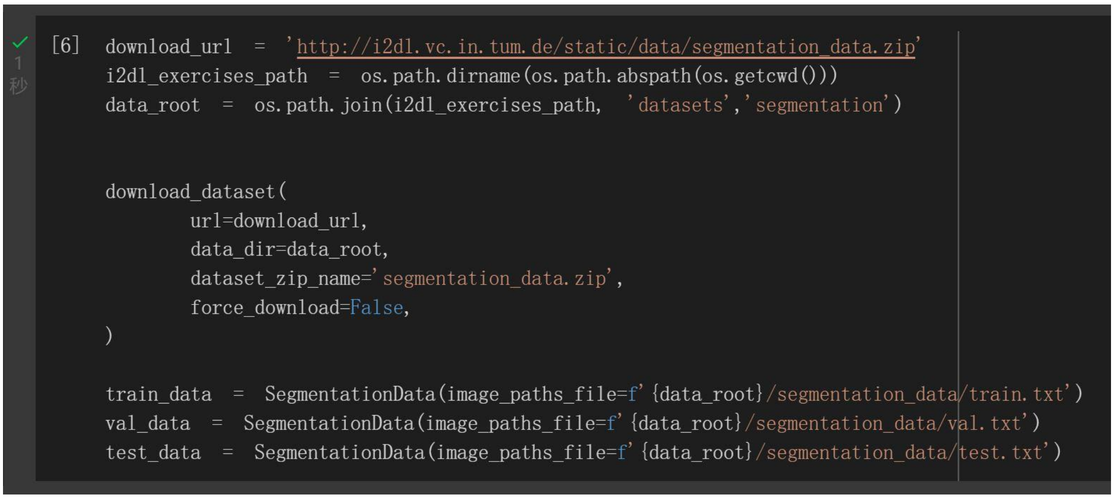
B093040024
郭晏涵

深度視覺

HW10: Segmentation

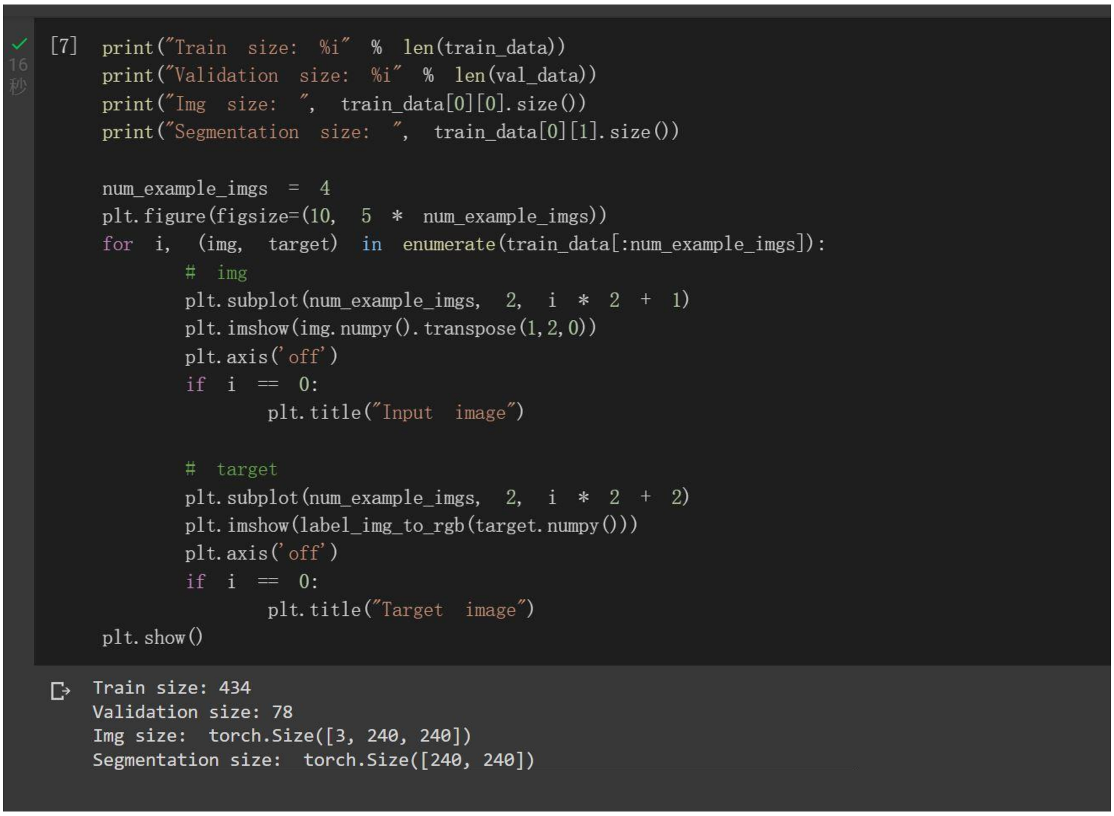
notebook 執行過程：

Load Data：下載資料集



```
1 秒
[6] download_url = 'http://i2dl.vc.in.tum.de/static/data/segmentation_data.zip'
    i2dl_exercises_path = os.path.dirname(os.path.abspath(os.getcwd()))
    data_root = os.path.join(i2dl_exercises_path, 'datasets','segmentation')


    download_dataset(
            url=download_url,
            data_dir=data_root,
            dataset_zip_name='segmentation_data.zip',
            force_download=False,
    )

    train_data = SegmentationData(image_paths_file=f'{data_root}/segmentation_data/train.txt')
    val_data = SegmentationData(image_paths_file=f'{data_root}/segmentation_data/val.txt')
    test_data = SegmentationData(image_paths_file=f'{data_root}/segmentation_data/test.txt')
```

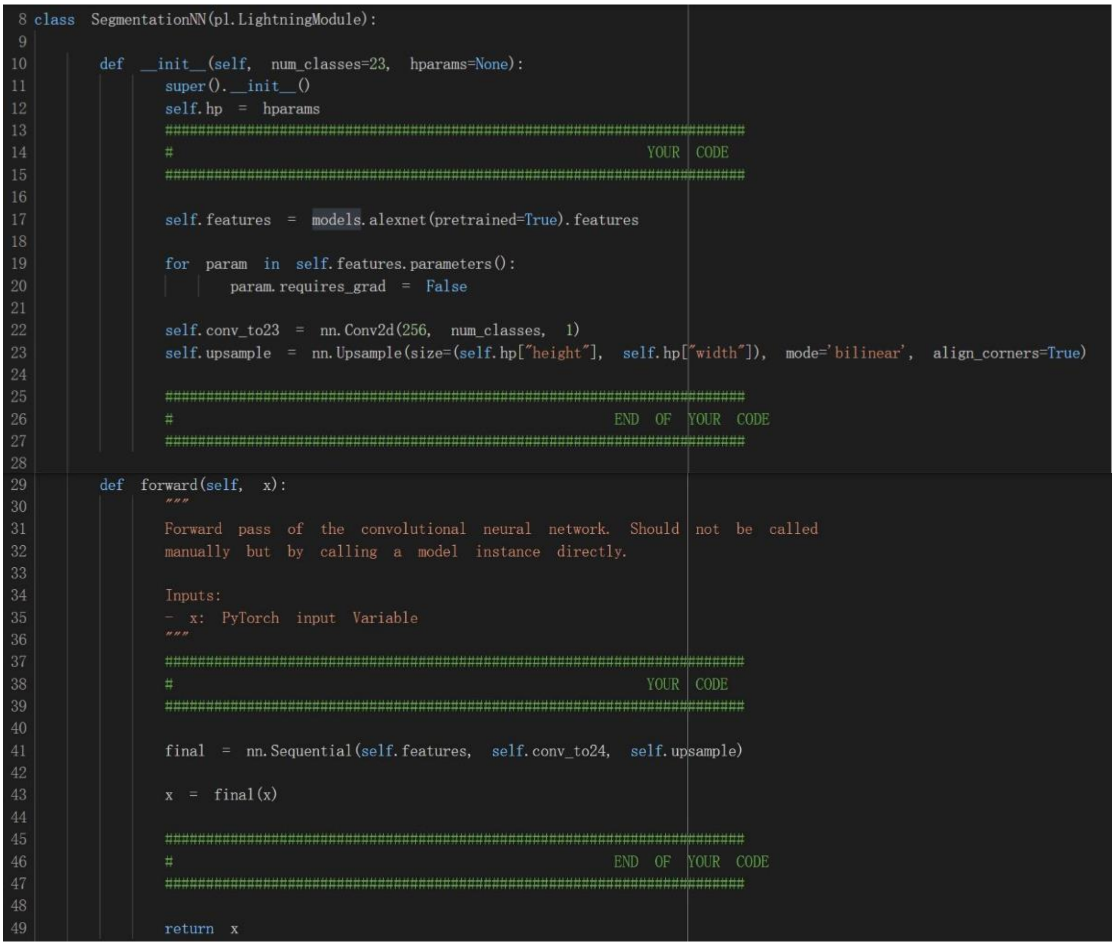
Visualize Data：查看資料狀況



```
16 秒
[7] print("Train size: %i" % len(train_data))
    print("Validation size: %i" % len(val_data))
    print("Img size: ", train_data[0][0].size())
    print("Segmentation size: ", train_data[0][1].size())

    num_example_imgs = 4
    plt.figure(figsize=(10, 5 * num_example_imgs))
    for i, (img, target) in enumerate(train_data[:num_example_imgs]):
            # img
            plt.subplot(num_example_imgs, 2, i * 2 + 1)
            plt.imshow(img.numpy().transpose(1,2,0))
            plt.axis('off')
            if i == 0:
                    plt.title("Input image")

            # target
            plt.subplot(num_example_imgs, 2, i * 2 + 2)
            plt.imshow(label_img_to_rgb(target.numpy()))
            plt.axis('off')
            if i == 0:
                    plt.title("Target image")
    plt.show()
```

```
Train size: 434
Validation size: 78
Img size:  torch.Size([3, 240, 240])
Segmentation size:  torch.Size([240, 240])
```

Design a Convolution Neural Network Model

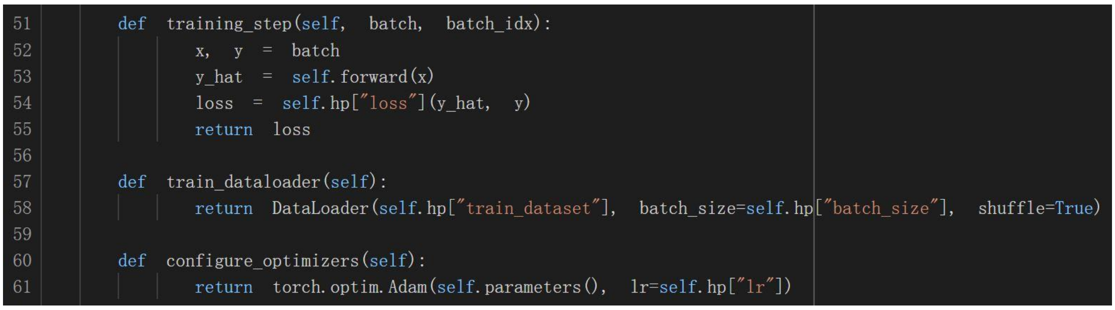


```
class SegmentationNN(pl.LightningModule):

    def __init__(self, num_classes=23, hparams=None):
        super().__init__()
        self.hp = hparams
        ########################################################################
        #                             YOUR CODE
        ########################################################################

        self.features = models.alexnet(pretrained=True).features

        for param in self.features.parameters():
            param.requires_grad = False

        self.conv_to23 = nn.Conv2d(256, num_classes, 1)
        self.upsample = nn.Upsample(size=(self.hp["height"], self.hp["width"]), mode='bilinear', align_corners=True)

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

    def forward(self, x):
        """
        Forward pass of the convolutional neural network. Should not be called
        manually but by calling a model instance directly.

        Inputs:
        - x: PyTorch input Variable
        """
        ########################################################################
        #                             YOUR CODE
        ########################################################################

        final = nn.Sequential(self.features, self.conv_to24, self.upsample)

        x = final(x)

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

        return x
```

__init__的部分包括一個 AlexNet 的 feature extractor、一個 output channel 為 23 的 convolution layer，和一個 upsampling layer。

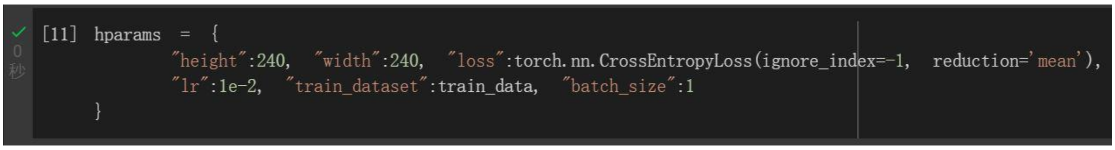
forward 的部分則是依據初始化所宣告的 object 來實現神經網路，經過 AlexNet 後將輸出轉為 23 個 class，最後使用 upsampling 將影像放大。

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



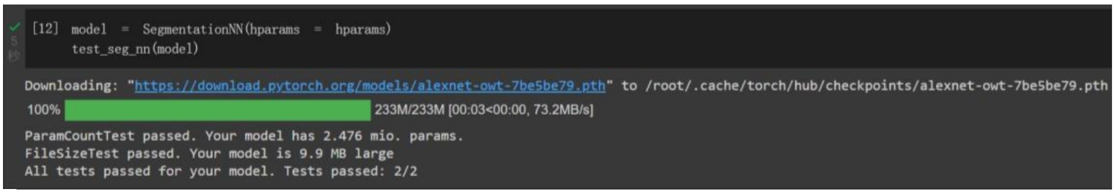
```
    def training_step(self, batch, batch_idx):
        x, y = batch
        y_hat = self.forward(x)
        loss = self.hp["loss"](y_hat, y)
        return loss

    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



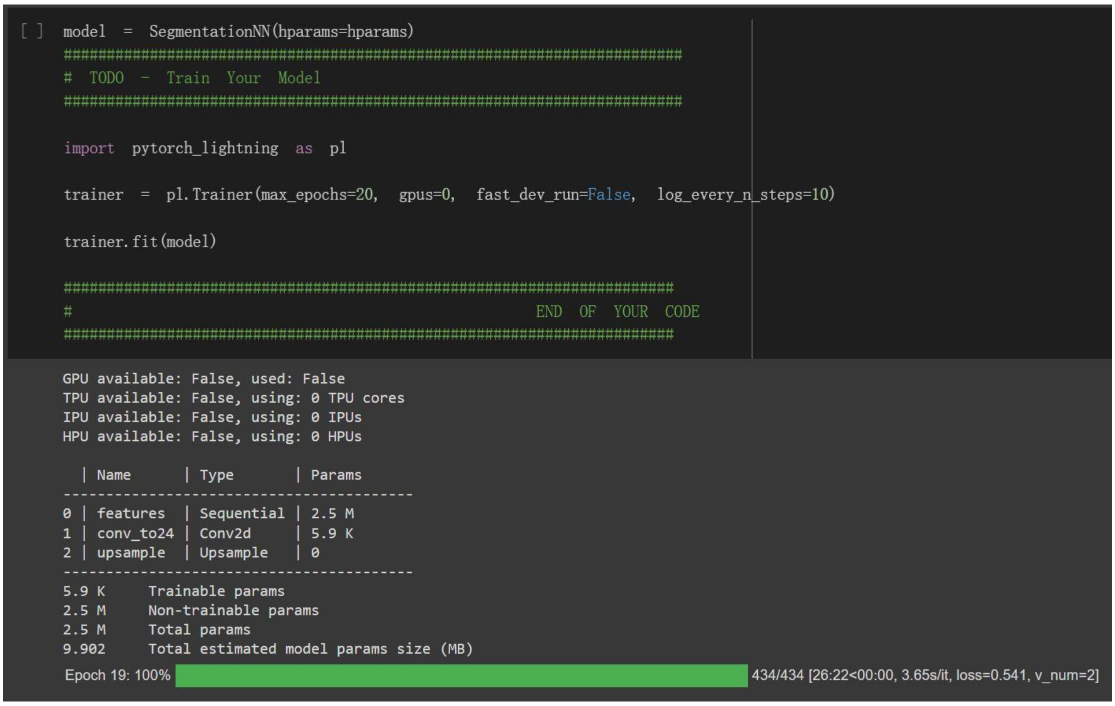
```
[11] hparams = {
         "height":240, "width":240, "loss":torch.nn.CrossEntropyLoss(ignore_index=-1, reduction='mean'),
         "lr":1e-2, "train_dataset":train_data, "batch_size":1
     }
```

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



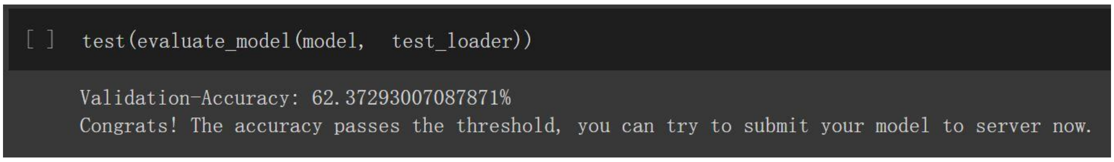
```
[12] model = SegmentationNN(hparams = hparams)
     test_seg_nn(model)

Downloading: "https://download.pytorch.org/models/alexnet-owt-7be5be79.pth" to /root/.cache/torch/hub/checkpoints/alexnet-owt-7be5be79.pth
100% 233M/233M [00:03<00:00, 73.2MB/s]
ParamCountTest passed. Your model has 2.476 mio. params.
FileSizeTest passed. Your model is 9.9 MB large
All tests passed for your model. Tests passed: 2/2
```

Model Training Using PyTorch Lightning：重複訓練，設定 20 個 epochs



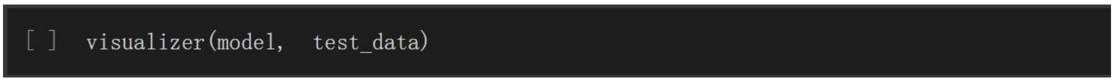
```
[ ] model = SegmentationNN(hparams=hparams)
    ########################################################################
    # TODO - Train Your Model
    ########################################################################

    import pytorch_lightning as pl

    trainer = pl.Trainer(max_epochs=20, gpus=0, fast_dev_run=False, log_every_n_steps=10)

    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 | features  | Sequential | 2.5 M
    1 | conv_to24 | Conv2d     | 5.9 K
    2 | upsample  | Upsample   | 0
    ----------------------------------------
    5.9 K     Trainable params
    2.5 M     Non-trainable params
    2.5 M     Total params
    9.902     Total estimated model params size (MB)
    Epoch 19: 100% 434/434 [26:22<00:00, 3.65s/it, loss=0.541, v_num=2]
```

Compute Validation Accuracy：計算準確率

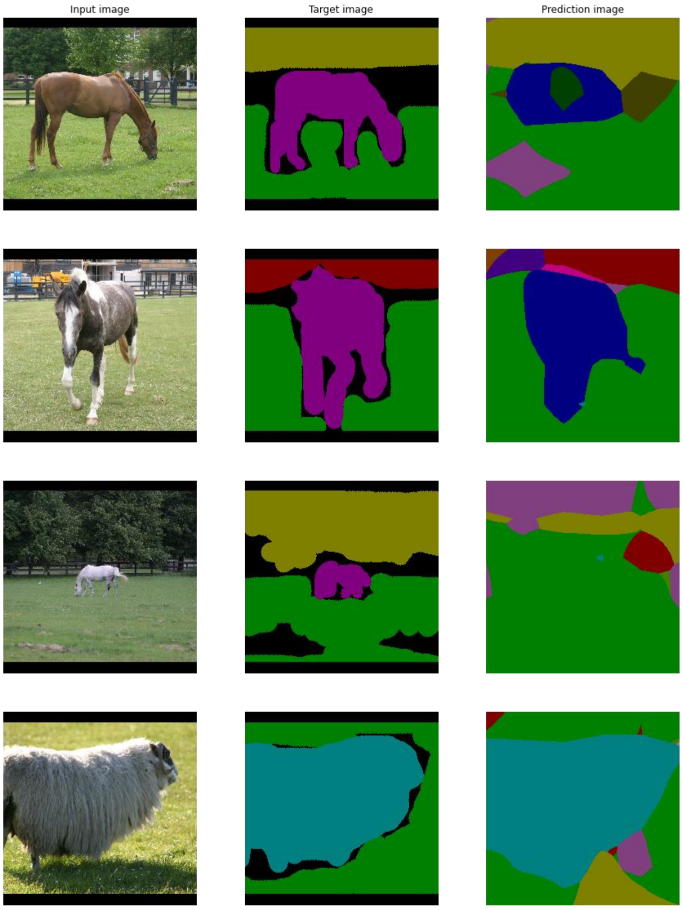


```
[ ]  test(evaluate_model(model, test_loader))

     Validation-Accuracy: 62.37293007087871%
     Congrats! The accuracy passes the threshold, you can try to submit your model to server now.
```

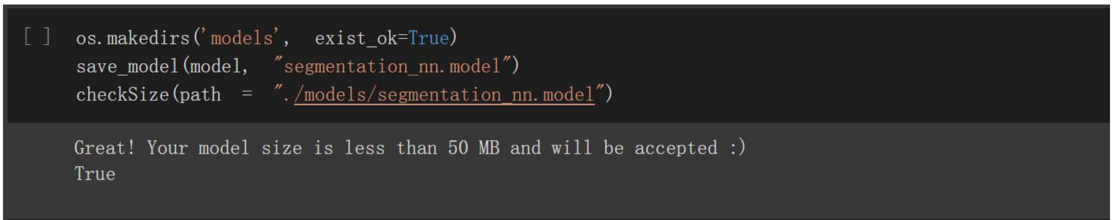
Visualize Predictions of the Model：以圖片檢視預測結果



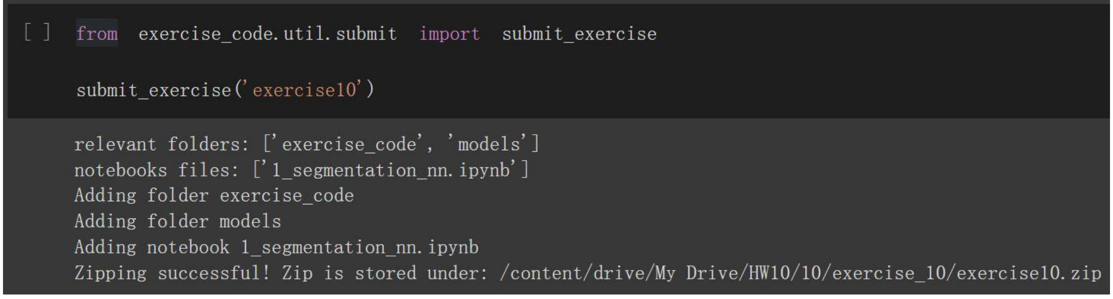
```
[ ]  visualizer(model, test_data)
```



Save the Model：存檔



```
[ ]  os.makedirs('models', exist_ok=True)
     save_model(model, "segmentation_nn.model")
     checkSize(path = "./models/segmentation_nn.model")

     Great! Your model size is less than 50 MB and will be accepted :)
     True
```



```
[ ]  from exercise_code.util.submit import submit_exercise

     submit_exercise('exercise10')

     relevant folders: ['exercise_code', 'models']
     notebooks files: ['1_segmentation_nn.ipynb']
     Adding folder exercise_code
     Adding folder models
     Adding notebook 1_segmentation_nn.ipynb
     Zipping successful! Zip is stored under: /content/drive/My Drive/HW10/10/exercise_10/exercise10.zip
```