

# 為什麼要使用 PyTorch

## ONNX 簡介

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

Keras

theano

 Microsoft | Cognitive Toolkit

 PaddlePaddle


  
TensorFlow

 mxnet

 DL4J

Caffe  Caffe2

PYTORCH

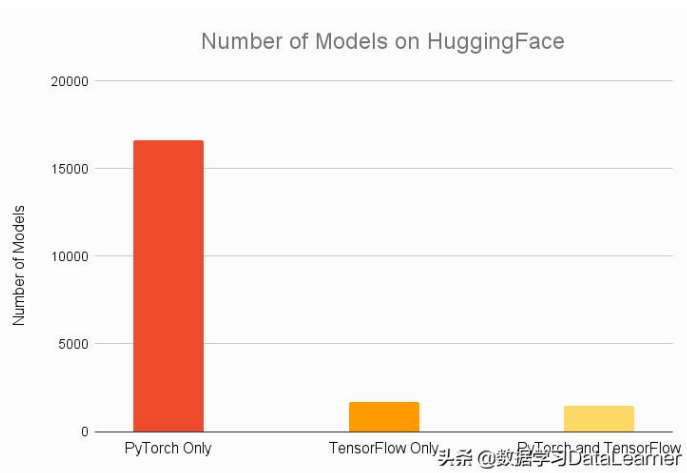
 Chainer



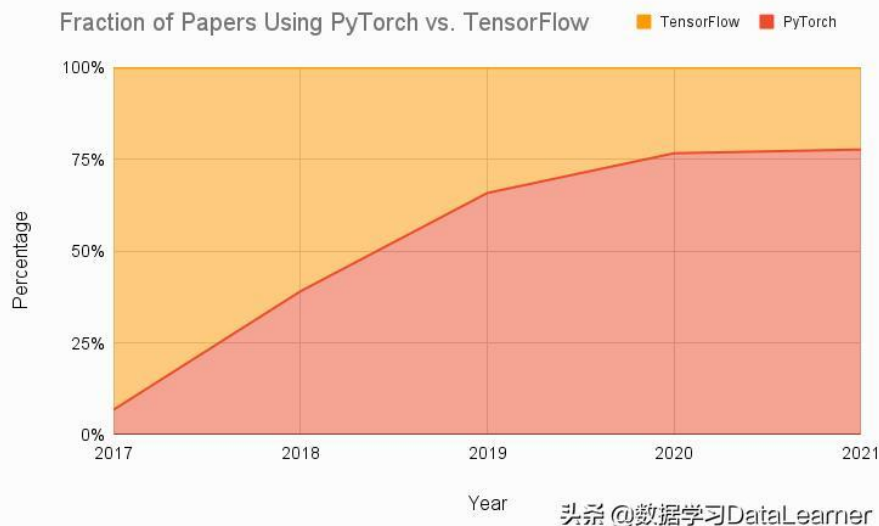
# PyTorch vs TensorFlow

Gradient上的頂級會議論文收集

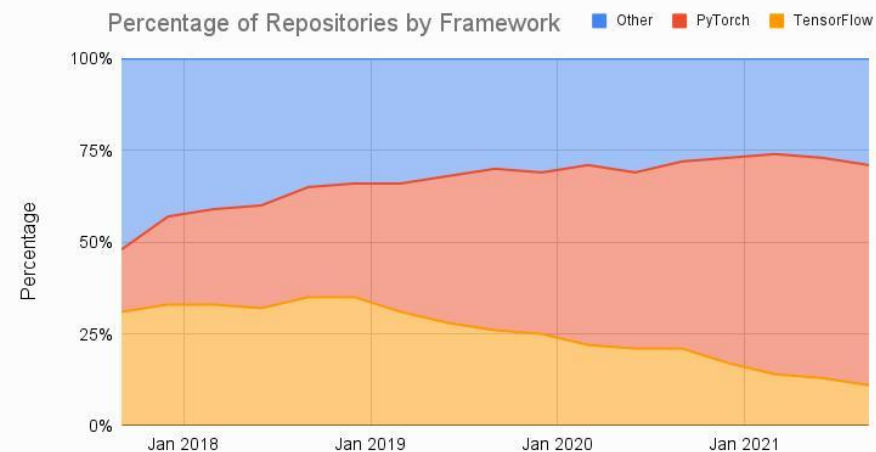
HuggingFace



Fraction of Papers Using PyTorch vs. TensorFlow

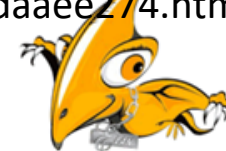


PapersWithCode上的使用率

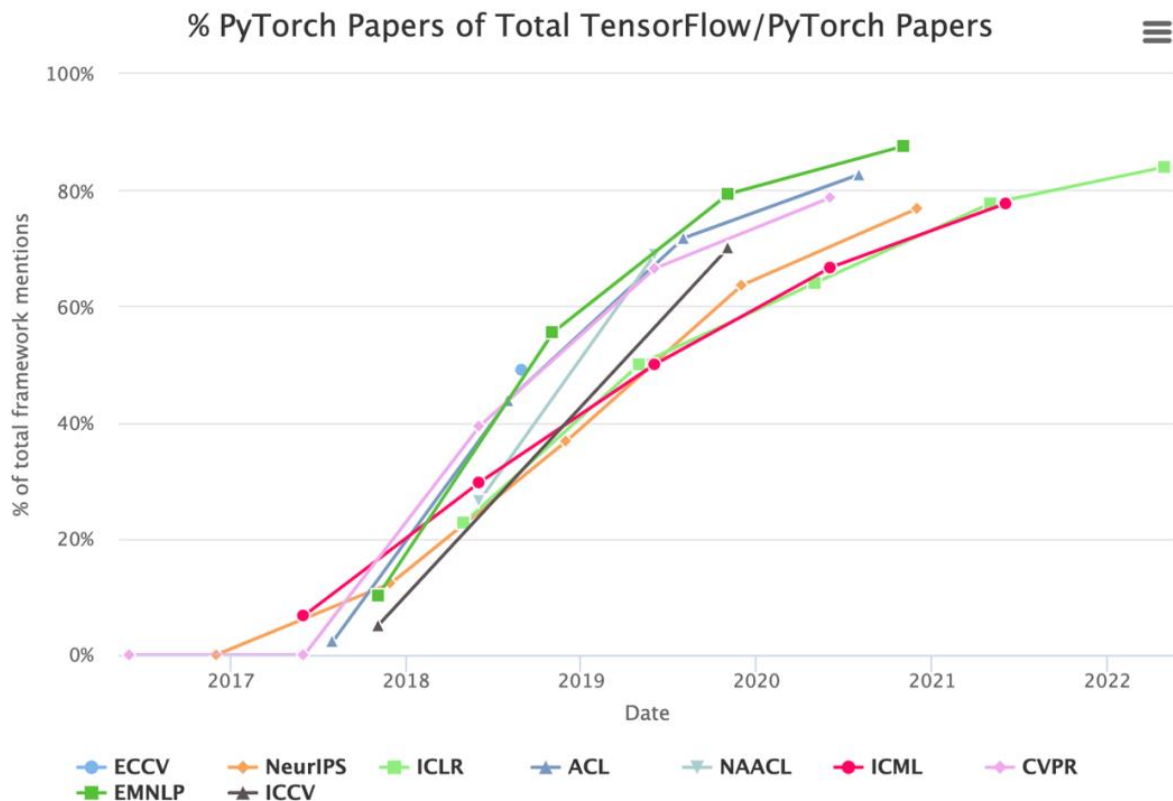


Repository creation

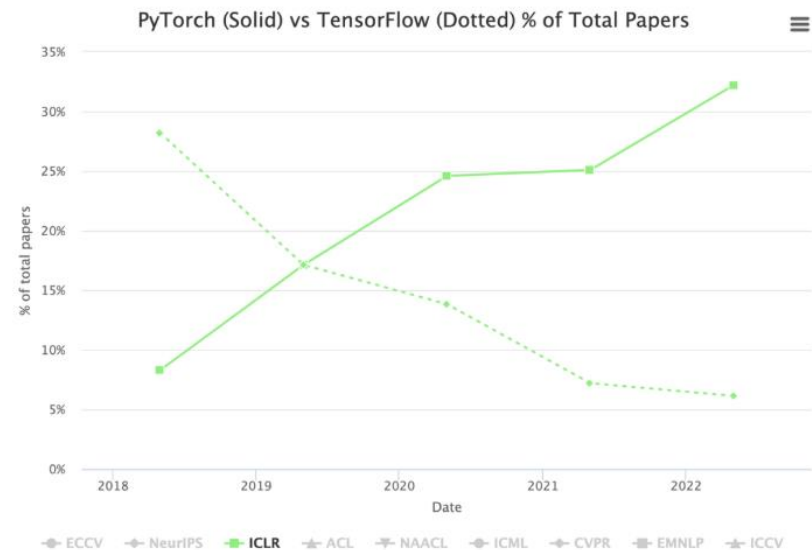
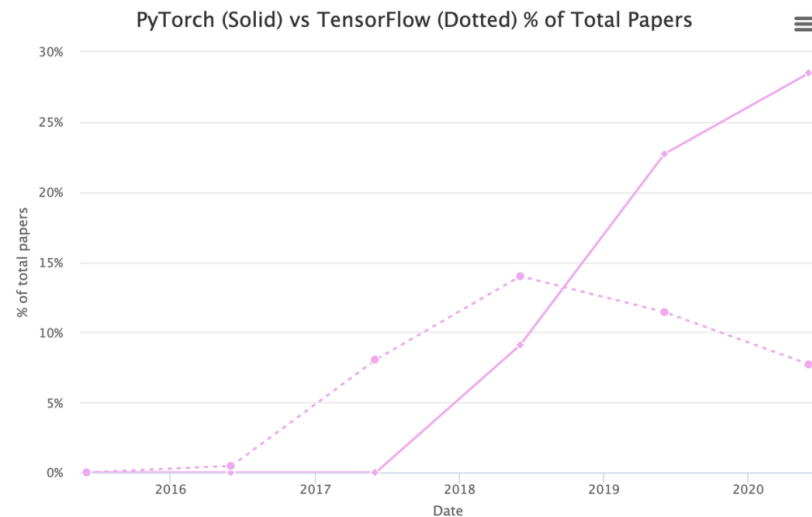
<https://inf.news/tech/e408b4c7ef075f836809f3a0daaee274.html>



# 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。



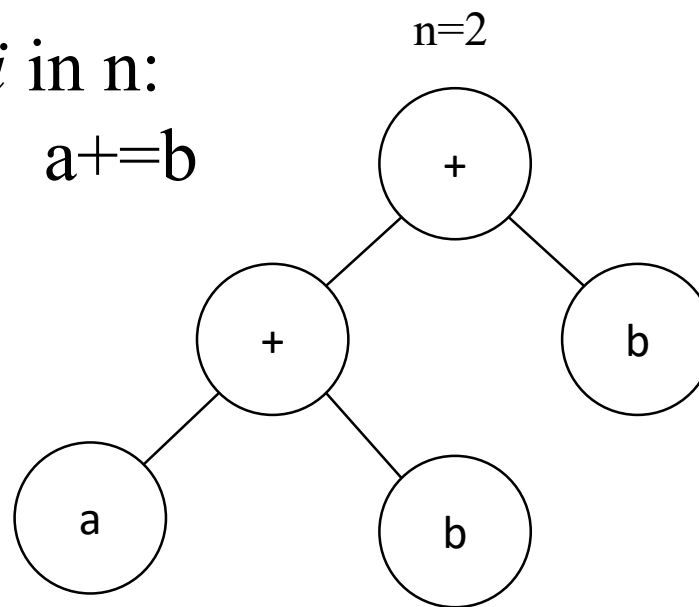
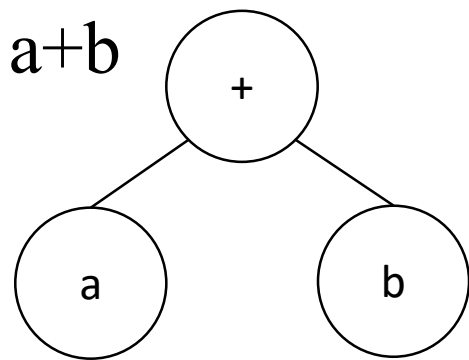
# 靜態圖 vs 動態圖

為了加速運算，有些框架會使用對神經網路「先編譯，後執行」的靜態圖來描述網路。

n是可變參數

for  $i$  in  $n$ :

$a += b$



除了 for-loop，if-else 也很會因為條件不同產生不同的計算圖



# 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{\\n}'.format(np.multiply(a,b)))
print('元素點對點相乘(方法2: a*b)\\n{\\n}'.format(a*b))
print('矩陣相乘(方法1: np.dot(a,b))\\n{\\n}'.format(np.dot(a,b)))
print('矩陣相乘(方法2: a.dot(b))\\n{\\n}'.format(a.dot(b)))
print('矩陣相乘(方法3: np.matmul(a,b))\\n{\\n}'.format(np.matmul(a,b)))

print('元素點對點相乘(方法1: np.multiply(a,b)):\\n{\\n}'.format(np.multiply(a,b)))
print('元素點對點相乘(方法2: a*b)\\n{\\n}'.format(a*b))
print('矩陣相乘(方法1: np.dot(a,b)):\\n{\\n}'.format(np.dot(a,b)))
print('矩陣相乘(方法2: a.dot(b)):\\n{\\n}'.format(a.dot(b)))
print('矩陣相乘(方法3: np.matmul(a,b)):\\n{\\n}'.format(np.matmul(a,b)))
```

```
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]])

print('元素點對點相乘(a*b):\\n{\\n}'.format(torch.multiply(a,b)))
print('矩陣相乘(方法1: torch.mm(a,b)):\\n{\\n}'.format(torch.mm(a,b)))
print('矩陣相乘(方法2: torch.matmul(a,b)):\\n{\\n}'.format(torch.matmul(a,b)))
print('矩陣相乘(方法3: a.matmul(b)):\\n{\\n}'.format(a.matmul(b)))

print('元素點對點相乘(a*b):\\n{\\n}'.format(torch.multiply(a,b)))
print('矩陣相乘(方法1: torch.mm(a,b)):\\n{\\n}'.format(torch.mm(a,b)))
print('矩陣相乘(方法2: torch.matmul(a,b)):\\n{\\n}'.format(torch.matmul(a,b)))
print('矩陣相乘(方法3: a.matmul(b)):\\n{\\n}'.format(a.matmul(b)))
```





# 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)





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/lightning-gpt · 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/PyCodeGPT · 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-GNN · 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 heterogenous 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-2 models)
- **pip install datasets** for huggingface datasets <3 (if you want to download datasets)
- **pip install tiktoken** for OpenAI's fast BPE code <3
- **pip install wandb** for optional logging <3
- **pip install tqdm** <3

Ben new bnb version

Code Blame 16 lines (16 loc) · 301 Bytes

```
1 wandb==0.12.21
2 einops==0.4.1
3 tokenizers==0.12.1
4 hydra-core==1.2.0
5 tqdm==4.64.0
6 black==21.9b0
7 boto3==1.20.54
8 accelerate==0.10.0
9 transformers==4.20.1
10 webdataset==0.2.5
11 jsonlines==2.0.0
12 gradio==3.0.23
13 --extra-index-url https://download.pytorch.org/whl/cu113
14 torch==1.11.0+cu113
15 pandas==1.4.3
16 bitsandbytes
```

main lightning-GPT / requirements.txt

lantiga Fix packages ✓

Code Blame 3 lines (3 loc) · 37 Bytes

```
1 lightning>=1.8.0
2 torch==1.10.0
3 numpy
```

main PyCodeGPT / requirements.txt

substill Add files via upload

Code Blame 4 lines (4 loc) · 44 Bytes

```
1 torch
2 transformers
3 sentencepiece
```

master GPT-GNN / requirements.txt

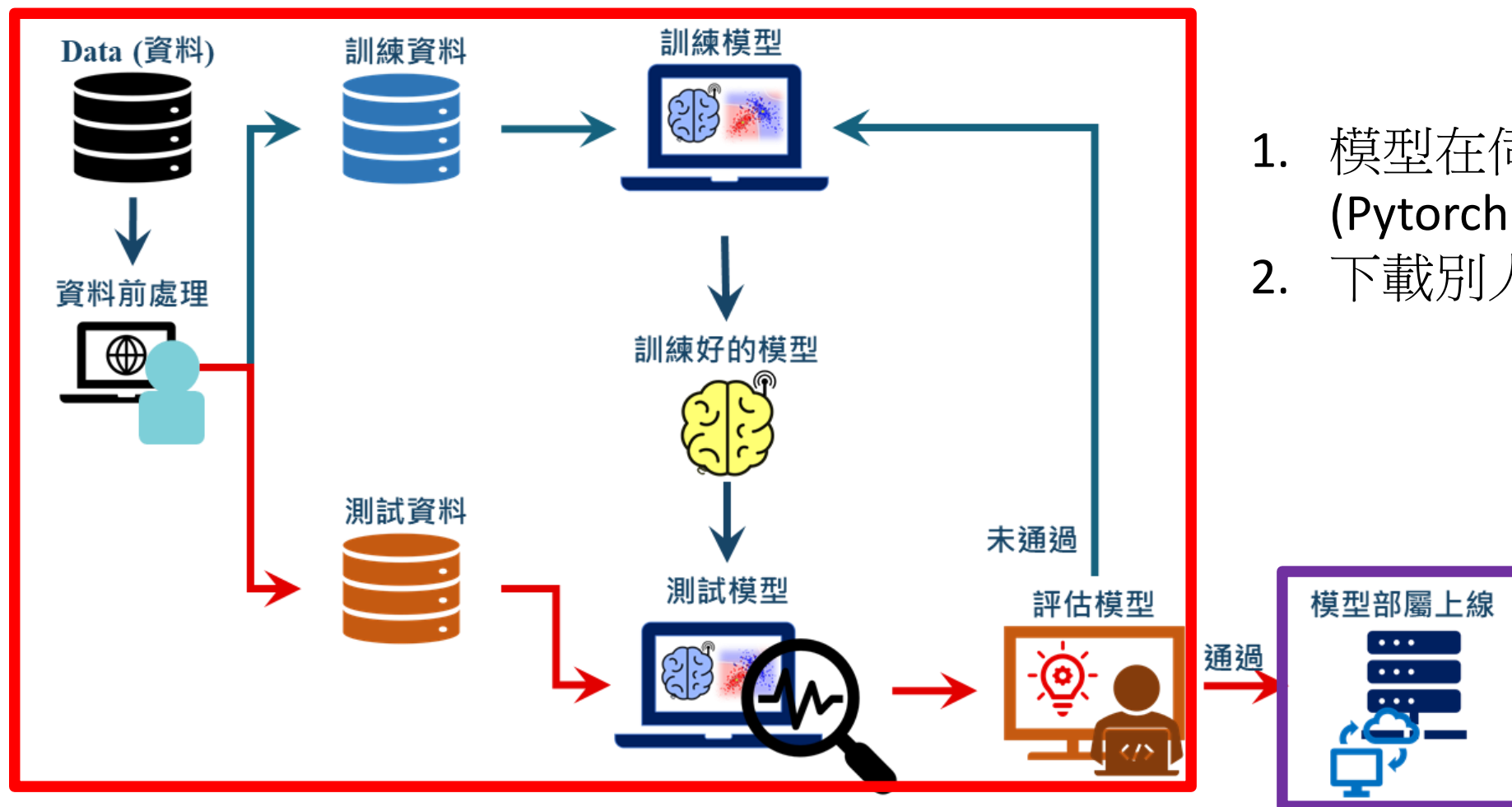
dependabot[bot] Bump numpy from 1.16.2 to 1.22.0

Code Blame 14 lines (14 loc) · 249 Bytes

```
1 dill==0.3.0
2 numpy==1.22.0
3 pandas==0.24.2
4 torch==1.3.0
5 torch-scatter==1.3.2
6 torch-cluster==1.4.5
7 torch-sparse==0.4.3
8 torch-spline-conv==1.1.1
9 torch-geometric==1.3.2
10 torchvision==0.4.1
11 tqdm==4.31.1
12 seaborn==0.9.0
13 matplotlib==3.0.3
14 transformers==2.8.0
```



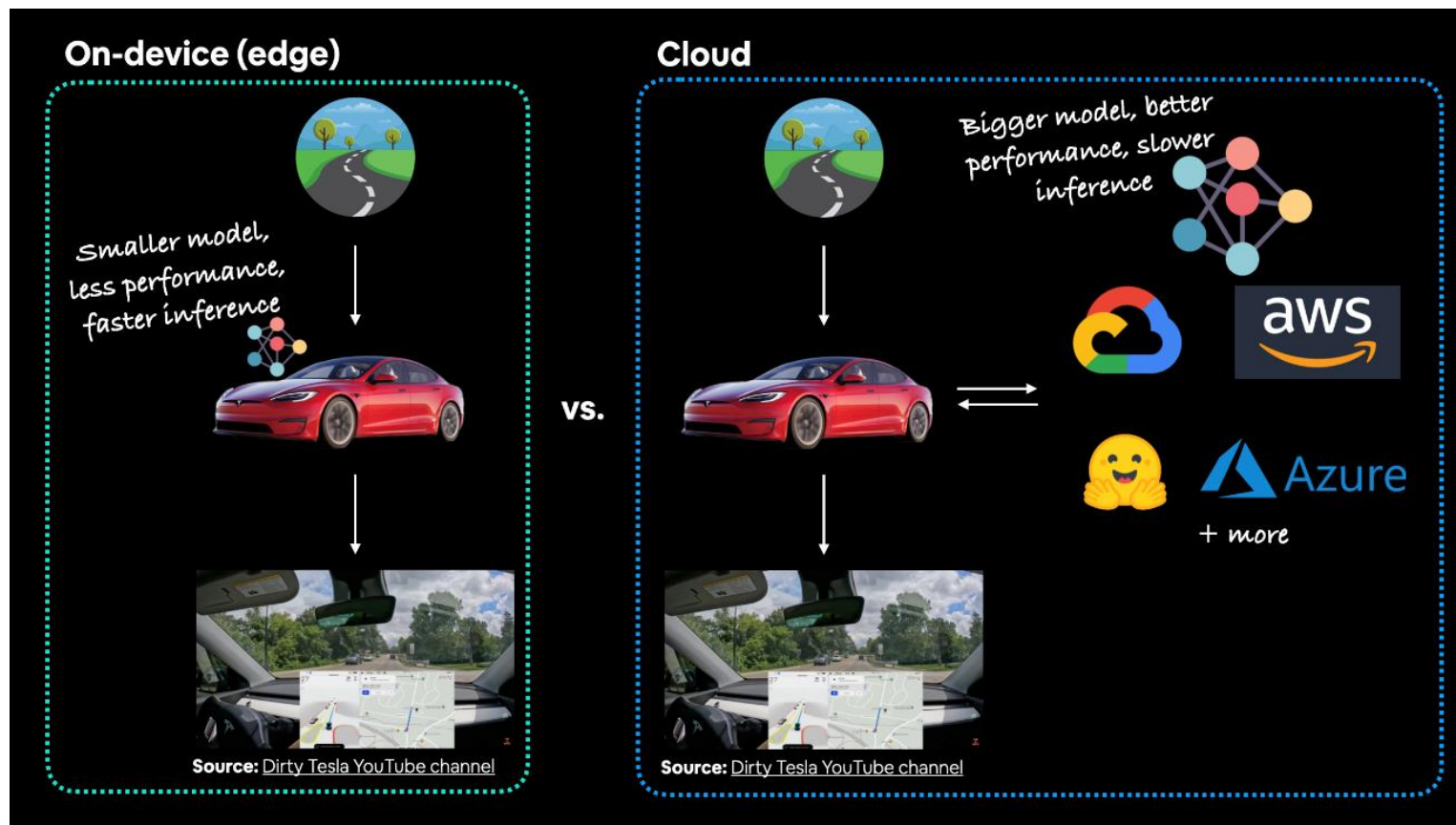
# PyTorch 如何協助深度學習專案的開發



1. 模型在伺服器上訓練 (Pytorch)
2. 下載別人訓練好的模型。



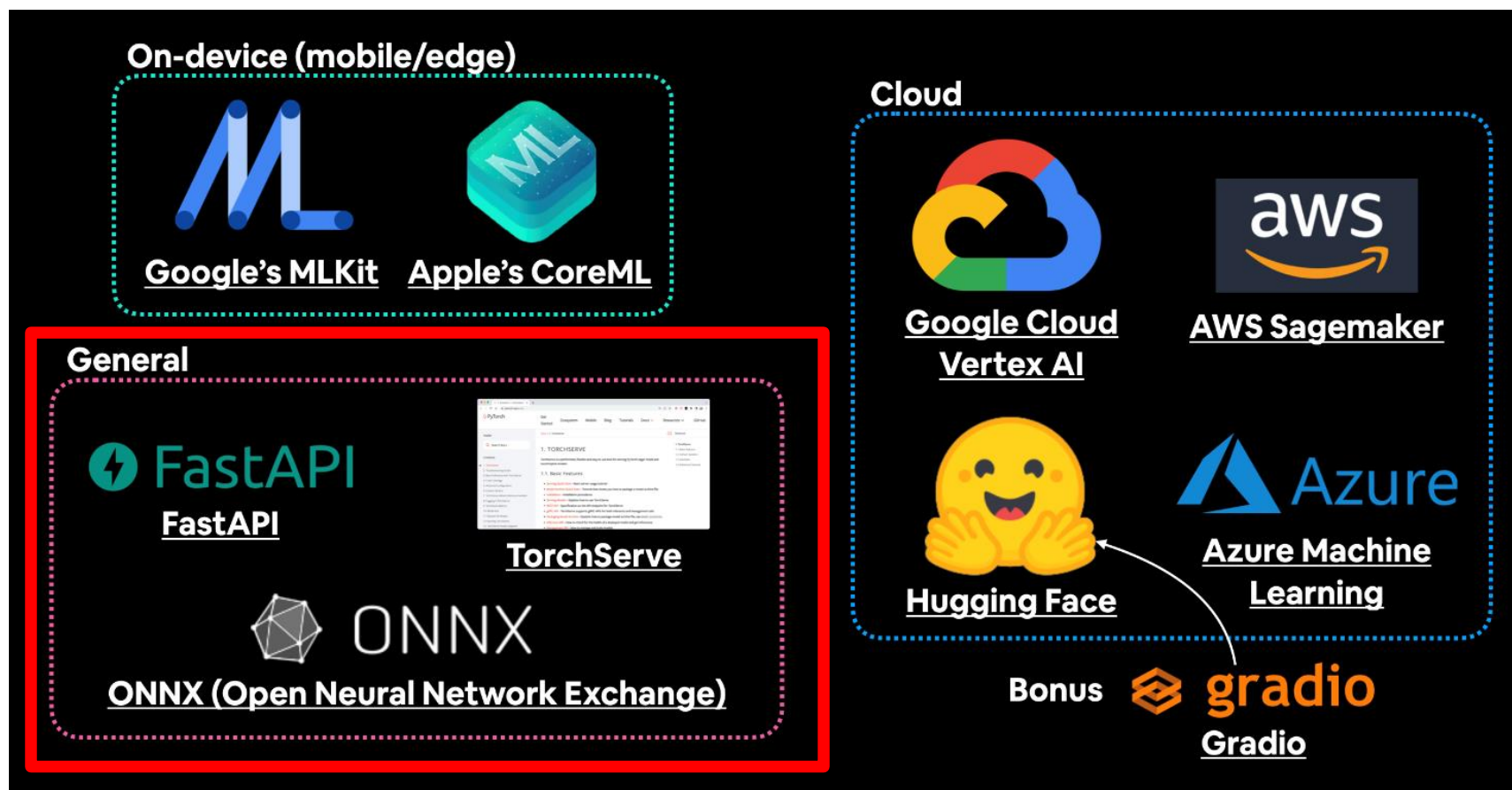
# PyTorch 如何協助深度學習專案的開發



[https://www.learnpytorch.io/09\\_pytorch\\_model\\_deployment/](https://www.learnpytorch.io/09_pytorch_model_deployment/)



# PyTorch 如何協助深度學習專案的開發



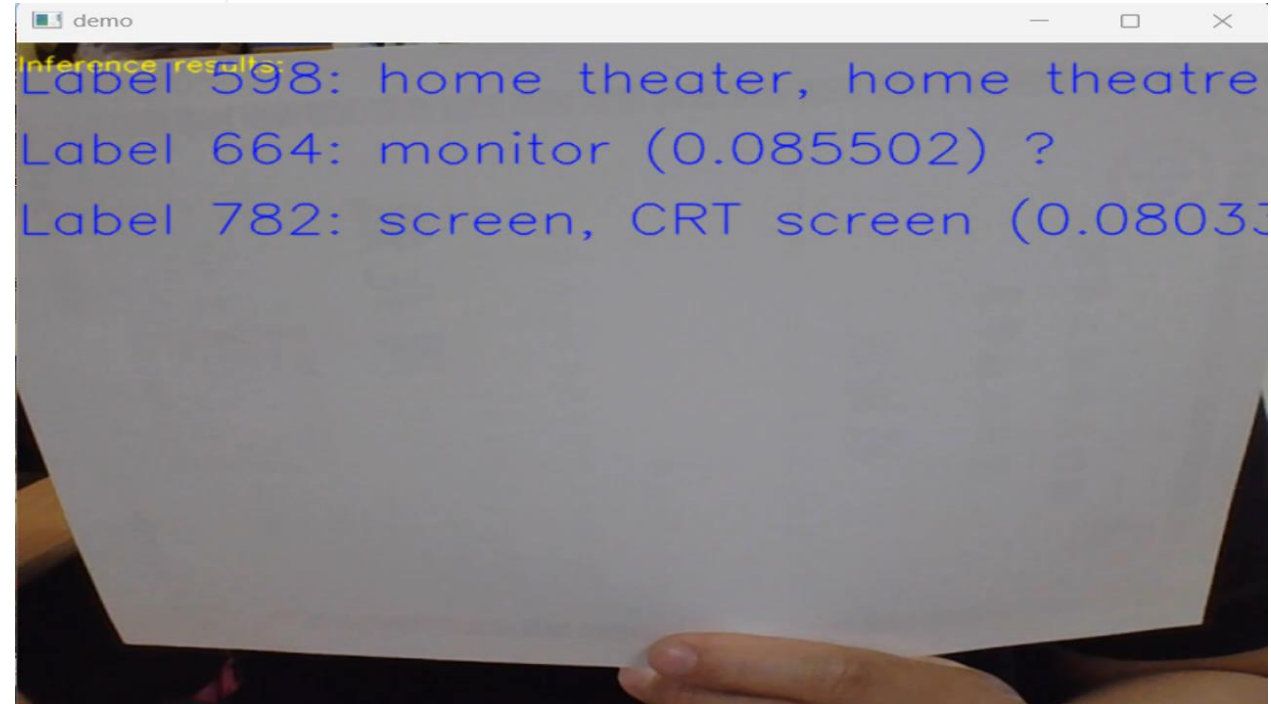


```

1 import cv2
2 import numpy as np
3 from PIL import Image
4 import onnxruntime as ort
5 import torchvision.transforms as trns
6 onnxmodel_path='./weight/mobilenetv2.onnx'
7 class_def = './weight/imagenet_classes.txt'
8 def softmax(x):
9     x = x.reshape(-1)
10    e_x = np.exp(x - np.max(x))
11    return e_x / e_x.sum(axis=0)
12 def postprocess(result):
13    return softmax(np.array(result)).tolist()
14 def main():
15     # Run the model on the backend
16    session = ort.InferenceSession(onnxmodel_path, None)
17    # get the name of the first input of the model
18    input_name = session.get_inputs()[0].name
19    # Load ImageNet classes
20    with open(class_def) as f:
21        classes = [line.strip() for line in f.readlines()]
22    # Define image transforms
23    transforms = trns.Compose([trns.Resize((224, 224)),
24                               trns.ToTensor(),
25                               trns.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
26    cap = cv2.VideoCapture(0)
27    while True:
28        ret, img = cap.read()
29        if not ret:
30            break
31        # Read image and run prepro
32        image = Image.fromarray(img).convert("RGB")
33        image_tensor = transforms(image)
34        image_tensor = image_tensor.unsqueeze(0)
35        image_np = image_tensor.numpy()
36        # model run
37        outputs = session.run([], {input_name: image_np})[0]
38        print("Output size:{}".format(outputs.shape))
39        # Result postprocessing
40        idx = np.argmax(outputs)
41        sort_idx = np.flip(np.argsort(outputs))
42        idx = np.argmax(outputs)
43        # outputs = np.sort(outputs[0,:])
44        probs = postprocess(outputs)
45        top_k=3
46        cv2.putText(img, "Inference results:", (0, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1, cv2.LINE_AA)
47        print("Inference results:")
48        for i, index in enumerate(sort_idx[:top_k]):
49            py = 35 + 50*i
50            text = "Label {}: {} ({:5f}) \n".format(index, classes[index], probs[index])
51            cv2.putText(img, text, (0, py), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)
52            print(text)
53        cv2.imshow('demo', img)
54        cv2.waitKey(1)
55    cap.release()
56 if __name__ == '__main__':
57     main()

```

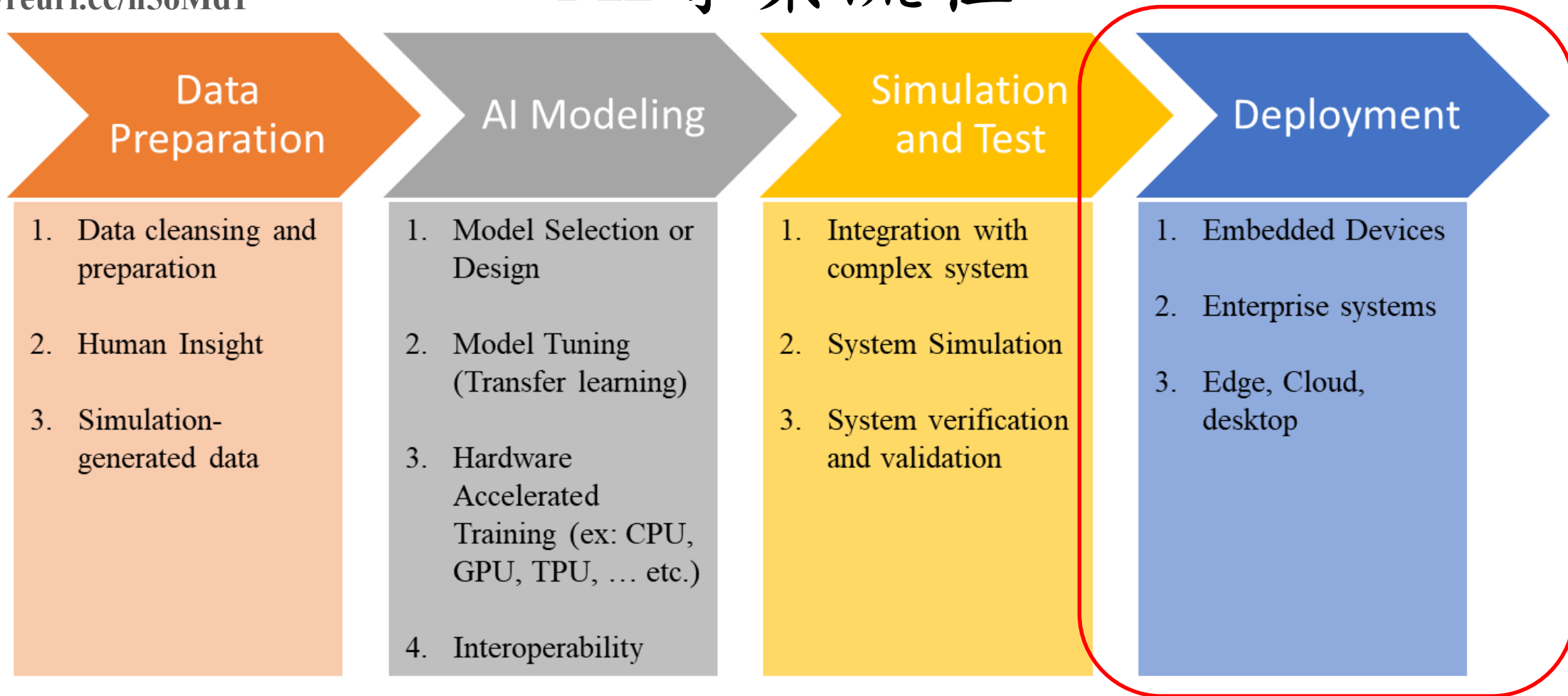
## Onnxruntime部屬訓練好的模型



Inference :  
TensorRT (Nvidia專屬)



# AI專案流程

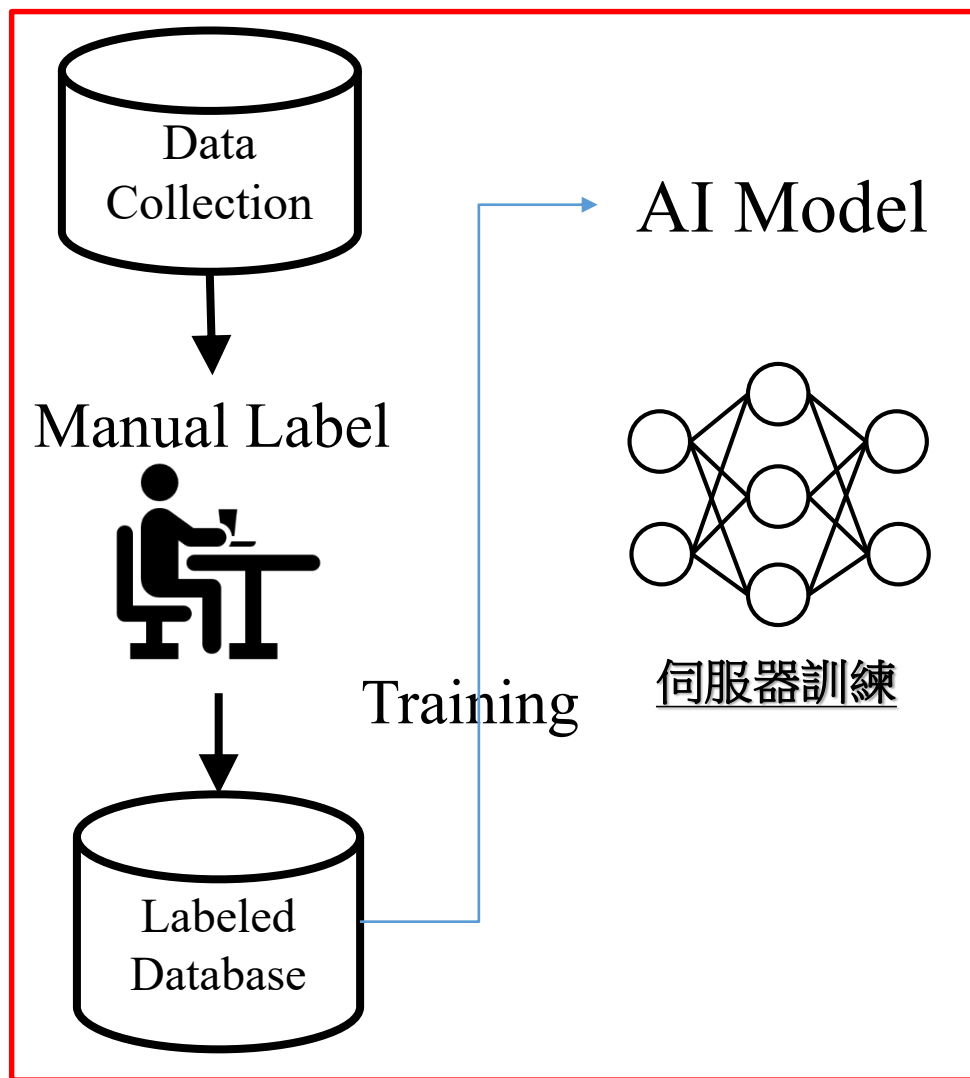


Modeling is an important step in the workflow, but the model is not the end of the journey





# 模型訓練和部屬



部屬  
(Deployment)

手機



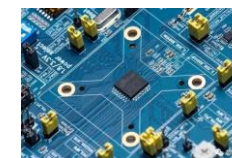
CPU



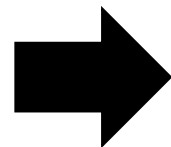
GPU



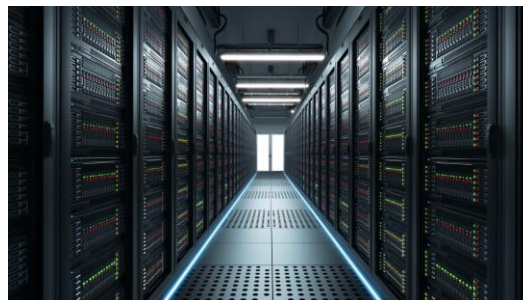
ASIC



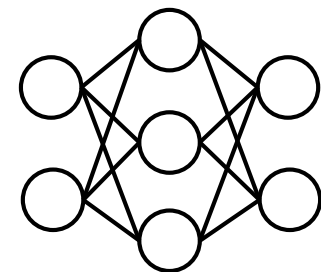
# 模型訓練框架



伺服器訓練



AI Model



# 模型訓練和部屬

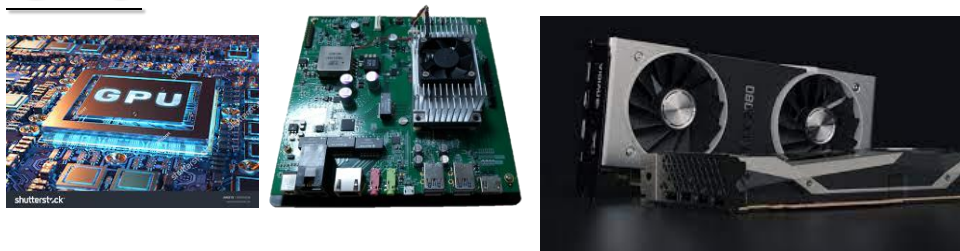
## 手機



## CPU



## GPU



## ASIC



1. 模型怎麼在不同的運算機器上跑推論

OS: Windows, Linus, RTOS

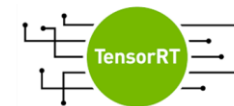
AI engine: Pytorch, Tensforflow, Caffe, Mxnet,...

Inference NN accelerator: CMSIS-NN, TensorFlow Lite, OpenVINO, TensorRT, Onnxruntime,...

2. 不同的訓練架構要怎麼互通



3. 是否有相同的運算架構做這樣的事情



PyTorch

TensorFlow

mxnet

Chainer

K Keras

Caffe2

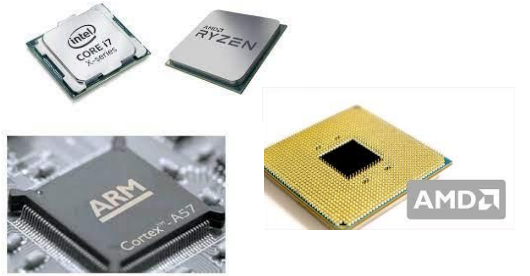


表示更容易部署的靜態圖

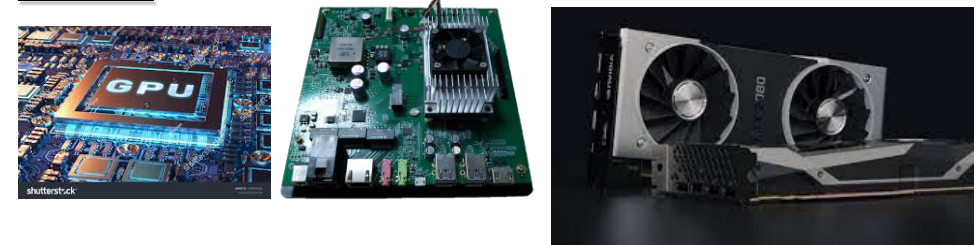
手機



CPU



GPU

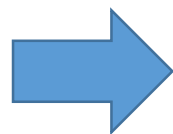


ASIC





ONNX



手機



CPU



GPU

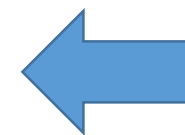
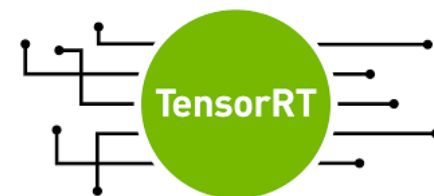


ASIC



FP-64/32/16  
INT-16/8/4/1

 ONNX  
RUNTIME



OpenVINO™

  
CMSIS-NN







# ONNX

## ONNX 簡介

- 神經網路框架格式間的互通性，讓不同標準的深度學習模型可以互相轉換，加快AI人工智慧的發展。
- ONNX: Open Neural Network Exchange: 開放神經網路交換格式。
- 由Facebook、微軟結盟推出
- 並獲得 Intel、AMD、ARM、IBM、華為和高通/聯發科晶片廠商支持加入。→大部分SoC都有Support。
- 目前官方支持加載ONNX模型並進行推理的深度學習框架有：Caffe2, PyTorch, MXNet, ML.NET, TensorRT 和 Microsoft CNTK，並且 TensorFlow 也非官方的支持ONNX。





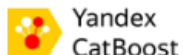
# ONNX

# ONNX 官網 (Frameworks)

## Frameworks & Converters

Use the frameworks you already know and love.

<https://onnx.ai/supported-tools.html>



CoreML



Optimum



Keras

LibSVM

MATLAB®

[M] MindSpore



NCNN



PaddlePaddle



SIEMENS



SciKit Learn



ZAMA







# ONNX 部屬



## Deploy Model

### Inference

Deploy your ONNX model using runtimes designed to accelerate inferencing.

BITMAIN

cādence®

CEVA®

Da takalab



deepC

groq™

habana

HAILO

MACE  
Mobile AI Compute Engine

NVIDIA

OpenVINO™



Optimum

ONNX  
MLIR

ONNX  
RUNTIME

Qualcomm

Rackchip

skymizer

SYNOPSYS®

Tencent

teradata.

Tensil

TensorFlow

tvm

TwinCAT® 3

vespa

Windows



# ONNX其他工具



## Additional Tools

### Optimize

Fine tune your model for size, accuracy, resource utilization, and performance.



ONNX

### Visualize

Better understand your model by visualizing its computational graph.

NETRON

Visual DL



LUTZ ROEDER'S

NETRON

Open Model...

<https://netron.app/>



# Netron 模型視覺化

+

🔍 🔄 🗑️ ⬆️ 排序 ▾ ☰ 檢視 ▾ ⋮

版書 > MasterTalks > Part2 > code ▾

名稱	修改日期	類型	大小
main_pytorch_ImageClassification_onn...	2021/12/24 下午 01:37	PY 檔案	3 KB
mobilenet_v2.onnx	2023/6/2 上午 08:45	ONNX 檔案	13,665 KB
mobilenet_v2.pth	2023/6/2 上午 08:45	PTH 檔案	13,962 KB
Untitled.ipynb	2023/6/2 上午 10:20	IPYNB 檔案	3 KB
記憶體使用.ipynb	2023/5/24 下午 01:45	IPYNB 檔案	10 KB

Open Model...

Tommy Huang a .png ⬆️ | Untitled.pdf ⬆️

🔍 查詢 顯示 ▾

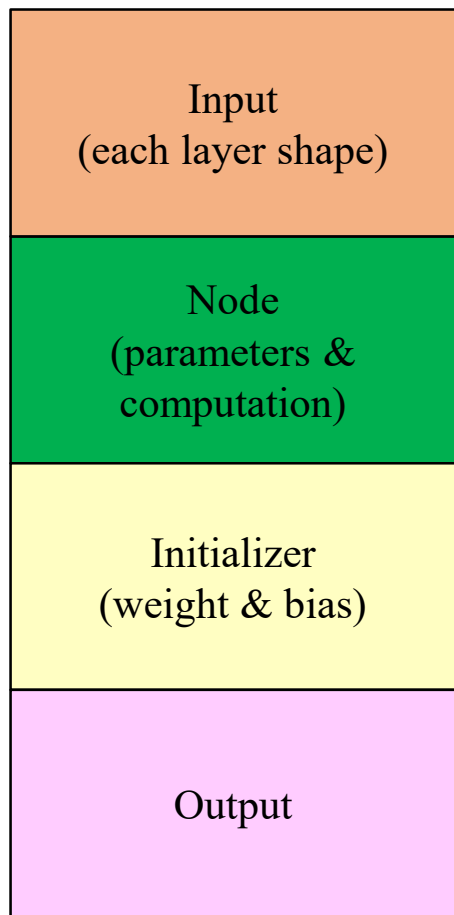


# ONNX Structure

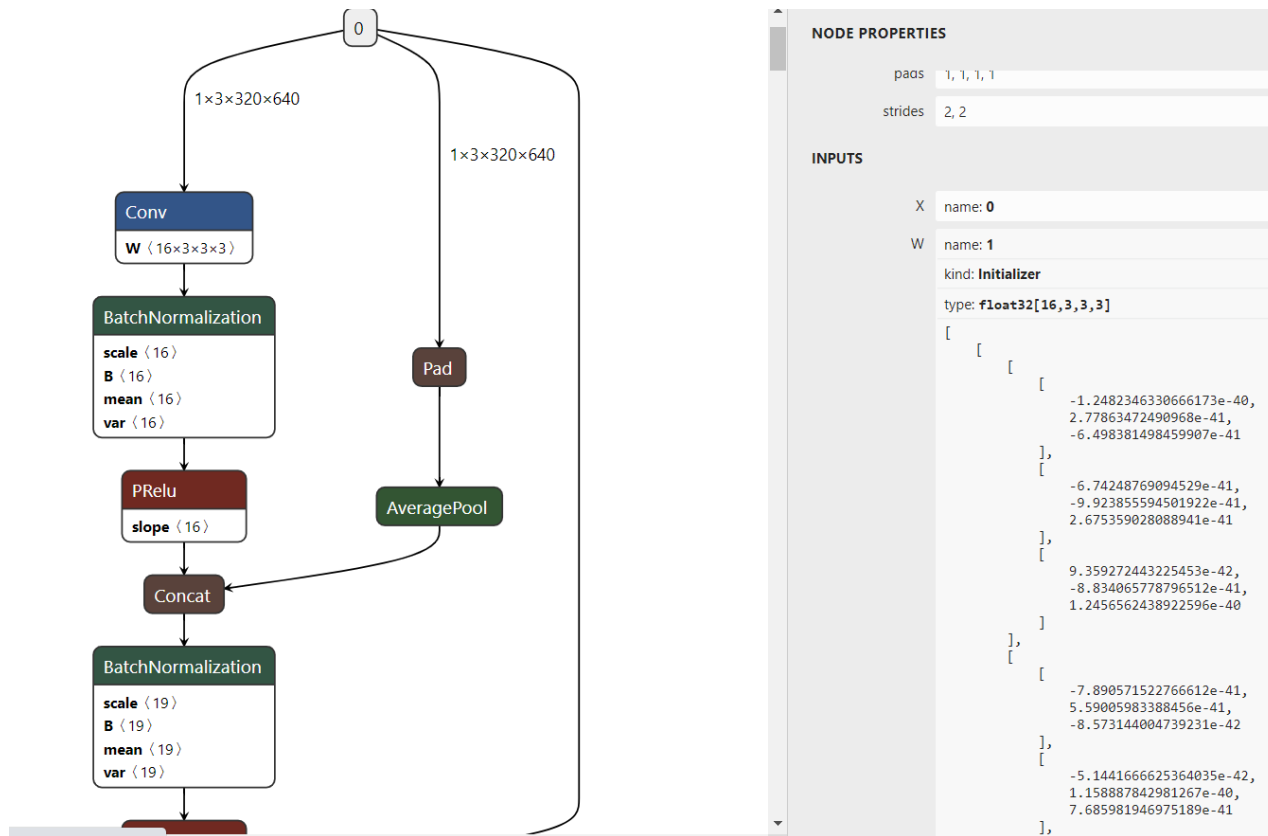
ONNX Model

ModelProto

GraphProto



Netron



<https://netron.app/>



# ONNXRuntime

- **ONNX Runtime**是由微軟維護的一個跨平台機器學習推理加速器→「推理引擎」。
  - ONNX Runtime 是直接對接ONNX 的，即ONNX Runtime 可以直接讀取並運行.onnx 檔案, 而不需要再把.onnx 格式的檔案轉換成其他格式的檔案。
- I. PyTorch → ONNX → ONNX Runtime
  - II. ONNX → ONNX Runtime
- ONNX Runtime 提供了Python 介面。

```
import onnxruntime
```

```
ort_session = onnxruntime.InferenceSession( "model.onnx")
```

```
ort_inputs = {'input': input_img}
```

```
ort_output = ort_session.run(['output'], ort_inputs)[0]
```



```

14 def main():
15     # Run the model on the backend
16     session = ort.InferenceSession(onnxmodel_path, None)
17     # get the name of the first input of the model
18     input_name = session.get_inputs()[0].name
19     # Load ImageNet classes
20     with open(class_def) as f:
21         classes = [line.strip() for line in f.readlines()]
22     # Define image transforms
23     transforms = trns.Compose([trns.Resize((224, 224)),
24                               trns.ToTensor(),
25                               trns.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])])
26     cap = cv2.VideoCapture(0)
27     while True:
28         ret, img = cap.read()
29         if not ret:
30             break
31         # Read image and run prepro
32         image = Image.fromarray(img).convert("RGB")
33         image_tensor = transforms(image)
34         image_tensor = image_tensor.unsqueeze(0)
35         image_np = image_tensor.numpy()
36         # model run
37         outputs = session.run([], {input_name: image_np})[0]
38         print("Output size:{}".format(outputs.shape))
39         # Result postprocessing
40         idx = np.argmax(outputs)
41         sort_idx = np.flip(np.argsort(outputs))
42         idx = np.argmax(outputs)
43         # outputs = np.sort(outputs[0,:])
44         probs = postprocess(outputs)
45         top_k=3
46         cv2.putText(img, "Inference results:", (0, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255),
47                     print("Inference results:")
48         for i, index in enumerate(sort_idx[:top_k]):
49             py = 35 + 50*i
50             text = "Label {}: {} ({}:5f) \n".format(index, classes[index], probs[index])
51             cv2.putText(img, text, (0, py), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)
52             print(text)
53             cv2.imshow('demo', img)
54             cv2.waitKey(1)
55         cap.release()
56 if __name__ == '__main__':
57     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
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))

```





# 模型部署中的問題

## PyTorch → ONNX

```
x = torch.randn(1, 3, 256, 256)
```

```
with torch.no_grad():  
    torch.onnx.export(  
        model,  
        x,  
        "model.onnx",  
        opset_version=11,  
        input_names=['input'],  
        output_names=['output'])
```

ONNX是靜態圖。

Onnx生成的方式是餵入一個3 channels大小是256\*256的資料，生成出來的onnx就不能吃其他size。





# 直接用 ONNX 建模

- 可以，但不建議。

試著完全用ONNX 的Python API 建構一個描述線性函數 $\text{output} = a * x + b$  的ONNX 模型

- ModelProto

- GraphProto

- NodeProto

- ValueInfoProto

1. 宣告 output, a, x, b 的值

2. 宣告計算的Node

- I. 乘法( $c = a * x$ )、II 加法( $\text{output} = c + b$ )

3. 建立graph圖

4. 建立model



# Q&A

