域。

環境變數

名稱	數值
task_type	object-detection
training_file	/datasets/cppe5/train.json
validation_file	/datasets/cppe5/val.json
only_top_layer	False
from_scratch	False
pretrained_model	facebook/detr-resnet-50
learning_rate	0.00002
num_train_epochs	30
auto_find_batch_size	True
score_thresholds	0.7
iou_thresholds	0.5

超參數

名稱	類型	最小值	最大值
learning_rate	小數	0.00001	0.01
num_train_epochs	整數	10	50

 **提示：** Smart ML 訓練任務 在使用 Bayesian 或 TPE 演算法時，會基於 目標參數 的結果來反覆調校出合適參數來為作為下次訓練任務的基準。

6. 檢閱 + 建立

最後，確認填寫的資訊無誤後，就可按下建立。

2.2 啟動訓練任務

完成訓練任務的設定後，回到 訓練任務管理 頁面，可以看到剛剛建立的任務。點擊該任務，可檢視訓練任務的詳細設定。當任務的狀態顯示為 **Ready**，即可點擊 **啟動** 圖示，執行訓練任務。

訓練任務詳細資料

配置

運行列表



基本資訊

訓練任務名稱 hf-demo

命令

python3.8 /usr/src/app/training.py

描述

Hugging Face Demo

狀態

Ready

啟動後，點擊上方的「運行列表」頁籤，可以在列表中查看該任務的執行狀況與排程。在訓練進行中，可以點擊任務右方清單中的「查看日誌」或「查看詳細狀態」，來得知目前執行的詳細資訊。

2.3 檢視訓練結果

操作步驟請參閱 **AI Maker > 檢視訓練結果** (/s/ai-

maker#%E6%AA%A2%E8%A6%96%E8%A8%93%E7%B7%B4%E7%B5%90%E6%9E%9C) 文件，其中訓練任務的 Metrics 可以參考以 eval_ 開頭的項目，以本例來說即為 eval_mAP，值越大越好。

Metrics								
<input type="checkbox"/> ID↓	epoch	eval_mAP	eval_samples_per_second	eval_loss	eval_runtime	loss	learning_rate	eval_steps
<input type="checkbox"/> 9585	50.000	0.659	9.942	0.974	2.917	0.805	0.000	1.371

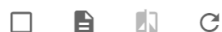
2.4 註冊模型

從一或多個結果中挑選出符合預期的運行列表，再點選右側「註冊模型」，將之儲存至模型管理中；若無符合預期結果，則重新調整環境變數與超參數的數值或數值範圍。

訓練任務詳細資料

配置

運行列表



Q 搜尋

運算開始時間 (其他+14)

ID	運算開始時間	運算結束時間	狀態	Metrics:
<input type="checkbox"/> 9952	2023-03-23 11:24	2023-03-23 12:48	Completed	註冊模型

在註冊模型視窗中，可輸入欲建立的模型目錄名稱，例如：hf-cppe5 或選擇既有的模型目錄。

註冊模型

模型目錄 *

hf-cppe5

取消

確定

儲存後，在模型管理列表中可找到該模型，點擊進入該模型的版本列表，可看到所儲存模型的版本、類別、描述與來源等資訊。

3. 建立推論服務

當您訓練好物件偵測任務網路，並儲存訓練好的模型後，即可藉由 **推論** 功能將其部署至應用程式或服務執行推論。

3.1 建立推論任務

首先點選左側服務列表「**推論**」，進入推論管理頁面，並按下「**+ 建立**」，建立一個推論服務。推論任務的建立步驟說明如下：

1. 基本資訊

首先先將 **建立方式** 改成 **自訂**，與前面設定相似，我們也是使用 **huggingface-object-detection** 的範本進行套用。不過，所要載入的模型名稱與版號仍須使用者手動設定。

- **名稱**
載入後模型的檔案名稱，使用者可自行輸入，本範例為 `model`。
- **模型名稱**
所要載入模型的名稱，即我們在 **2.4 註冊模型** 中所儲存的模型。
- **版本**
所要載入模型的版號，亦是 **2.4 註冊模型** 中所產生的版號。
- **掛載位置**
載入後模型所在位置，與程式進行中的讀取有關。這值會由 **huggingface-object-detection** 推論範本設定。



提示： 名稱不可重複，請自行命名。

建立推論

1

2

3

4

5

6

基本資訊

硬體設定

儲存設定

變數設定

進階設定

檢閱 + 建立

名稱 *

hf-demo

描述

Hugging Face Demo

建立方式 *

☐ 自動偵測 ☒ 自訂

選擇範本

選擇

huggingface-object-detection

來源模型 *

名稱	模型名稱	模型版本	掛載位置
<div>model</div>	<div>hf-pppe5</div>	<div>v1</div>	<div>/output</div>
<div>新增</div>			

映像檔來源 *

選擇

huggingface

映像檔標籤 *

v1

2. 硬體設定

參考目前的可用配額與需求，從列表中選出合適的硬體資源。如果想要推論服務的反應較為即時，請選擇包含 **GPU** 的規格。

3. 儲存設定

此步驟無須設定。

4. 變數設定

在變數設定步驟，這些慣用的指令與參數，會在套用範本時自動帶入。

建立推論

✓

基本資訊

✓

硬體設定

✓

儲存設定

4

變數設定

5

進階設定

6

檢閱 + 建立

5. 檢閱 + 建立

最後，確認填寫的資訊，就可按下建立。

💡 提示：

除了採用範本方式套用外，系統也額外提供自動偵測方法，按下 **自動偵測** 按鈕，模型類型會顯示為 **huggingface**，而推論伺服器則會採用 **HuggingFace Server**。

建立方式 *

☒ 自動偵測 ☐ 自訂

來源模型 *

模型名稱

模型版本

hf-clip5

v1

偵測

支援類型清單 [🔗](#)

模型類型 *

huggingface

推論伺服器 *

HuggingFace Server








3.2 進行推論

設置任務完成後，請到該服務的推論詳細設定確認是否成功啟動。當服務的狀態顯示為 **Ready** 時，即可以開始連線到推論服務進行推論。

推論詳細資料

配置

監控



上次更新 2022-10-31 21:59:22

基本資訊

推論名稱

hf-demo

命令

/start.sh

描述

Hugging Face Demo

狀態

Ready

目前推論服務為了安全性考量沒有開放對外埠服務，但我們可以透過 **筆記本服務** 來跟已經建立好的推論服務溝通，溝通的方式就是靠 **推論詳細資料** 頁面下方 **網路** 資訊所顯示的網址。

網路					
	連接埠	對外服務	對外埠	服務協定	網址
	9999	null		TCP	http://hf-demo-i.36e81d89-0c4... 



提示：推論服務網址

- 為安全性考量，目前推論服務提供的 **網址** 僅能在系統的內部網路中使用，無法透過外部的網際網路存取。
- 若要對外提供此推論服務，請參考 **AI Maker > 對外提供服務** (/s/ai-maker/#E9%80%B2%E8%A1%8C%E6%8E%A8%E8%AB%96) 說明。

若要查看推論監控，可點擊「**監控**」頁籤，即可在監控頁面看到相關資料，下圖為經過一段時間後的推論結果。

推論詳細資料

配置

監控



上次更新 2022-11-17 17:14:11

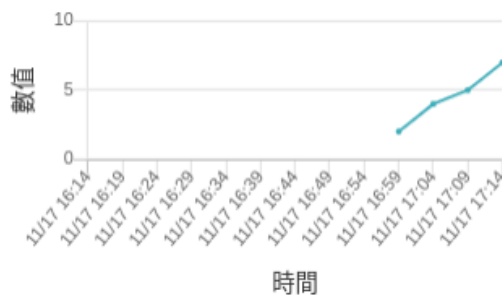
hf-demo 監控

1 小時

期間: 2022-11-17 16:14 ~ 2022-11-17 17:14

推論 API 計數

總 API 請求次數: 7



點選期間選單可篩選特定期間呼叫推論 API 的統計資訊，例如：1 小時、3 小時、6 小時、12 小時、1 天、7 天、14 天、1 個月、3 個月、6 個月、1 年或自訂。

推論詳細資料

配置

監控



上次更新 2022-10-31 21:59:22

hf-demo 監控

1 小時

期間: 2022-10-31 20:59 ~ 2022-10-31 21:59

💡 關於觀測期間的起始與結束時間

例如當前時間為 15:10，則：

- 1 小時 的範圍是指 15:00 ~ 16:00（並非指過去一小時 14:10 ~ 15:10）
- 3 小時 的範圍是指 13:00 ~ 16:00
- 6 小時 的範圍是指 10:00 ~ 16:00
- 以此類推

3.2.1 使用 curl 指令測試推論服務

如要測試推論服務，可使用 **筆記本服務** (/s/notebook)，其中套用 PyTorch 的開發框架，例如 PyTorch-21.02-py3，藉由 curl 來呼叫 API 進行推論：

```
!curl -X POST hf-demo-i.36e81d89-0c43-4e89-a7d8-a58705042436:9999 -T "cppe5/val
```

當 server 端收到影像後會進行物件偵測，最終回傳所辨識出的物件類別、分數以及四點坐標位置

```
[{"score":0.9337628483772278,"label":"2","box":  
{"xmin":1142,"ymin":148,"xmax":1354,"ymax":345}},  
{"score":0.9408917427062988,"label":"2","box":  
{"xmin":649,"ymin":158,"xmax":849,"ymax":385}},  
{"score":0.9733251929283142,"label":"0","box":  
{"xmin":314,"ymin":78,"xmax":1710,"ymax":1077}},  
{"score":0.9843624830245972,"label":"4","box":  
{"xmin":893,"ymin":320,"xmax":1112,"ymax":545}}]
```

3.2.2 使用 Python 程式執行推論服務

除了使用 curl，也可以透過 **筆記本服務** (/s/notebook) 加上 PyTorch 的開發框架，例如 PyTorch-21.02-py3，來啟動 JupyterLab 與推論服務進行連線，程式碼範例如下說明：

1. 發送請求

在這邊使用 requests 模組產生 HTTP 的 POST 請求。其中 **endpoint** 變數需要填入推論服務的網址連結。

```
1 import json
2 import requests
3 import matplotlib.pyplot as plt
4 from PIL import Image
5
6 # colors for visualization
7 COLORS = [[0.000, 0.447, 0.741], [0.850, 0.325, 0.098], [0.929, 0.694,
8             [0.494, 0.184, 0.556], [0.466, 0.674, 0.188], [0.301, 0.745,
9 id2label = ['coveralls', 'mask', 'face shield', 'gloves', 'goggles']
10
11 def predict(filename):
12     endpoint = "http://hf-demo-i.36e81d89-0c43-4e89-a7d8-a58705042436
13     data = open(filename, 'rb').read()
14     return requests.post(endpoint, data=data)
15
16 def plot(image, result):
17     plt.figure(figsize=(16,10))
18     plt.imshow(image)
19     ax = plt.gca()
20     colors = COLORS * 100
21     i = 0
22     for data in result:
23         xmin = data['box']['xmin']
24         ymin = data['box']['ymin']
25         xmax = data['box']['xmax']
26         ymax = data['box']['ymax']
27         label = id2label[int(data['label'])]
28         score = data['score']
29         ax.add_patch(plt.Rectangle((xmin, ymin), xmax - xmin, ymax - y
30         text = f'{label}: {score:0.2f}'
31         ax.text(xmin, ymin, text, fontsize=15, bbox=dict(facecolor='ye
32         i = (i + 1) % len(COLORS)
33     plt.axis('off')
34     plt.show()
35
36 path = "cppe5/val/000001.JPEG"
37 response = predict(path)
38 plot(Image.open(path), response.json())
```

2. 取回結果

完成物件偵測辨識後，結果將以 JSON 格式回傳，透過程式將物件對應的名稱、分數以及位置在影片中標示出來，本推論程式僅會回傳分數大於 0.9 的物件。

