Network In Network

Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." *arXiv:1312.4400* (2013)

李長青

Outline

- 動機
- 資料前處理(Self-made Dataset)
- Network in Network架構調整
- NiN_v2 v.s. Inception
- K-fold交叉驗證 + SubsetRandomSampler
- 歸納整理

動機

近幾天以來台灣的疫情升溫相信大家有目共睹，民眾的恐慌也在心中蔓延，身為台灣的一份子，我們希望為了社會做出些微的貢獻，所以我們選擇將Network in Network用在口罩辨識的領域中，將其用來守護台灣防疫陣線，遏止疫情進一步擴散，最少做到督促自身，提醒自己出門時刻要戴好口罩。

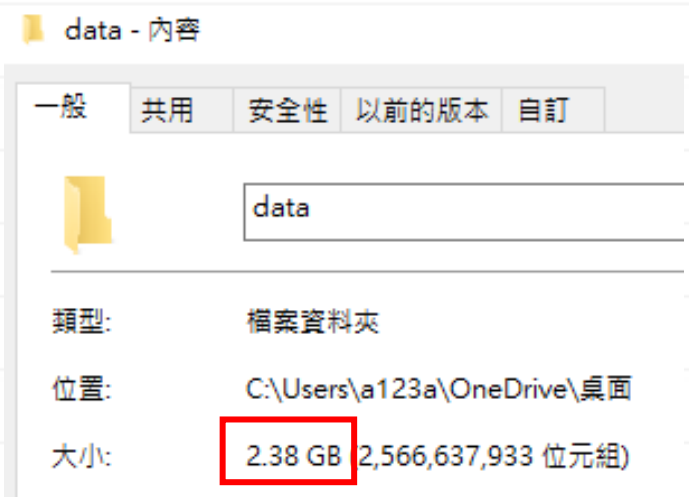
Self-made Dataset-01蒐集圖片

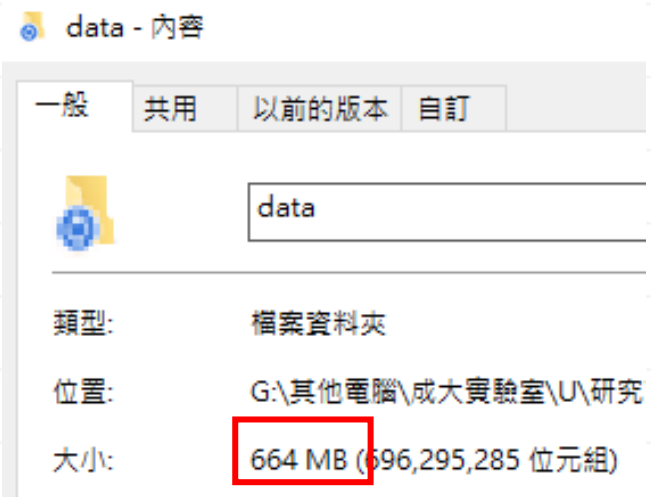
- 花了許多時間蒐集布口罩的Dataset
- 眼睛辨識自製Dataset:
  - Face_with_mask (3680張)
    - cloth_mask (950張)
    - surgical_mask (2730張)
  - Face_no_mask (1352張)

Self-made Dataset-02批量自適應圖片解析度

- 原始圖 -> 解析度大小不一
  - 3024x4032
  - 614x410
  - …

- 處理後 -> 統一調成 500x (H or W)
  - 500x281
  - 333x500
  - …





訓練時間: 1分鐘25秒/epoch

訓練時間: 36秒/epoch

(每回合 節省2倍 訓練時間)

Self-made Dataset-03批量隨機分割train/val

Run Markdown

準備:

- 資料夾結構：

```
|-- ori_photo
    |-- class_0
    |-- class_1
    |-- class_2
    |-- class_3
|-- data_to_train+val.ipynb
```

結果:

- 生成的資料夾:

```
|-- data
    |-- train
        |-- class_0
        |-- class_1
        |-- class_2
        |-- class_3
    |-- val
        |-- class_0
        |-- class_1
        |-- class_2
        |-- class_3
```

```python
def split_data(input_file_path, output_file_path, split_rate, seed='random'):
    if seed == 'fixed':
        random.seed(0)
    else:
        random.seed()
    # 獲取當前文件路徑
    cwd = os.getcwd()
    input_dataset_path = os.path.join(cwd, input_file_path)
    output_dataset_path = os.path.join(cwd, output_file_path)
    assert os.path.exists(input_dataset_path), f"path '{input_dataset_path}' does not exist."

    dataset_classes = [dataset_class for dataset_class in os.listdir(input_dataset_path) if
                       os.path.isdir(os.path.join(input_dataset_path, dataset_class))]
    # 訓練集
    train_path = os.path.join(output_dataset_path, 'train')
    make_dir(train_path)
    for dataset_class in dataset_classes:
        make_dir(os.path.join(train_path, dataset_class))
    # 驗證集
    val_path = os.path.join(output_dataset_path, 'val')
    make_dir(val_path)
    for dataset_class in dataset_classes:
        make_dir(os.path.join(val_path, dataset_class))

    for dataset_class in dataset_classes:
        input_dataset_class_path = os.path.join(input_dataset_path, dataset_class)
        images = os.listdir(input_dataset_class_path)
        images_num = len(images)
        # 隨機採取驗證集
        val_images = random.sample(images, k=int(images_num * split_rate))
        for index, image in enumerate(images):
            # 每張圖像路徑
            image_path = os.path.join(input_dataset_class_path, image)
            if image in val_images:
                # 將圖像文件copy到驗證圖像路徑
                copy(image_path, os.path.join(val_path, dataset_class))
            else:
                copy(image_path, os.path.join(train_path, dataset_class))
            #print(f"[{dataset_class}] is processing: {index + 1}/{images_num}")  # 處理分割狀況用
    print('Process Finished.')


if __name__ == '__main__':
    original_data_file_path = 'ori_photo'
    spilit_data_file_path = 'data'
    split_rate = 0.2
    split_data(original_data_file_path, spilit_data_file_path, split_rate)
```

```
Process Finished.
```

Transform:

\+ 數據正規化
隨機水平翻轉
隨機Resize

(測試下來只使用這三個)

Network in Network架構調整

- NiN_v1

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
NIN_v1                                   [32, 3]                   --
├─Sequential: 1-1                        [32, 3]                   --
│    └─Conv2d: 2-1                       [32, 192, 224, 224]       14,592
│    └─ReLU: 2-2                         [32, 192, 224, 224]       --
│    └─Conv2d: 2-3                       [32, 160, 224, 224]       30,880
│    └─ReLU: 2-4                         [32, 160, 224, 224]       --
│    └─Conv2d: 2-5                       [32, 96, 224, 224]        15,456
│    └─ReLU: 2-6                         [32, 96, 224, 224]        --
│    └─MaxPool2d: 2-7                    [32, 96, 112, 112]        --
│    └─Dropout: 2-8                      [32, 96, 112, 112]        --
│    └─Conv2d: 2-9                       [32, 192, 112, 112]       460,992
│    └─ReLU: 2-10                        [32, 192, 112, 112]       --
│    └─Conv2d: 2-11                      [32, 192, 112, 112]       37,056
│    └─ReLU: 2-12                        [32, 192, 112, 112]       --
│    └─Conv2d: 2-13                      [32, 192, 112, 112]       37,056
│    └─ReLU: 2-14                        [32, 192, 112, 112]       --
│    └─AvgPool2d: 2-15                   [32, 192, 56, 56]         --
│    └─Dropout: 2-16                     [32, 192, 56, 56]         --
│    └─Conv2d: 2-17                      [32, 192, 56, 56]         331,968
│    └─ReLU: 2-18                        [32, 192, 56, 56]         --
│    └─Conv2d: 2-19                      [32, 192, 56, 56]         37,056
│    └─ReLU: 2-20                        [32, 192, 56, 56]         --
│    └─Conv2d: 2-21                      [32, 3, 56, 56]           579
│    └─ReLU: 2-22                        [32, 3, 56, 56]           --
│    └─AdaptiveAvgPool2d: 2-23           [32, 3, 1, 1]             --
│    └─Flatten: 2-24                     [32, 3]                   --
==========================================================================================
```

- NiN_v2

```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
NIN_v2                                   [32, 3]                   --
├─Sequential: 1-1                        [32, 3]                   --
│    └─Sequential: 2-1                   [32, 96, 55, 55]          --
│    │    └─Conv2d: 3-1                  [32, 96, 55, 55]          34,944
│    │    └─ReLU: 3-2                    [32, 96, 55, 55]          --
│    │    └─Conv2d: 3-3                  [32, 96, 55, 55]          9,312
│    │    └─ReLU: 3-4                    [32, 96, 55, 55]          --
│    │    └─Conv2d: 3-5                  [32, 96, 55, 55]          9,312
│    │    └─ReLU: 3-6                    [32, 96, 55, 55]          --
│    └─Dropout: 2-2                      [32, 96, 55, 55]          --
│    └─MaxPool2d: 2-3                    [32, 96, 27, 27]          --
│    └─Sequential: 2-4                   [32, 256, 27, 27]         --
│    │    └─Conv2d: 3-7                  [32, 256, 27, 27]         614,656
│    │    └─ReLU: 3-8                    [32, 256, 27, 27]         --
│    │    └─Conv2d: 3-9                  [32, 256, 27, 27]         65,792
│    │    └─ReLU: 3-10                   [32, 256, 27, 27]         --
│    │    └─Conv2d: 3-11                 [32, 256, 27, 27]         65,792
│    │    └─ReLU: 3-12                   [32, 256, 27, 27]         --
│    └─Dropout: 2-5                      [32, 256, 27, 27]         --
│    └─MaxPool2d: 2-6                    [32, 256, 13, 13]         --
│    └─Sequential: 2-7                   [32, 384, 13, 13]         --
│    │    └─Conv2d: 3-13                 [32, 384, 13, 13]         885,120
│    │    └─ReLU: 3-14                   [32, 384, 13, 13]         --
│    │    └─Conv2d: 3-15                 [32, 384, 13, 13]         147,840
│    │    └─ReLU: 3-16                   [32, 384, 13, 13]         --
│    │    └─Conv2d: 3-17                 [32, 384, 13, 13]         147,840
│    │    └─ReLU: 3-18                   [32, 384, 13, 13]         --
│    └─Dropout: 2-8                      [32, 384, 13, 13]         --
│    └─AvgPool2d: 2-9                    [32, 384, 7, 7]           --
│    └─Sequential: 2-10                  [32, 3, 7, 7]             --
│    │    └─Conv2d: 3-19                 [32, 3, 7, 7]             10,371
│    │    └─ReLU: 3-20                   [32, 3, 7, 7]             --
│    │    └─Conv2d: 3-21                 [32, 3, 7, 7]             12
│    │    └─ReLU: 3-22                   [32, 3, 7, 7]             --
│    │    └─Conv2d: 3-23                 [32, 3, 7, 7]             12
│    │    └─ReLU: 3-24                   [32, 3, 7, 7]             --
│    └─AdaptiveAvgPool2d: 2-11           [32, 3, 1, 1]             --
│    └─Flatten: 2-12                     [32, 3]                   --
==========================================================================================
```

Network in Network架構調整 藍色=修改後

- NiN_v1

```
==========
Total params: 965,635
Trainable params: 965,635
Non-trainable params: 0
Total mult-adds (G): 349.71
==========
Input size (MB): 19.27
Forward/backward pass size (MB): 7914.96
Params size (MB): 3.86
Estimated Total Size (MB): 7938.09
==========
```

x3 →

÷3 →

÷18 →

- NiN_v2

```
==========
Total params: 1,991,003
Trainable params: 1,991,003
Non-trainable params: 0
Total mult-adds (G): 29.00
==========
Input size (MB): 19.27
Forward/backward pass size (MB): 416.31
Params size (MB): 7.96
Estimated Total Size (MB): 443.54
==========
```

Spend time: 1分鐘41秒/epoch

Average Accuracy: 53.92%

Spend time: 36秒/epoch

Average Accuracy: 56.36%

- Spend time 控制變因:
batch size=16
Input_size=224
torch.manual_seed(0)

- Avg Accuracy 控制變因:
lr=0.001 (根據val_acc*0.1)
loss_func = CrossEntropy
optim = SGD
Train_data = 4026 (K-fold)
Val_data = 1006 (K-fold)

NiN_v2 v.s. Inception

藍色=修改後

- NiN_v2

```
==========================================
Total params: 1,991,003
Trainable params: 1,991,003
Non-trainable params: 0
Total mult-adds (G): 29.00
==========================================
Input size (MB): 19.27
Forward/backward pass size (MB): 416.31
Params size (MB): 7.96
Estimated Total Size (MB): 443.54
==========================================
```

x12

x1.5

x3

- Inception

```
==========================================
Total params: 24,351,718
Trainable params: 24,351,718
Non-trainable params: 0
Total mult-adds (G): 45.37
==========================================
Input size (MB): 9.63
Forward/backward pass size (MB): 1189.97
Params size (MB): 97.41
Estimated Total Size (MB): 1297.02
==========================================
```

Spend time: 36秒/epoch

Average Accuracy: 56.36%

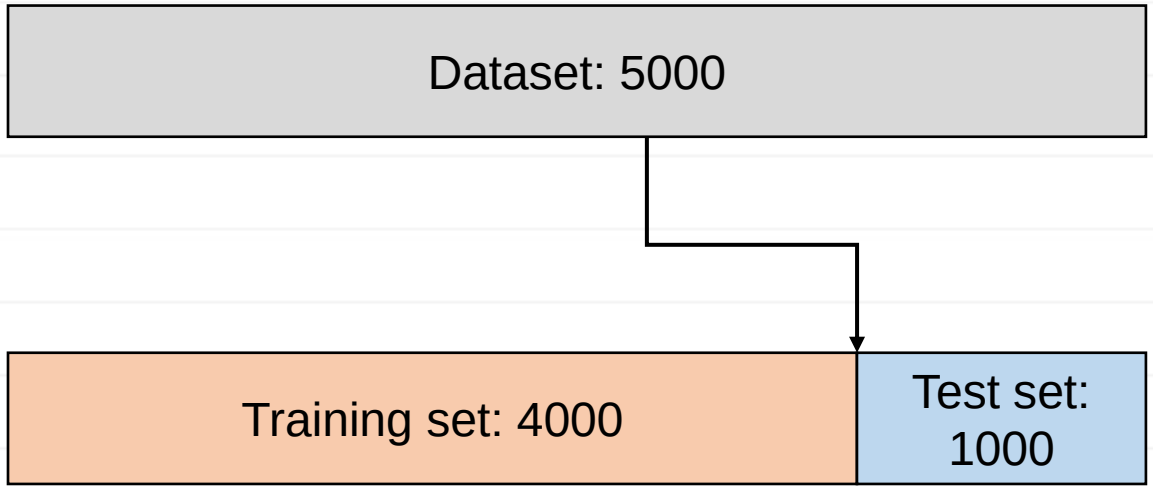
Spend time: 1分鐘3秒/epoch

Average Accuracy: 68.22%

- Spend time 控制變因:
batch size=16
Input_size=224
torch.manual_seed(0)

- Avg Accuracy 控制變因:
lr=0.001 (根據val_acc*0.1)
loss_func = CrossEntropy
optim = SGD
Train_data = 4026 (K-fold)
Val_data = 1006 (K-fold)

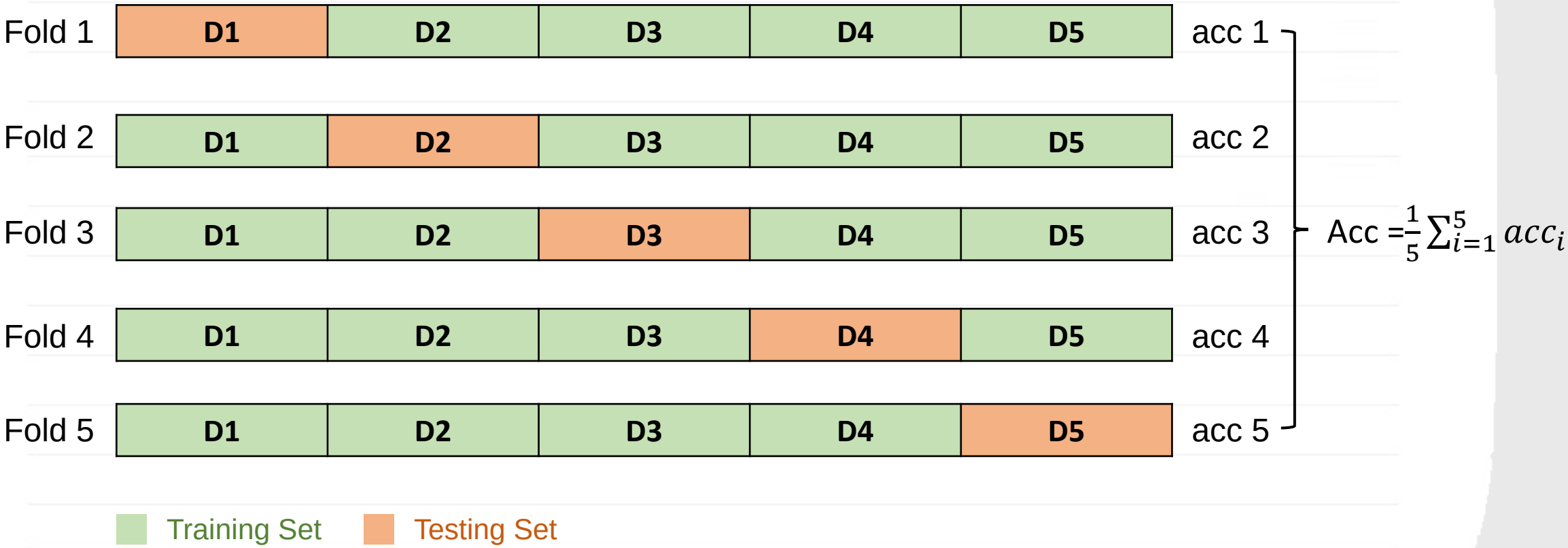
Simple hold-out split



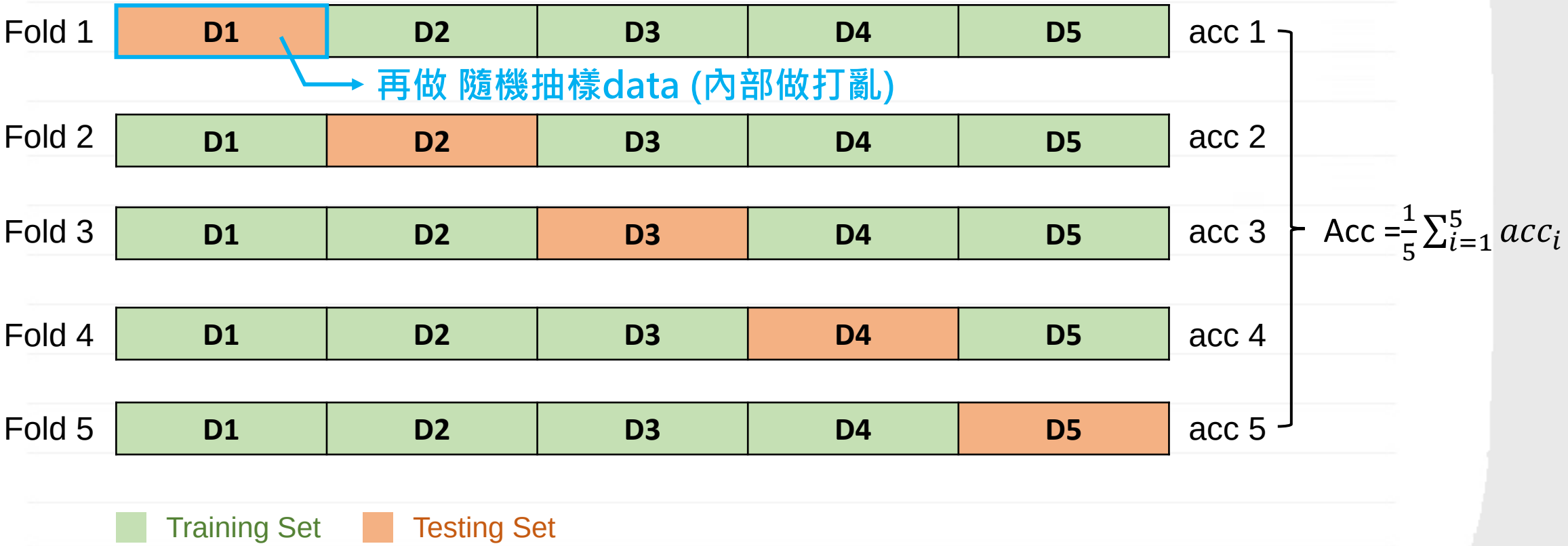
K-fold交叉驗證

- 僅切一小份資料是否就能有效的評估訓練時模型的好壞?
- 模型對我們所切的驗證集是否會過度擬和?
- 目的: 得到可靠穩定的模型
- 方法: 在每次的迭代中會選擇一組作為驗證集，其餘 (k-1) 組作為訓練集。

K-fold交叉驗證 (以5-fold為例)



K-fold交叉驗證 + SubsetRandomSampler

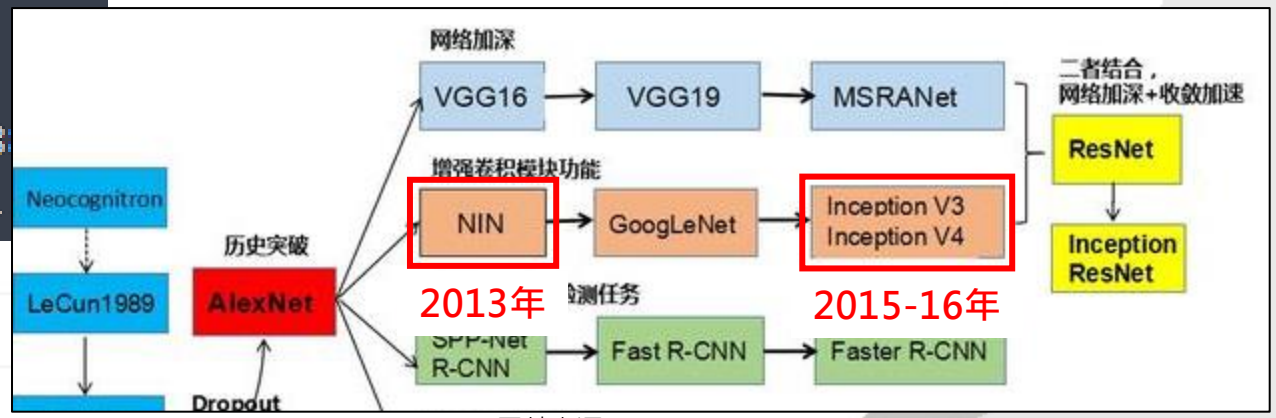


K-fold交叉驗證 + SubsetRandomSampler

- NIN_v2結果

```
--------------------------------
5次 K-fold 交叉驗證結果:
K_fold 0 -> 52.234 %
K_fold 1 -> 18.272 %
K_fold 2 -> 17.793 %
K_fold 3 -> 53.280 %
K_fold 4 -> 56.362 %
K_fold Average Accuracy: 39.58
--------------------------------
```

- 可以看出，在辨識不同材質口罩上，NIN模型的適應能力不是很好



圖片來源: https://kknews.cc/code/xg5jya8.html

K-fold交叉驗證 + SubsetRandomSampler

NiN v.s. Inception (Backbone)

- NIN_v2結果:

```
--------------------------------
5次 K-fold 交叉驗證結果:
K_fold 0 -> 52.234 %
K_fold 1 -> 18.272 %
K_fold 2 -> 17.793 %
K_fold 3 -> 53.280 %
K_fold 4 -> 56.362 %
K_fold Average Accuracy: 39.588 %
--------------------------------
```

- Inception結果:

```
--------------------------------
5次 K-fold 交叉驗證結果:
K_fold 0 -> 68.222 %
K_fold 1 -> 68.421 %
K_fold 2 -> 68.191 %
K_fold 3 -> 69.085 %
K_fold 4 -> 67.396 %
K_fold Average Accuracy: 68.263 %
--------------------------------
```

歸納整理

	Spend time	圖片解析度	用途
原始Dataset	1分鐘25秒/epoch	不固定	可節省訓練時間
處理後Dataset	36秒/epoch	500x(W or H)	

	Spend time	Parameters	Average Accuracy	K-fold + Rand_Sampler 考驗模型適應能力
NIN	1分鐘41秒/epoch	965k mult-add 349G	53.92%	26.01%
NIN_修改後	36秒/epoch	1,991k mult-add 29G	56.36%	39.58%
Inception	1分鐘3秒/epoch	24,351k Mult-add 45G	68.22%	68.26%