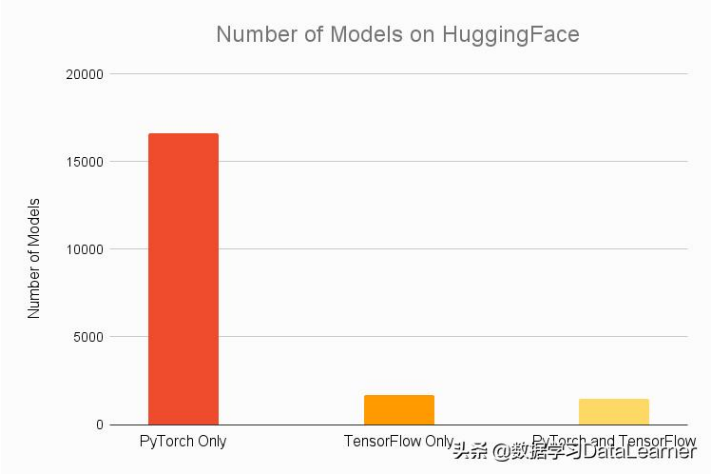
1-2 為什麼要使用 PyTorch PyTorch 如何協助深度學習專案的開發

黃志勝（Tommy Huang）
義隆電子 人工智慧研發部
國立陽明交通大學 AI學院 合聘助理教授
國立台北科技大學 電資學院合聘助理教授

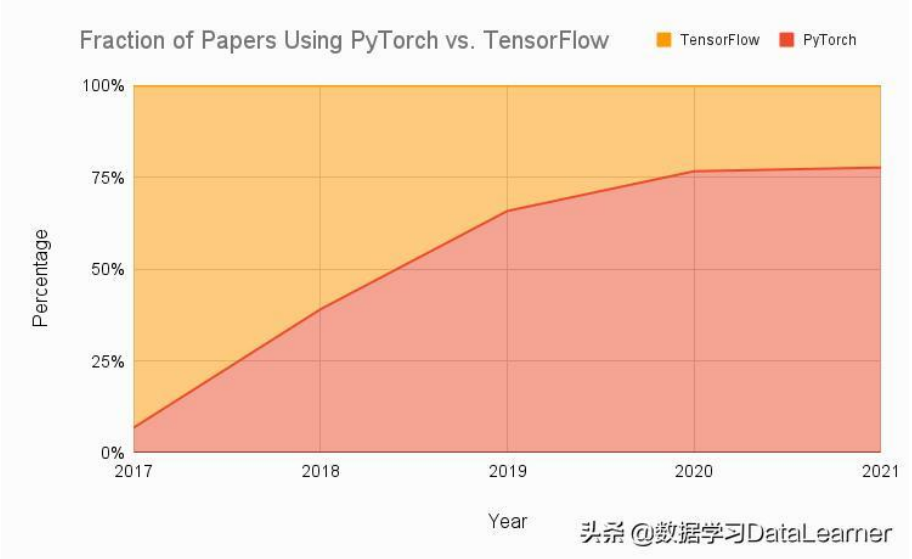
為什麼選PyTorch

PyTorch vs TensorFlow

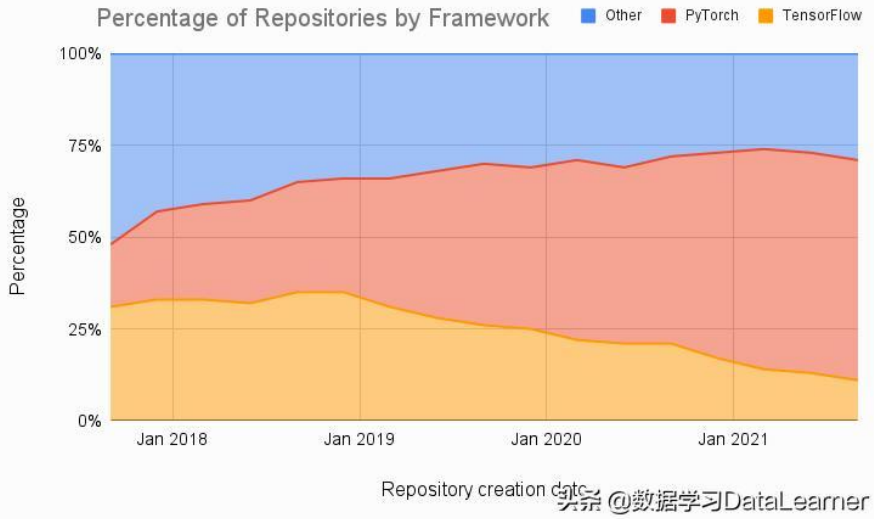
HuggingFace



Gradient上的頂級會議論文收集

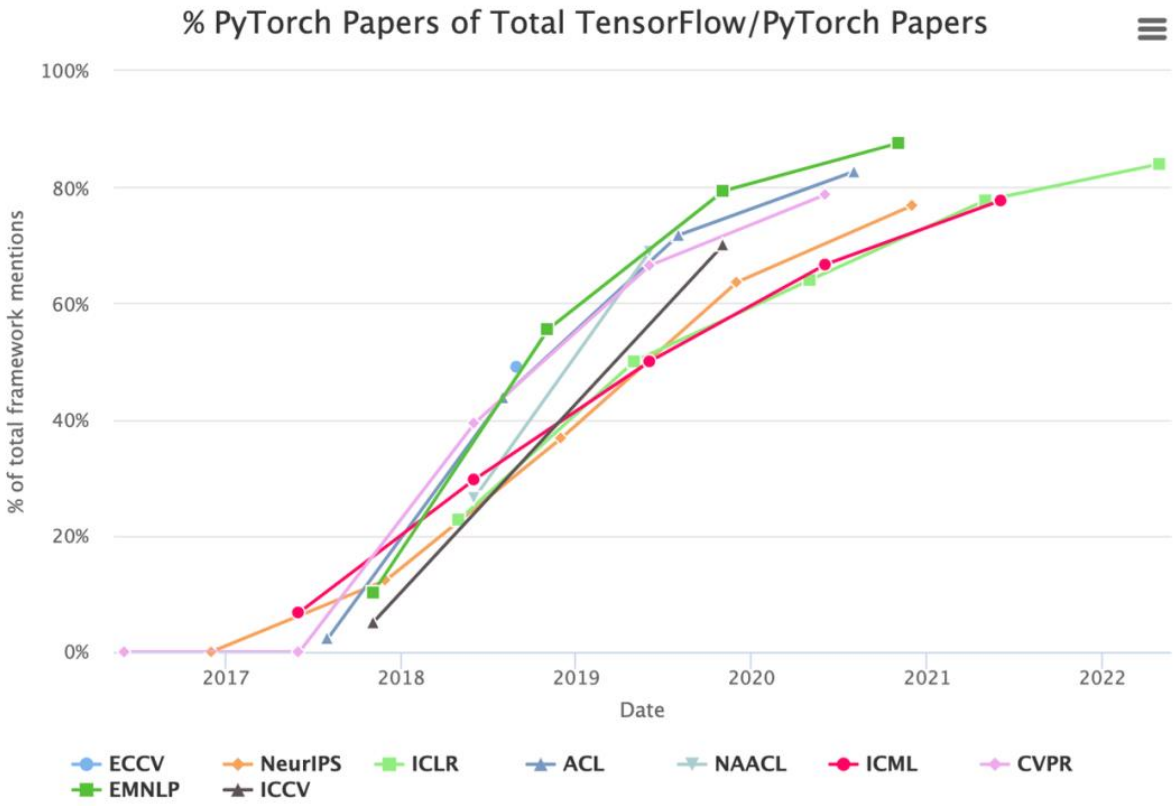


PapersWithCode上的使用率

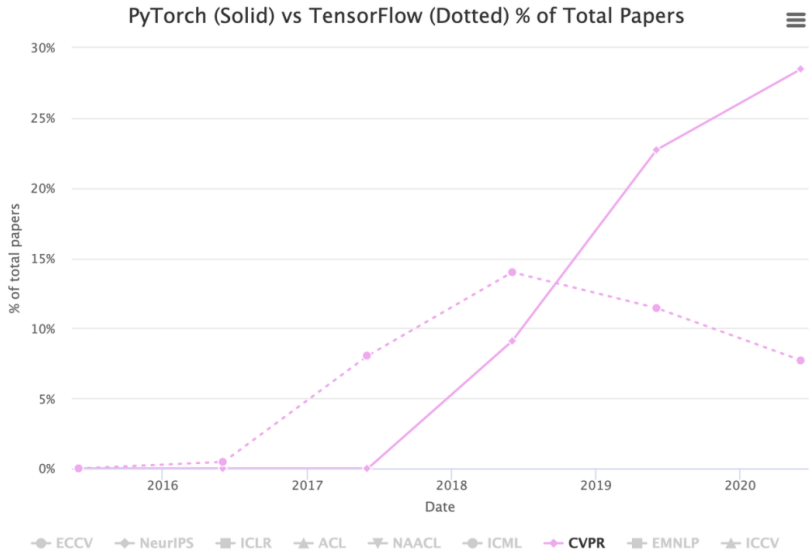


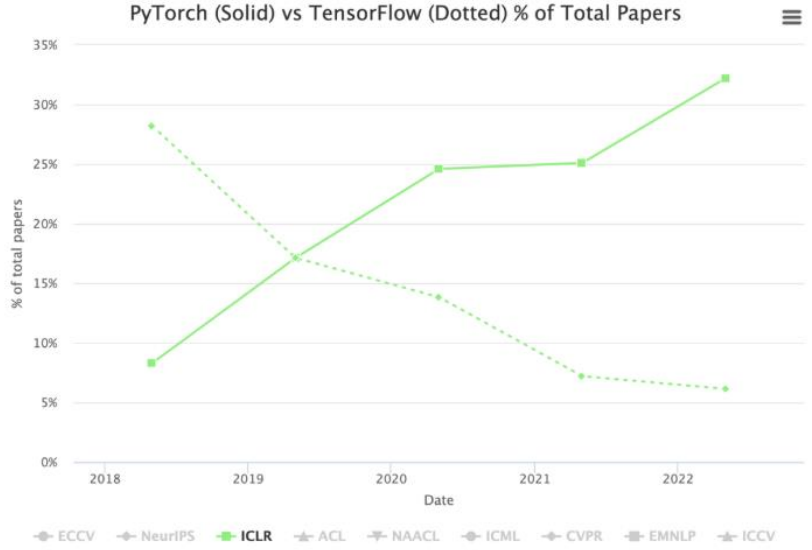
https://inf.news/tech/e408b4c7ef075f836809f3a0daaee274.html

PyTorch vs TensorFlow



https://cloud.tencent.com/developer/article/1957310





PyTorch vs TensorFlow

- **易於使用**：
  PyTorch以其簡潔的API和直觀的語法聞名。
- **動態圖（Dynamic Computational Graph）：**
  PyTorch使用動態圖的方式，這意味著在運行時可以即時更改模型結構和調整參數，方便調試和快速迭代。相比之下，TensorFlow到2.0版本才導入動態建模，但2.0和1.X不相容。
- **強大的社群支持和研究界的廣泛應用：**
  PyTorch擁有一個活躍的開源社群，在許多研究機構和學術界都使用PyTorch進行深度學習研究。
- **自然語言處理（NLP）領域的強大支持：**
  PyTorch在自然語言處理領域廣受歡迎，並具有許多流行的NLP庫，例如Transformers和TorchText。這使得PyTorch成為NLP研究和應用的首選框架。

PyTorch vs TensorFlow

- **專案需求：**
  根據你的項目特點，如是否需要高性能計算、低級別的操作控制、分佈式訓練等，確定哪個框架更適合滿足你的需求。
- **社群和資源支持：**
  考慮框架的社群支持和資源可用性，包括開源項目、課程、文檔和社群討論。這些資源對於解決問題和學習新技術非常重要。
- **學習曲線：**
  評估你對深度學習框架的熟悉程度和學習曲線。如果你是初學者，PyTorch可能會更容易上手。如果你已經熟悉TensorFlow或有相關的經驗，那麼繼續使用它可能更合適。
- **領域特定應用：**
  考慮你是否專注於某個特定的領域，如計算機視覺、自然語言處理或聲音識別。了解每個框架在該領域中的工具和開源碼的支持情況。
- **團隊合作：**
  如果你正在與團隊合作，確保詢問團隊成員對框架的偏好和經驗。這有助於確定使用哪個框架能夠提供更好的協作和共享代碼的能力。

PyTorch

語法跟numpy差異不大，容易上手

```
import numpy as np
a = np.array([[1,2,3],
              [4,5,6]])
b = np.array([[2,2,2],
              [3,3,3]])
c = np.array([[1,2],
              [3,4],
              [5,6]])
print('元素點對點相乘(方法1:np.multiply(a,b)):\\\
       \n{}'.format(np.mult
print('元素點對點相乘(方法2:a
       \n{}'.format(a*b))
print('矩陣相乘(方法1: np.dot
       \n{}'.format(np.dot(
print('矩陣相乘(方法2: a.dot(
       \n{}'.format(a.dot(c
print('矩陣相乘(方法3: np.mat
       \n{}'.format(np.matr
```

```
元素點對點相乘(方法1:np.multiply(a,b)):\
[[ 2  4  6]
 [12 15 18]]
元素點對點相乘(方法2:a*b):
[[ 2  4  6]
 [12 15 18]]
矩陣相乘(方法1: np.dot(a,c)):
[[22 28]
 [49 64]]
矩陣相乘(方法2: a.dot(c)):
[[22 28]
 [49 64]]
矩陣相乘(方法3: np.matmul(a,c)):
[[22 28]
 [49 64]]
```

```
import torch
a = torch.tensor([[1,2,3],
                  [4,5,6]])
b = torch.tensor([[2,2,2],
                  [3,3,3]])
c = torch.tensor([[1,2],
                  [3,4],
                  [5,6
('元素點對點相乘(a
   \n{}'.format(a
('矩陣相乘(方法1:
   \n{}'.format(
('矩陣相乘(方法2:
   \n{}'.format(
('矩陣相乘(方法3:
   \n{}'.format(
```

```
元素點對點相乘(a*b):\
tensor([[ 2,  4,  6],
        [12, 15, 18]])
矩陣相乘(方法1: torch.mm(a,c)):\
tensor([[22, 28],
        [49, 64]])
矩陣相乘(方法2: torch.matmul(a,c)):\
tensor([[22, 28],
        [49, 64]])
矩陣相乘(方法3: a.matmul(c)):\
tensor([[22, 28],
        [49, 64]])
```

PyTorch

語法跟numpy差異不大，容易上手

```
import numpy as np
import torch
a_np = np.array([[6.0, 2.0],
                 [4.0, 5.0]])
a_torch = torch.tensor(a_np)
print(np.linalg.norm(a_np))
print(torch.linalg.norm(a_torch))

print(np.linalg.inv(a_np))
print(torch.linalg.inv(a_torch))
```

```
9.0
tensor(9., dtype=torch.float64)
[[ 0.22727273 -0.09090909]
 [-0.18181818  0.27272727]]
tensor([[ 0.2273, -0.0909],
        [-0.1818,  0.2727]], dtype=torch.float64)
```

Google

GPT training github

影片 圖片 書籍 新聞 購物 地圖 航班 財經

約有 2,970,000 項結果 (搜尋時間：0.37 秒)

github.com
https://github.com › karpathy › minGPT · 翻譯這個網頁

karpathy/minGPT - GitHub

A PyTorch re-implementation of GPT, both training and inference. minGPT tries to be small, clean, interpretable and educational, as most of the currently ...
minGPT/model.py · README.md · Demo.ipynb · Generate.ipynb

github.com
https://github.com › karpathy › nanoGPT · 翻譯這個網頁

karpathy/nanoGPT: The simplest, fastest repository ... - GitHub

2022年12月29日 — The simplest, fastest repository for training/finetuning medium-sized GPTs. It is a rewrite of minGPT that prioritizes teeth over education.
minGPT · Training on M1 "MPS" #28 · nanoGPT/model.py at master · Train.py

github.com
https://github.com › Lightning-AI › ligh... · 翻譯這個網頁

lightning-GPT - GitHub

Train and run GPTs with Lightning. Contribute to Lightning-Universe/lightning-GPT development by creating an account on GitHub.

github.com
https://github.com › microsoft › PyCode... · 翻譯這個網頁

microsoft/PyCodeGPT: A pre-trained GPT model for ... - GitHub

we aims to train median-large pre-trained models (model size with 110M) based on GPT-Neo: PyCodeGPT-110M: derived from GPT-Neo 125M with a vocabulary size ...

github.com
https://github.com › fattorib › Little-GPT · 翻譯這個網頁

GPT* - Training faster small transformers using ALiBi ... - GitHub

GPT* is a collection of transformer models based on GPT2-Small, GPT2-Medium, and GPT2-XL with the following architecture modifications to speed up training and ...

github.com
https://github.com › UCLA-DM › GPT-... · 翻譯這個網頁

GPT-GNN: Generative Pre-Training of Graph Neural Networks

GPT-GNN is a pre-training framework to initialize GNNs by generative pre-training. It can be applied to large-scale and heterogensous graphs. You can see our ...

github.com/karpathy/nanoGPT

README.md

Dependencies:

- pytorch <3
- numpy <3
- `pip install transformers` for huggingface transformers <3 (to load GPT-
- `pip install datasets` for huggingface datasets <3 (if you want to downl
  OpenWebText)
- `pip install tiktoken` for OpenAI's fast BPE code <3
- `pip install wandb` for optional logging <3
- `pip install tqdm` <3

main Little-GPT / requirements

Ben new bnb version

Code Blame 16 lines (16 loc) · 301 Bytes

```
wandb==0.12.21
einops==0.4.1
tokenizers==0.12.1
hydra-core==1.2.0
tqdm==4.64.0
black==21.9b0
boto3==1.20.54
accelerate==0.10.0
transformers==4.20.1
webdataset==0.2.5
jsonlines==2.0.0
gradio==3.0.23
--extra-index-url https://download.pytorch.org/whl/cu113
torch==1.11.0+cu113
pandas==1.4.3
bitsandbytes
```

main lightning-GPT / requirements.txt

lantiga Fix packages

Code Blame 3 lines (3 loc) · 37 Bytes

```
lightning>=1.8.0
torch>=1.10.0
numpy
```

main PyCodeGPT / requirements.txt

substill Add files via upload

Code Blame 4 lines (4 loc) · 44 Bytes

```
torch
transformers
sentencepiece
```

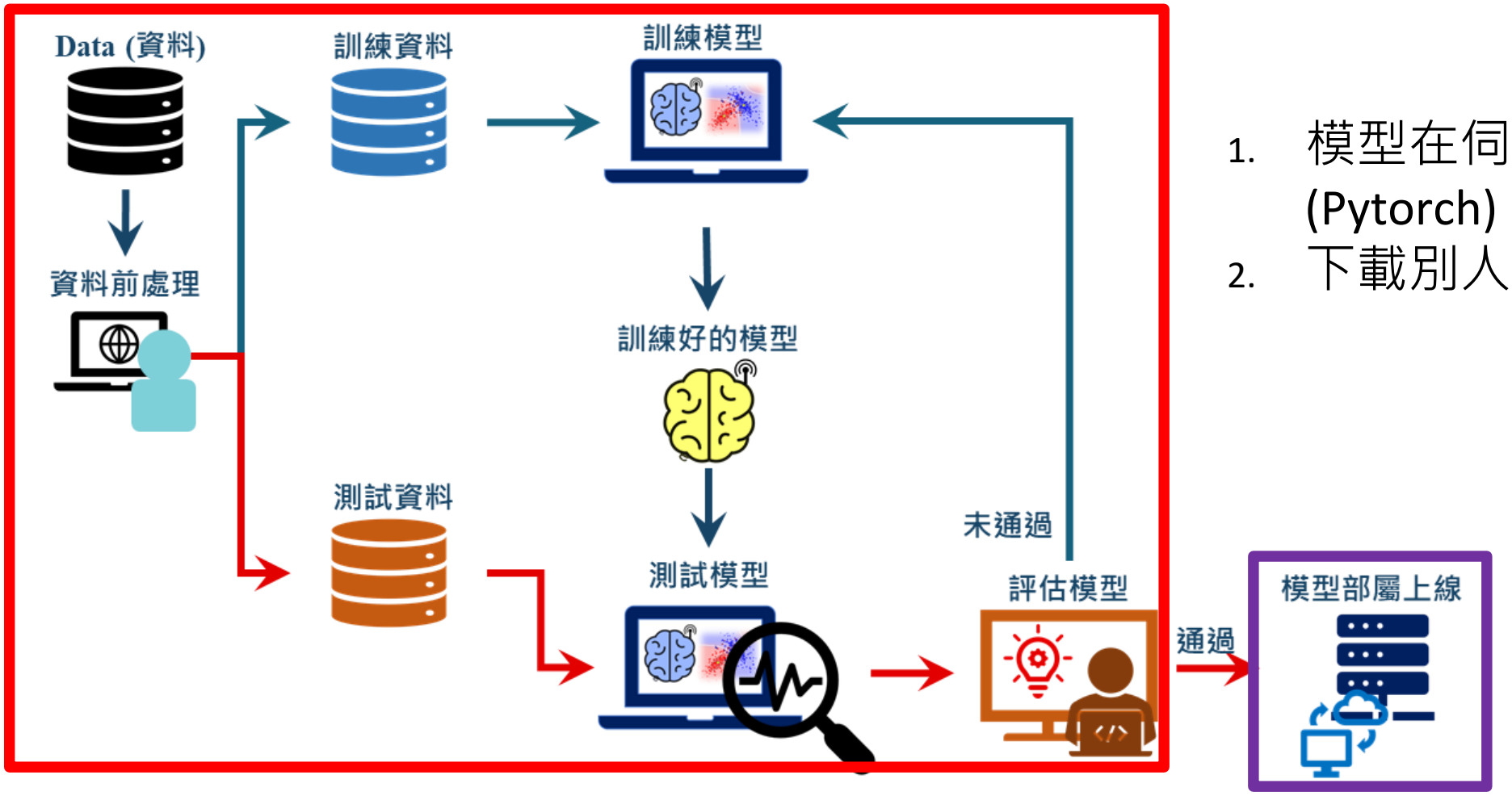
master GPT-GNN / requirements.txt

dependabot[bot] Bump numpy from 1.16.2 to 1.22.0

Code Blame 14 lines (14 loc) · 249 Bytes

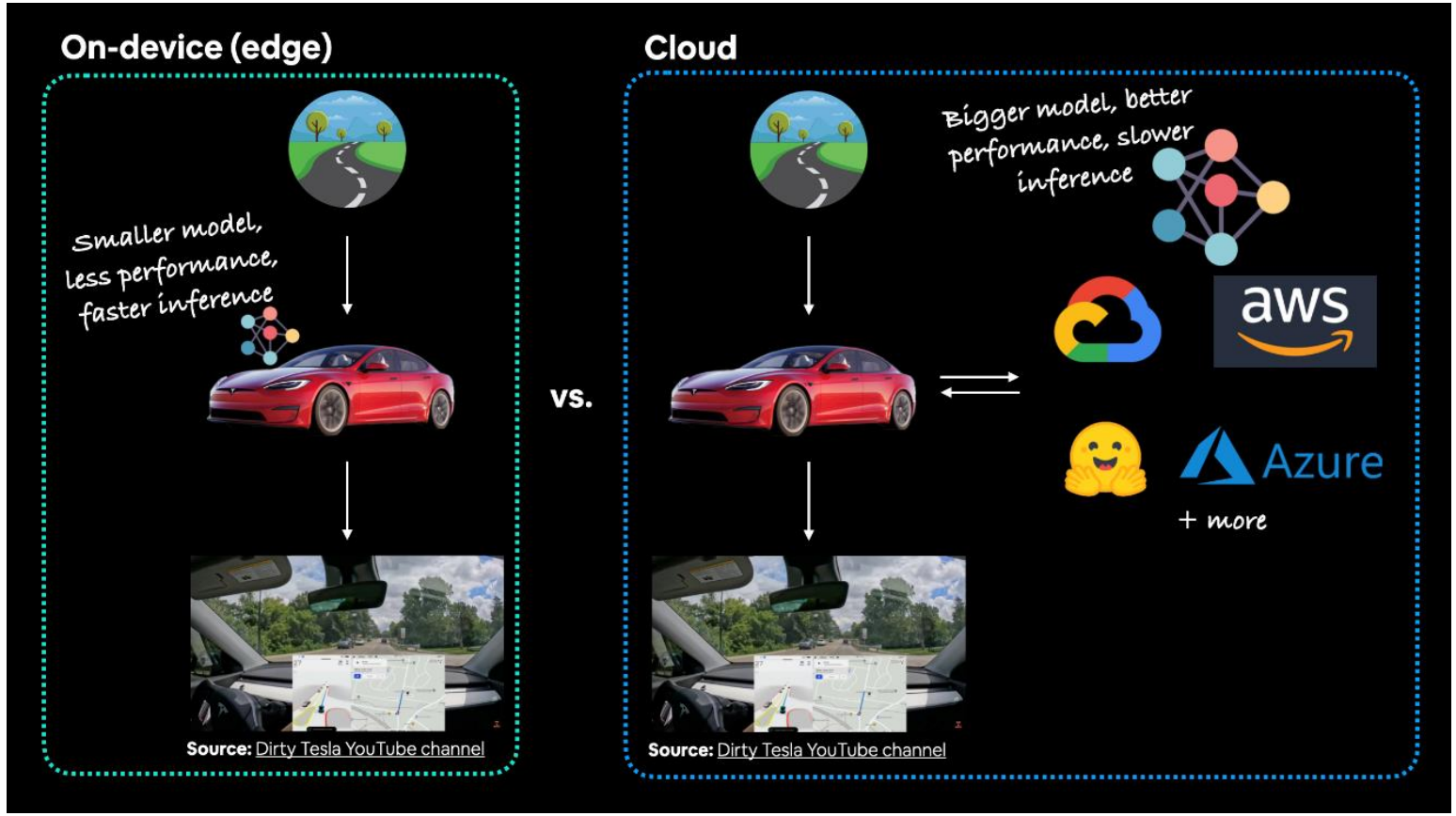
```
dill==0.3.0
numpy==1.22.0
pandas==0.24.2
torch==1.3.0
torch-scatter==1.3.2
torch-cluster==1.4.5
torch-sparse==0.4.3
torch-spline-conv==1.1.1
torch-geometric==1.3.2
torchvision==0.4.1
tqdm==4.31.1
seaborn==0.9.0
matplotlib==3.0.3
transformers==2.8.0
```

PyTorch 如何協助深度學習專案的開發



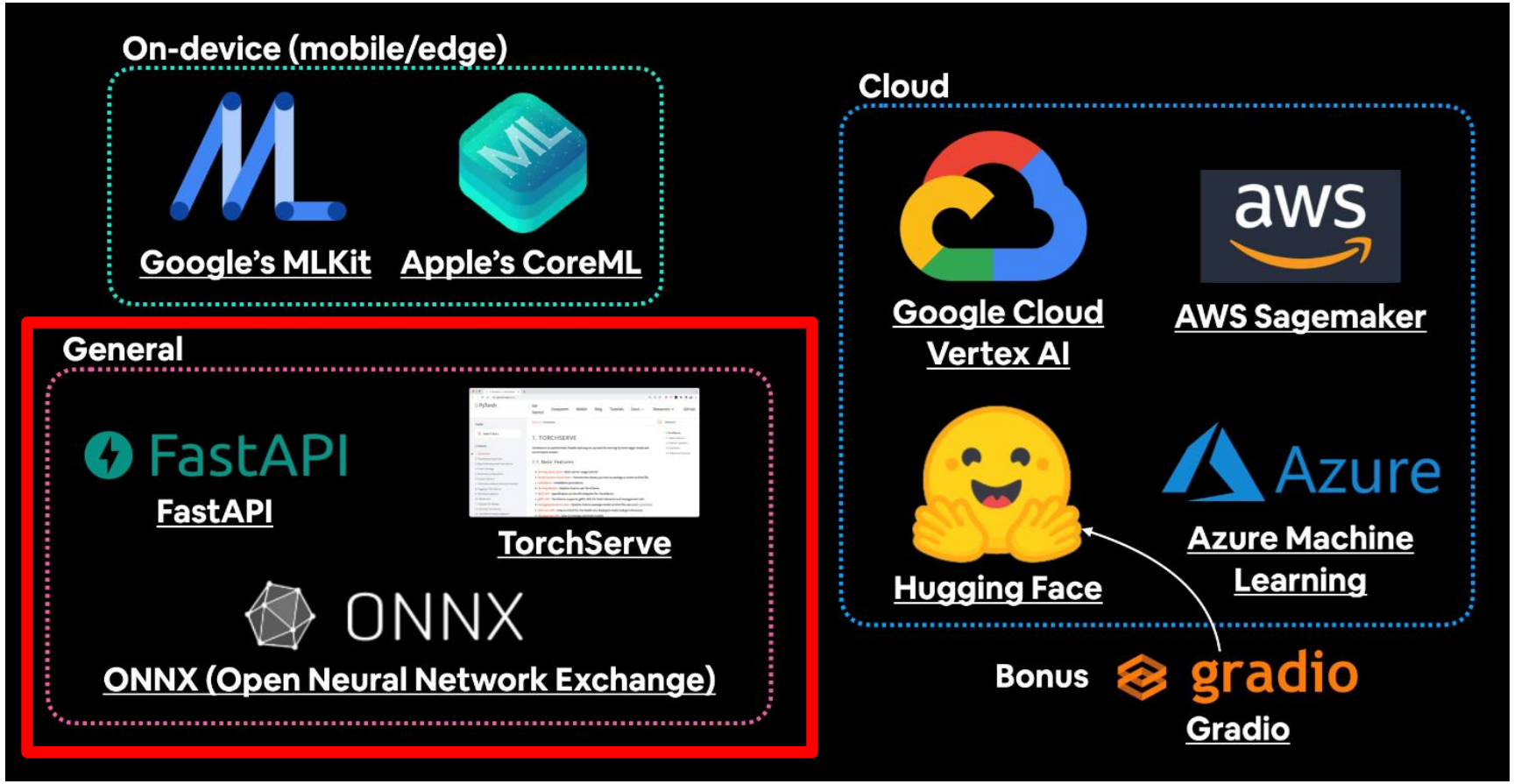
1. 模型在伺服器上訓練 (Pytorch)
2. 下載別人訓練好的模型。

PyTorch 如何協助深度學習專案的開發



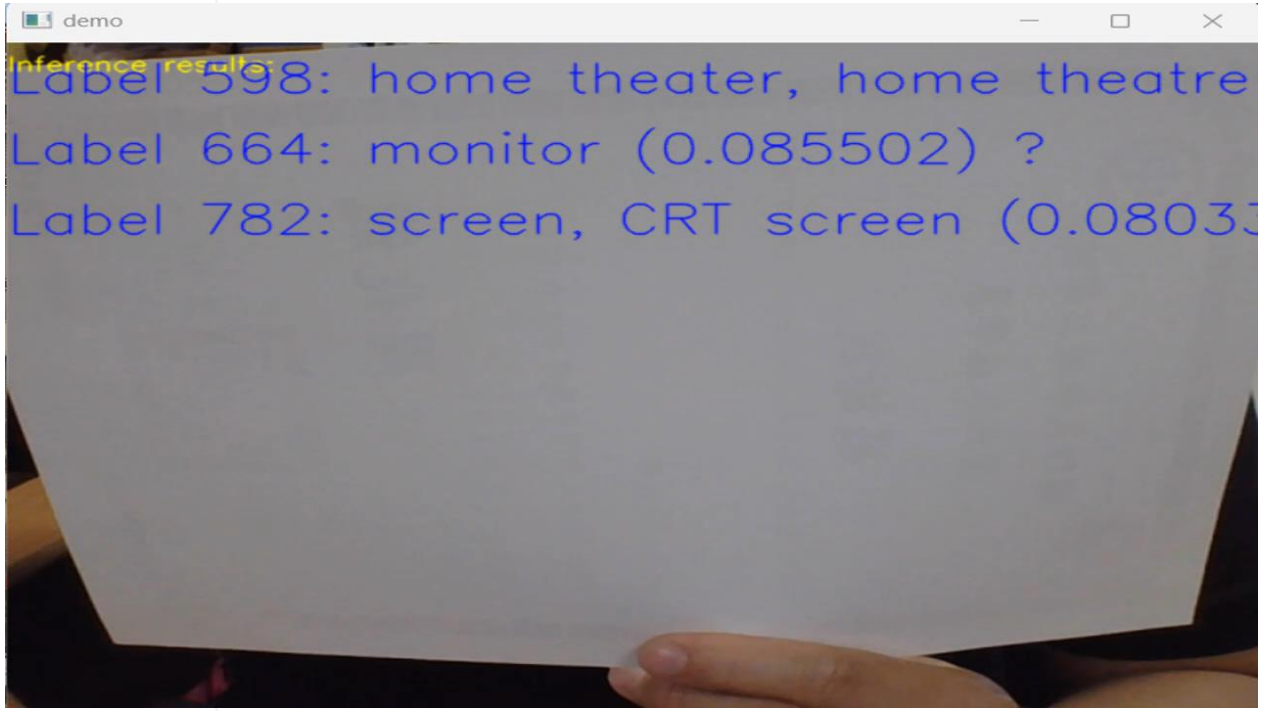
https://www.learnpytorch.io/09_pytorch_model_deployment/

PyTorch 如何協助深度學習專案的開發



```python
import cv2
import numpy as np
from PIL import Image
import onnxruntime as ort
import torchvision.transforms as trns
onnxmodel_path='./weight/mobilenetv2.onnx'
class_def = './weight/imagenet_classes.txt'
def softmax(x):
    x = x.reshape(-1)
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum(axis=0)
def postprocess(result):
    return softmax(np.array(result)).tolist()
def main():
    # Run the model on the backend
    session = ort.InferenceSession(onnxmodel_path, None)
    # get the name of the first input of the model
    input_name = session.get_inputs()[0].name
    # Load ImageNet classes
    with open(class_def) as f:
        classes = [line.strip() for line in f.readlines()]
    # Define image transforms
    transforms = trns.Compose([trns.Resize((224, 224)),
                               trns.ToTensor(),
                               trns.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
    cap = cv2.VideoCapture(0)
    while True:
        ret, img = cap.read()
        if not ret:
            break
        # Read image and run prepro
        image = Image.fromarray(img)#.convert("RGB")
        image_tensor = transforms(image)
        image_tensor = image_tensor.unsqueeze(0)
        image_np = image_tensor.numpy()
        # model run
        outputs = session.run([], {input_name: image_np})[0]
        print("Output size:{}".format(outputs.shape))
        # Result postprocessing
        idx = np.argmax(outputs)
        sort_idx = np.flip(np.squeeze(np.argsort(outputs)))
        idx = np.argmax(outputs)
        # outputs = np.sort(outputs[0,:])
        probs = postprocess(outputs)
        top_k=3
        cv2.putText(img, "Inference results:", (0, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1, cv2.LINE_AA)
        print("Inference results:")
        for i, index in enumerate(sort_idx[:top_k]):
            py = 35 + 50*i
            text = "Label {}: {} ({:5f}) \n".format(index, classes[index],probs[index])
            cv2.putText(img, text, (0, py), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)
            print(text)
        cv2.imshow('demo', img)
        cv2.waitKey(1)
    cap.release()Z
if __name__ == '__main__':
    main()
```

Onnxruntime部屬訓練好的模型



Inference :
TensorRT (Nvidia專屬)