B093040024
郭晏涵

深度視覺

HW9: Facial Keypoint Detection

notebook 執行過程：

Load Data：下載資料集

Load and Visualize Data

To load the data, we have already prepared a Pytorch Dataset class `FacialKeypointsDataset` for you. You can find it in `exercise_code/data/facial_keypoints_dataset.py`. Run the following cell to download the data and initialize your dataset:

```
[6] download_url = 'https://vision.in.tum.de/webshare/g/i2dl/facial_keypoints.zip'
    i2dl_exercises_path = os.path.dirname(os.path.abspath(os.getcwd()))
    data_root = os.path.join(i2dl_exercises_path, "datasets", "facial_keypoints")
    train_dataset = FacialKeypointsDataset(
            train=True,
            transform=transforms.ToTensor(),
            root=data_root,
            download_url=download_url
    )
    val_dataset = FacialKeypointsDataset(
            train=False,
            transform=transforms.ToTensor(),
            root=data_root,
    )
    print("Number of training samples:", len(train_dataset))
    print("Number of validation samples:", len(val_dataset))
```

```
Number of training samples: 1546
Number of validation samples: 298
```

Visualize Data：查看資料狀況

```
[7] image, keypoints = train_dataset[0]["image"], train_dataset[0]["keypoints"]
    print("Shape of the image:", image.size())
    print("Smallest value in the image:", torch.min(image))
    print("Largest value in the image:", torch.max(image))
    print(image)
```

```
Shape of the image: torch.Size([1, 96, 96])
Smallest value in the image: tensor(0.0118)
Largest value in the image: tensor(1.)
tensor([[[0.3804, 0.2039, 0.2275,  ..., 0.9922, 0.9961, 0.9961],
         [0.3333, 0.2157, 0.2588,  ..., 0.9922, 0.9961, 0.9961],
         [0.2941, 0.2588, 0.2902,  ..., 0.9922, 0.9961, 0.9961],
         ...,
         [0.1294, 0.1255, 0.1255,  ..., 0.9255, 1.0000, 1.0000],
         [0.1294, 0.1255, 0.1216,  ..., 0.9490, 0.9843, 0.9804],
         [0.1216, 0.1176, 0.1216,  ..., 0.9255, 1.0000, 0.9922]]])
```

```
[8] keypoints = train_dataset[0]["keypoints"]
    print(keypoints)
```

```
tensor([[ 0.4685, -0.2319],
        [-0.4253, -0.1953],
        [ 0.2908, -0.2214],
        [ 0.5992, -0.2214],
        [-0.2685, -0.2109],
        [-0.5873, -0.1900],
        [ 0.1967, -0.3827],
        [ 0.7656, -0.4295],
        [-0.2035, -0.3758],
        [-0.7389, -0.3573],
        [ 0.0086,  0.2333],
        [ 0.4163,  0.6620],
        [-0.3521,  0.6985],
        [ 0.0138,  0.6045],
        [ 0.0190,  0.9076]])
```

Design a Convolution Neural Network Model

```
class KeypointModel(pl.LightningModule):
    """Facial keypoint detection model"""
    def __init__(self, hparams):
        """
        Initialize your model from a given dict containing all your hparams
        Warning: Don't change the method declaration (i.e. by adding more
            arguments), otherwise it might not work on the submission server
        """
        super(KeypointModel, self).__init__()
        self.hp = hparams
        ############################################################################
        # TODO: Define all the layers of your CNN, the only requirements are:
        # 1. The network takes in a batch of images of shape (Nx1x96x96)
        # 2. It ends with a linear layer that represents the keypoints.
        # Thus, the output layer needs to have shape (Nx30),
        # with 2 values representing each of the 15 keypoint (x, y) pairs
        #
        # Some layers you might consider including:
        # maxpooling layers, multiple conv layers, fully-connected layers,
        # and other layers (such as dropout or batch normalization) to avoid
        # overfitting.
        ############################################################################

        self.maxpool = nn.MaxPool2d(2, 2)
        self.dropout = nn.Dropout(p=0.2)
        self.conv1 = nn.Conv2d(1, 32, 5)
        self.conv2 = nn.Conv2d(32, 64, 3)
        self.conv3 = nn.Conv2d(64, 128, 3)
        self.fc1 = nn.Linear(128*10*10, 30)

    def forward(self, x):

        # check dimensions to use show_keypoint_predictions later
        if x.dim() == 3:
            x = torch.unsqueeze(x, 0)
        ############################################################################
        # TODO: Define the forward pass behavior of your model
        # for an input image x, forward(x) should return the
        # corresponding predicted keypoints
        ############################################################################

        x = self.maxpool(F.relu(self.conv1(x)))
        x = self.maxpool(F.relu(self.conv2(x)))
        x = self.maxpool(F.relu(self.conv3(x)))
        x = x.view(x.size(0), -1)
        x = self.dropout(x)
        x = self.fc1(x)

        ############################################################################
        #                                                          END OF YOUR CODE
        ############################################################################
        return x
```

__init__的部分包括三個種不同 filter size 的 convolution、一個 size 為 2 的 maxpooling、一個 dropout 負責隨機捨去 node、一個 fully connected layer。

forward 的部分則是依據初始化所宣告的 object 來實現 CNN 的步驟，依序為 covl、ReLU、maxpooling，重複三回合後將資料攤平(view)後進入 fcl。

Define Functions for pl.Trainer：計算 loss、創建 dataloader、最佳化

```
    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.forward(batch[x])
        loss = self.hp["loss"](y_hat, batch[y].view(-1, 30))
        return loss

    def train_dataloader(self):
        return DataLoader(self.hp["train_dataset"], batch_size=self.hp["batch_size"], shuffle=True)

    def configure_optimizers(self):
        return torch.optim.Adam(self.parameters(), lr=self.hp["lr"])
```

Define Hyperparameters to the Model in a Dictionary：設定超參數

```
[15] hparams = {
        "loss":torch.nn.MSELoss(), "lr":1e-5, "train_dataset":train_dataset, "batch_size":20
    }
```

Test Whether the Model Follows the Basic Rules：測試 model 是否符合規定

```
[16] model = KeypointModel(hparams)
     test_keypoint_nn(model)
```

```
KeypointShapeTest passed.
ParamCountTest passed. Your model has 0.477 mio. params.
FileSizeTest passed. Your model is 1.9 MB large
All tests passed for your model. Tests passed: 3/3
```

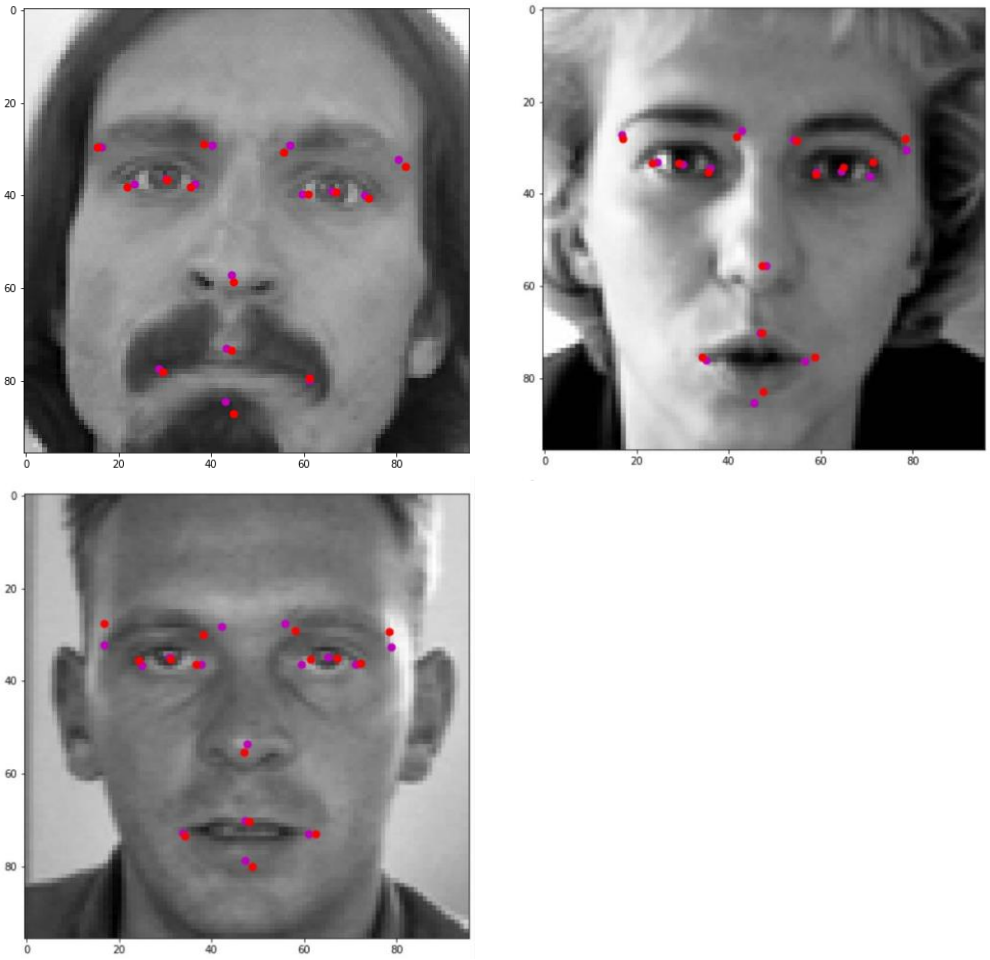
Model Training Using PyTorch Lightning：重複訓練，設定 50 個 epochs

```
################################################################################
# TODO - Train Your Model
################################################################################

import pytorch_lightning as pl

model = KeypointModel(hparams)

trainer = pl.Trainer(max_epochs=50, gpus=0, fast_dev_run=False)

trainer.fit(model)

################################################################################
#                                                          END OF YOUR CODE
################################################################################
```

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
HPU available: False, using: 0 HPUs

  | Name    | Type      | Params
-------------------------------------
0 | maxpool | MaxPool2d | 0
1 | dropout | Dropout   | 0
2 | conv1   | Conv2d    | 832
3 | conv2   | Conv2d    | 18.5 K
4 | conv3   | Conv2d    | 73.9 K
5 | fc1     | Linear    | 384 K
-------------------------------------
477 K     Trainable params
0         Non-trainable params
477 K     Total params
1.909     Total estimated model params size (MB)
Epoch 49: 100% 78/78 [22:35<00:00, 17.38s/it, loss=0.00187, v_num=2]
```

Visualize Predictions of the Model：以圖片檢視預測結果

```
[ ]  show_keypoint_predictions(model,  val_dataset)
```



Compute Validation Score：計算分數

```
[ ]  print("Score:",  evaluate_model(model,  val_dataset))
```

```
Score: 505.79374660804467
```

Save the Model：存檔

```
[ ]  save_model(model,  "facial_keypoints.p")
```

```
'models/facial_keypoints.p'
```

```
[ ]  # Now zip the folder for upload
     from exercise_code.util.submit import submit_exercise

     submit_exercise('exercise09')
```

```
relevant folders: ['exercise_code', 'models']
notebooks files: ['1_facial_keypoints.ipynb', 'Optional-spatial_batchnorm.ipynb']
Adding folder exercise_code
Adding folder models
Adding notebook 1_facial_keypoints.ipynb
Adding notebook Optional-spatial_batchnorm.ipynb
Zipping successful! Zip is stored under: /content/drive/My Drive/HW9/9/exercise_09_cleaned/exercise09.zip
```