對本書的讚譽

這本不可或缺的技術指南提供了清楚且實用的操作指導，協助你實作 *stable diffusion* 與微調語言模型，是所有 *AI* 開發者的必備之作。

—*Vicki Reyzelman*，*Mave Sparks* 首席 *AI* 解決方案架構師

作為一本全方面的實用指南，這本專門為了渴望掌握生成式 *AI* 的讀者而寫的書籍結合了理論與現實應用，提供可執行的 *Python* 程式碼與清晰的見解，探討語言模型和擴散技術的基礎知識，以及微調和建立文本生成圖像應用程式…等進階主題，帶領讀者在這個快速發展的領域中建構模型、發揮創意、並保持領先。他們的專業素養與實踐方法，使本書成為初學者和老練實踐者的無價資源。

—*Anil Sood*，*Ernst & Young US* 資深經理

這本無價的指南，為我們掀開生成式 *AI* 的神秘面紗，將實用的見解與動手實踐的技術和範例結合起來，並涵蓋多個領域。如果你對未來的 *AI* 有興趣，這是必讀之作。

—*Vishwesh Ravi Shrimali*，汽車業工程師

這本精心編撰的指南可以幫助廣泛的讀者理解複雜的 *AI* 概念。作者們清楚地解釋 *transformer* 和擴散模型，如果你想要真正瞭解當今生成式 *AI* 的基石，這本書是絕佳讀物。

—*Sai M Vuppalapati*，*Tubi TV* 資料與 *AI/ML* 平台產品經理

本書對任何想要瞭解生成式 *AI* 潛力的人來說是一座寶庫，它專門探討如何解決彼此相關的現實問題，並提供實用的指導，巧妙地將複雜的概念銜接起來，讓業餘愛好者及專業人士更容易接觸生成式 *AI*。對於準備投入這個充滿活力的領域並探索生成式 *AI* 之威力的人來說，這是必讀之作。

—*Lipi Deepaakshi Patnaik*，*Zeta* 高級軟體開發者

碁峰 www.gotop.com.tw

transformer 區塊

在做了簡單的語言模型實驗後，接下來要介紹 transformer-based 語言生成模型的架構圖，如圖 2-9 所示。

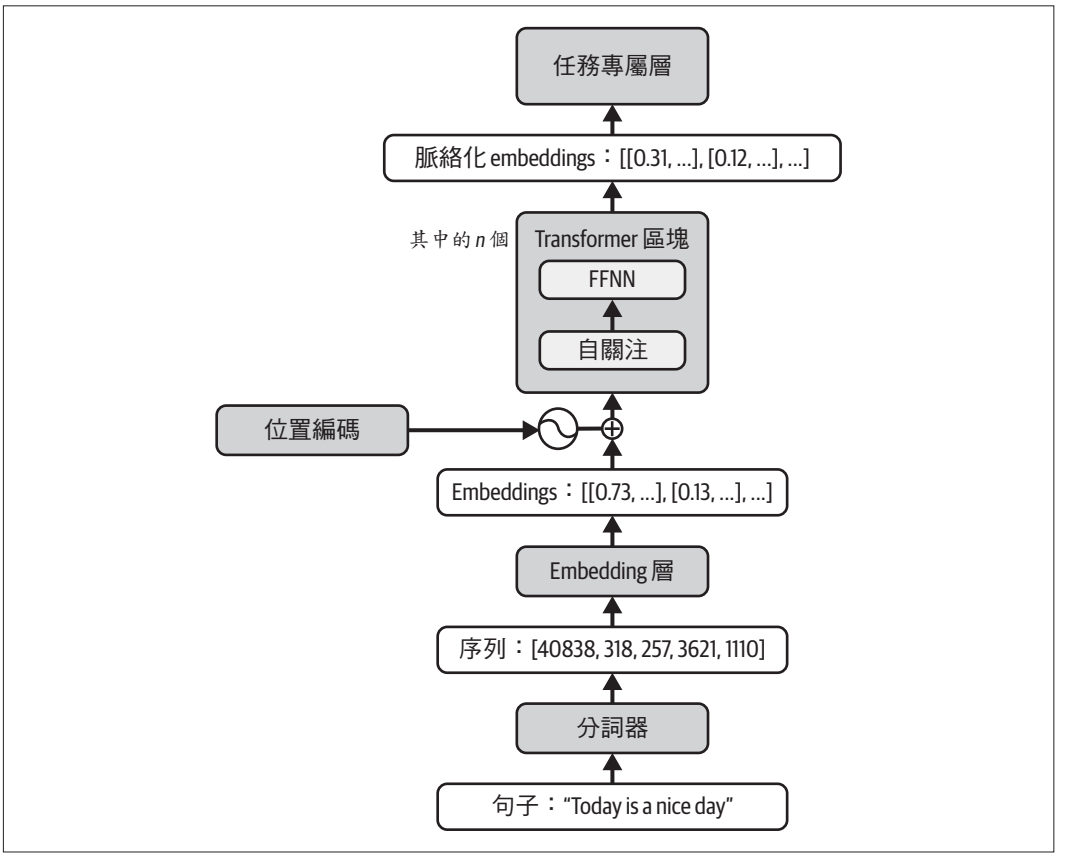


圖 2-9　transformer-based 語言模型的架構

碁峰 www.gotop.com.tw

隨著 transformer 模型在文本領域的成功，其他領域的社群也開始試圖將這些技術應用在其他模態上，導致 transformer 模型被用來處理圖像識別、分割、物體檢測、影片理解…等任務，如圖 2-14 所示。

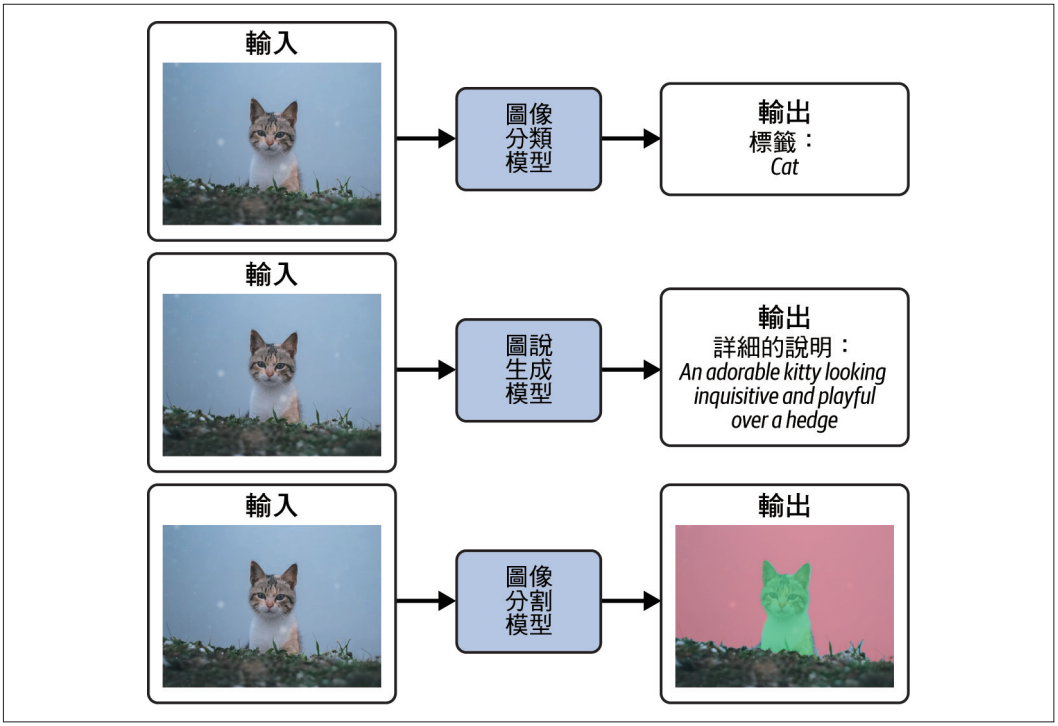


圖 2-14　transformer 模型可於圖像分類、物體檢測、圖像分割等任務

長期以來，摺積神經網路被普遍地視為電腦視覺技術的首選 SOTA 模型，隨著 Vision Transformers（ViTs，*http://arxiv.org/abs/2010.11929*）的引入，近年來的研究開始探索如何使用關注機制和 transformer 技術來處理視覺任務。ViTs 並未完全捨棄 CNN。在圖像處理流程中，CNN 被用來提取圖像的特徵圖，以檢測高階的邊緣、紋理…等模式。從 CNN 取得的特徵圖會被分成尺寸一致、不重疊的圖塊（patches）。這些圖塊可以視為類似一系列詞元的東西，所以，關注機制可以學習不同位置的圖塊之間的關係。

碁峰 www.gotop.com.tw

然而，相較於 CNN ，ViTs 需要更多資料（3 億張圖像！）和運算資源才能得到好結果。近年來，相關的研究有了更多進展，例如 DeiT 模型透過 CNN 常見的擴增與正則化技術，得以利用 transformer 架構來處理中型資料集（120 萬張圖像）。其他模型如 DETR、SegFormer 和 Swin Transformer 也進一步推動了這個領域，支援圖像分類、物體檢測、圖像分割、影片分類、文件理解、圖像恢復、超解析度…等多項任務。

零樣本圖像分類是 transformer-based 圖像模型的強大應用之一。傳統的圖像分類器需要用固定的類別來訓練，但零樣本圖像分類可在推論階段指定類別，可讓你靈活地使用單一模型來處理各種圖像分類任務，即使是模型未曾明確地學過如何執行的任務。作為示範，我們將使用 *PIL* 程式庫（一種被廣泛用來進行視覺相關預處理的工具）來載入一張圖像：

```
import requests
from PIL import Image

from genaibook.core import SampleURL

# 下載圖像，並用 PIL 程式庫來載入它
url = SampleURL.CatExample
image = Image.open(requests.get(url, stream=True).raw)
image
```

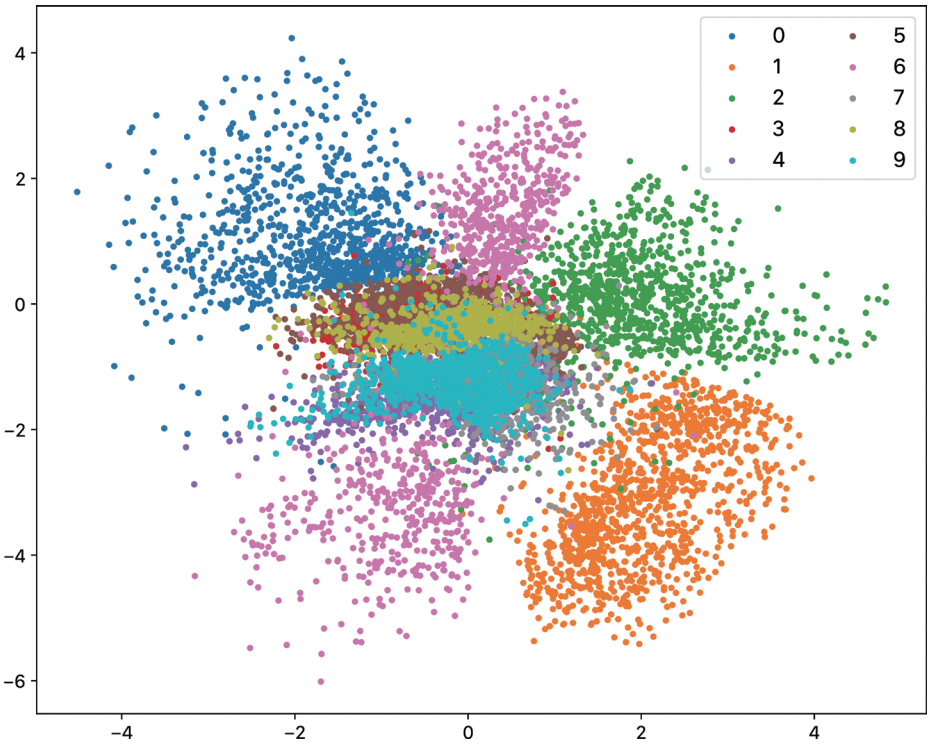
碁峰 www.gotop.com.tw

將潛空間視覺化

我們只用兩個維度的潛空間來輕鬆地將它的結構視覺化。接下來要展示測試資料集的所有已編碼向量，我們使用 `label` 欄位來為每一個類別指定不同的顏色。x 軸是已編碼向量的第一個值，y 軸是第二個值：

```
images_labels_dataloader = DataLoader(mnist["test"], batch_size=512)

import pandas as pd

df = pd.DataFrame(
    {
        "x": [],
        "y": [],
        "label": [],
    }
)

for batch in tqdm(
    iter(images_labels_dataloader), total=len(images_labels_dataloader)
):
    encoded = ae_model.encode(batch["image"].to(device)).cpu()
    new_items = {
        "x": [t.item() for t in encoded[:, 0]],
        "y": [t.item() for t in encoded[:, 1]],
        "label": batch["label"],
    }
    df = pd.concat([df, pd.DataFrame(new_items)], ignore_index=True)

plt.figure(figsize=(10, 8))

for label in range(10):
    points = df[df["label"] == label]
    plt.scatter(points["x"], points["y"], label=label, marker=".")

plt.legend();
```


碁峰 www.gotop.com.tw



AutoEncoder 在潛空間中將資料集的不同圖像分成壁壘分明的不同區域。例如，數字 0 的圖像（深藍色）與數字 1 的圖像（橘色）被明顯分開。記住，在訓練過程中，我們並未使用關於圖像標籤的任何資訊，但即使如此，模型仍然根據資料點的視覺特徵，自動將它分到不同的區域。然而，這個過程是在完全未受控制的情況下進行的，所以潛空間的形狀或結構是不可預測的。

因此，雖然潛空間足夠豐富，可以捕捉資料集的圖像特徵，但目前我們還不清楚如何用它來生成圖像。在理想情況下，為了生成與 MNIST 類似的新圖像，我們要將編碼器移開，並將潛空間裡的隨機樣本傳給解碼器。然而，我們可以從這張圖中看到一些問題：

- 表徵空間是朝著所有方向分散的。
- 在中央有很多點是重疊的，而且有大片的空白區域。
- 圖不對稱，在 y 軸上，負值多於正值。

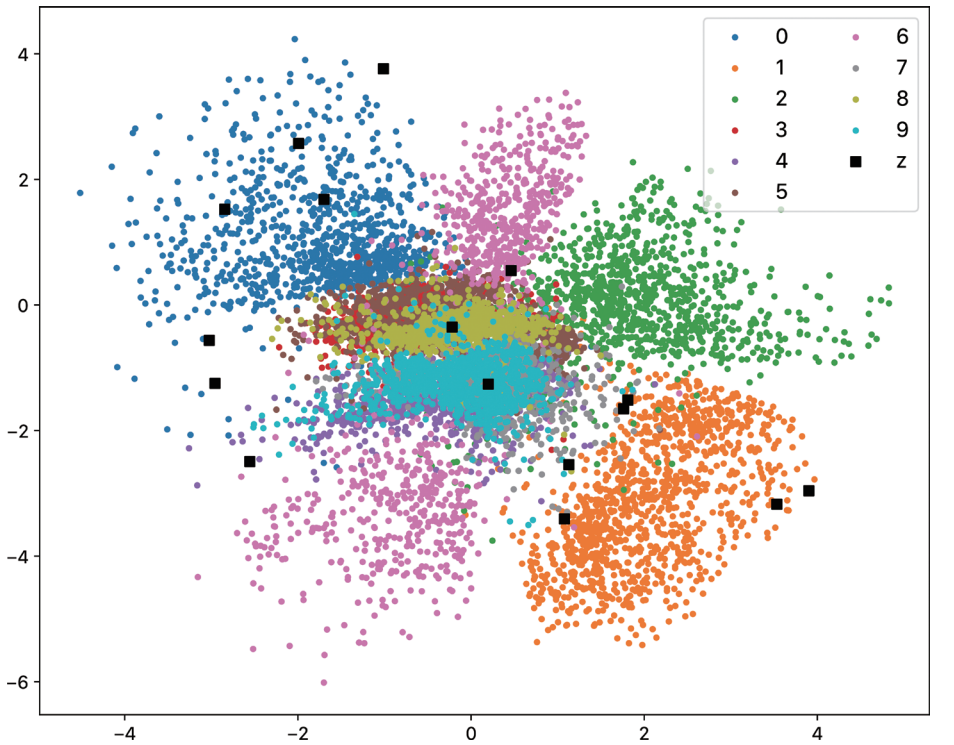
碁峰 www.gotop.com.tw

所以選出潛空間的適當區域以產生高品質的圖像是有挑戰性的任務。我們來看看使用解碼器產生的圖像範例，先產生隨機的潛樣本（通常寫成 z）：

```
N = 16  # 我們將生成 16 個點
z = torch.rand((N, 2)) * 8 - 4
```

接著將生成的潛樣本畫在之前展示的潛空間表徵上，將它視覺化：

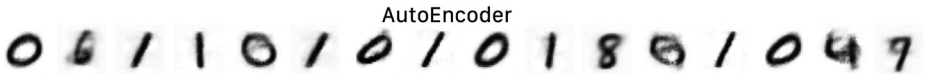
```
plt.figure(figsize=(10, 8))

for label in range(10):
    points = df[df["label"] == label]
    plt.scatter(points["x"], points["y"], label=label, marker=".")

plt.scatter(z[:, 0], z[:, 1], label="z", marker="s", color="black")
plt.legend();
```



碁峰 www.gotop.com.tw

最後，讓解碼器使用剛才建立的潛樣本來生成圖像：

```
ae_decoded = ae_model.decode(z.to(device))
show_images(ae_decoded.cpu(), imsize=1, nrows=1, suptitle="AutoEncoder")
```



當樣本接近潛空間中的某個區域時，模型生成的圖像很合理，但是當樣本位於這些區域之外時，圖像的說服力會大幅下降。此外，有些數字被過度表示（overrepresented），因為模型在潛空間中為它們分配比較大的區域。

接下來將討論如何使用不同類型的 AutoEncoder 讓潛空間更有秩序一些，以及如何讓生成更容易。

在繼續介紹之前，下面有一些（逐步提高難度的）練習，現在你可以進行這些練習（或是在看完本章時），來更深入理解概念：

1. 如果模型是用 16 維潛空間來訓練的，它的生成效果怎麼樣？
2. 使用之前的參數來重新訓練模型（直接執行本章的程式碼），但使用不同的隨機數來初始化[5]，並將潛空間視覺化。潛空間的形狀和結構可能不同。這是你意料中的結果嗎？為什麼？
3. 編碼器提取的圖像特徵的品質如何？丟棄 AutoEncoder 的解碼器部分，並在編碼器後面建立一個數字分類器。例如，你可以訓練幾層線性層，並在它們之間使用非線性函數。最終的線性層應輸出一個 10 維的向量，代表資料集的 10 個標籤。只訓練這些階層，而不必更新編碼器的權重。你能夠獲得多少準確率？具有 16 維潛空間的模型，是否比只有 2 維的模型更好？

Variational AutoEncoders

在上一節，我們探討了簡單的 AutoEncoder 如何在低維潛空間中學習輸入資料的有效表徵。AutoEncoder 可以準確地編碼任何樣本，並在稍後恢復（或解碼）它。這個功能很適合用來提取特徵或製作資料表徵，但不適合生成新樣本。

5　你可以使用 `torch.manual_seed(num)` 來指定種子。

碁峰 www.gotop.com.tw

簡單的 UNet

為了更深入瞭解 UNet 結構，我們來從零開始建立一個簡單的 UNet。圖 4-7 是一個基本 UNet 的架構圖。

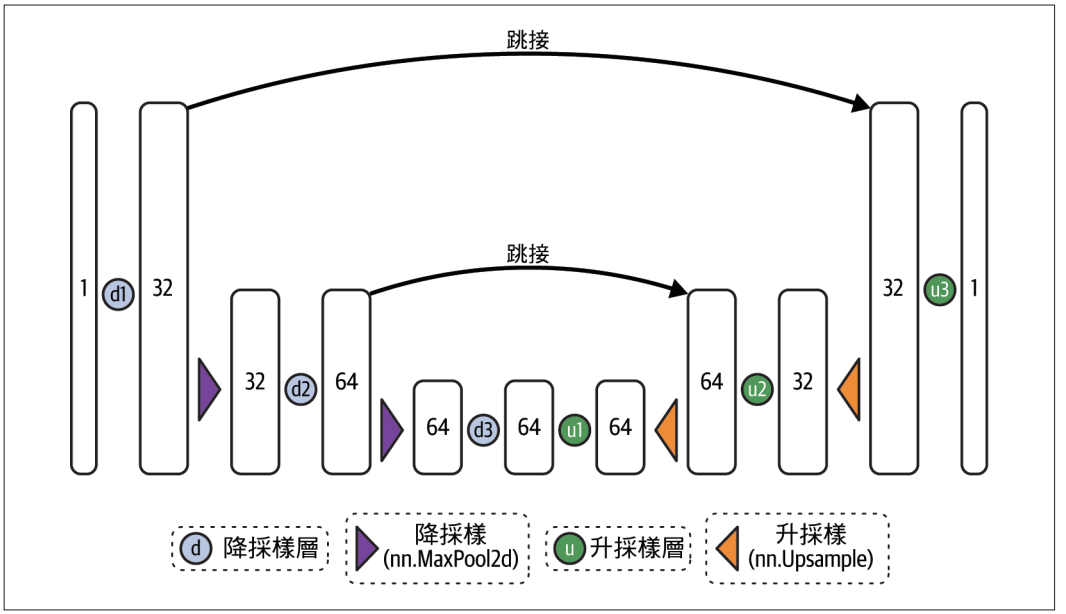


圖 4-7　基本 UNet 的架構

我們將設計一個處理單通道圖像（例如灰階圖像）的 UNet，它可以用來建立一個處理 MNIST 之類的資料集的擴散模型。我們將在降採樣路徑中使用三層，在升採樣路徑中再使用三層。每一層有一個摺積層，然後有一個啟動函數，然後是升採樣或降採樣步驟，具體取決於它們位於編碼路徑還是解碼路徑中。如前所述，跳接會直接從降採樣區塊連到升採樣區塊。實現跳接的方法有很多種。

我們的做法是將降採樣區塊的輸出加到對應的升採樣區塊的輸入。另一種做法是將降採樣區塊的輸出與升採樣區塊的輸入連接起來，甚至可以在跳接中加入額外的階層。

碁峰 www.gotop.com.tw

在最初的做法中，我們先保持簡單，以下是這個網路的程式碼：

```
from torch import nn


class BasicUNet(nn.Module):
    """ 最簡單的 UNet 實作。"""

    def __init__(self, in_channels=1, out_channels=1):
        super().__init__()
        self.down_layers = nn.ModuleList(
            [
                nn.Conv2d(in_channels, 32, kernel_size=5, padding=2),
                nn.Conv2d(32, 64, kernel_size=5, padding=2),
                nn.Conv2d(64, 64, kernel_size=5, padding=2),
            ]
        )
        self.up_layers = nn.ModuleList(
            [
                nn.Conv2d(64, 64, kernel_size=5, padding=2),
                nn.Conv2d(64, 32, kernel_size=5, padding=2),
                nn.Conv2d(32, out_channels, kernel_size=5, padding=2),
            ]
        )

        # 使用 SiLU 觸發函式，這種函式的各種特性
        # 都有很好的表現（平滑性、非單調性…等）。
        self.act = nn.SiLU()
        self.downscale = nn.MaxPool2d(2)
        self.upscale = nn.Upsample(scale_factor=2)

    def forward(self, x):
        h = []
        for i, l in enumerate(self.down_layers):
            x = self.act(l(x))
            if i < 2:  # 除了第三個（最後一個）降採樣層以外
                h.append(x)  # 儲存輸出以用於跳接
                x = self.downscale(x)  # 為下一層降採樣

        for i, l in enumerate(self.up_layers):
            if i > 0:  # 除了第一個升採樣層以外
                x = self.upscale(x)  # 升採樣
                x += h.pop()  # 取出已儲存的輸出（跳接）
            x = self.act(l(x))

        return x
```


碁峰 www.gotop.com.tw

當你傳入一個外形為 (1, 28, 28) 的灰階圖像時，它在這個模型中經歷的過程會是：

1. 圖像經過降採樣區塊。第一層是有 32 個過濾器的 2D 摺積，它會將圖像轉換為外形 [32, 28, 28]。
2. 圖像接著被最大池化降採樣，外形變成 [32, 14, 14]。MNIST 資料集的圖像是白色的數字、黑色的背景（黑色以數字 0 來表示）。我們用最大池化來選擇一塊區域內的最大值，關注最亮的像素 [12]。
3. 圖像經過第二個降採樣區塊。第二層是有 64 個過濾器的 2D 摺積層，它會將圖像轉換為 [64, 14, 14]。
4. 經過再次降採樣後，外形變成 [64, 7, 7]。
5. 在降採樣區塊中還有第三層，但這次不做降採樣，因為現在已經使用非常小的 7×7 區塊了。這會讓外形維持 [64, 7, 7]。
6. 然後做逆向處理，將圖像升採樣至 [64, 14, 14]、[32, 14, 14]，最後是 [1, 28, 28]。

使用這個架構和 MNIST 訓練集訓練出來的擴散模型可以產生如圖 4-8 所示的樣本（程式碼可在補充教材中找到（*https://oreil.ly/handsonGenAIcode*），為了節省篇幅，在此省略）。

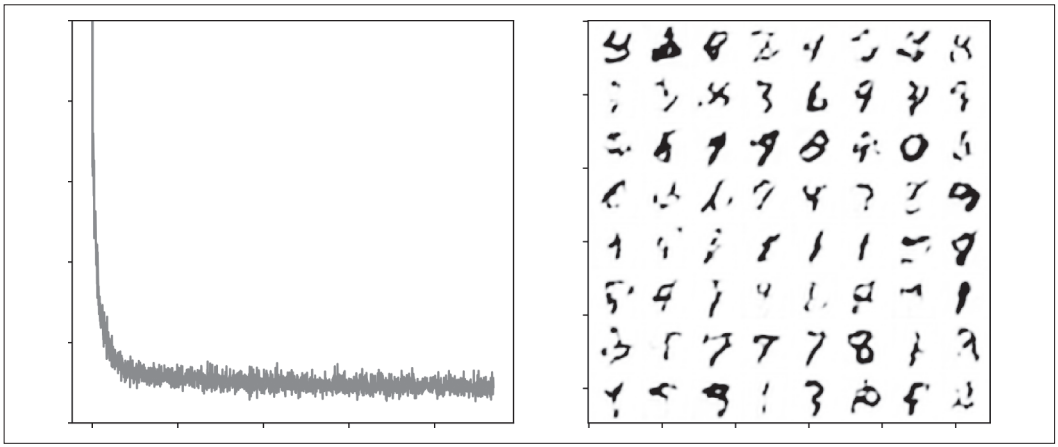


圖 4-8　基本 UNet 的損失曲線和輸出的結果

12 為了視覺化，我們展示的 MNIST 是白底黑字，但訓練資料集採取反過來的做法。

碁峰 www.gotop.com.tw

改進 UNet

這個簡單的 UNet 確實可以處理這種相對簡單的任務，但若要處理更複雜的資料呢？以下有一些選項：

加入更多參數

為此，你可以在每一個區塊中使用多個摺積層、在每一個摺積層中使用更多過濾器，或是讓網路更深。

加入正規化，例如批次正規化

批次正規化可以幫助模型更快速且更穩定地學習，因為它可以確保每一層的輸出的中間值是 0，標準差是 1。

加入正則化，例如 *dropout*

dropout 可防止模型過度擬合訓練資料，在處理較小的資料集時，這件事特別重要。

加入關注機制

加入自關注層可讓模型在不同時間專注於圖像的不同部分，讓 UNet 能夠學習更複雜的功能。加入類似 transformer 的關注層也可以增加可學習參數的數量。但是在處理較高解析度的圖像時，關注層的計算成本比常規的摺積層高得多，因此，關注層通常只在處理較低解析度的圖像時使用（例如 UNet 中的低解析度區塊）。

圖 4-9 是使用 *diffusers* 程式庫的 UNet 來處理 MNIST 的結果，UNet 包含上述的改善機制。

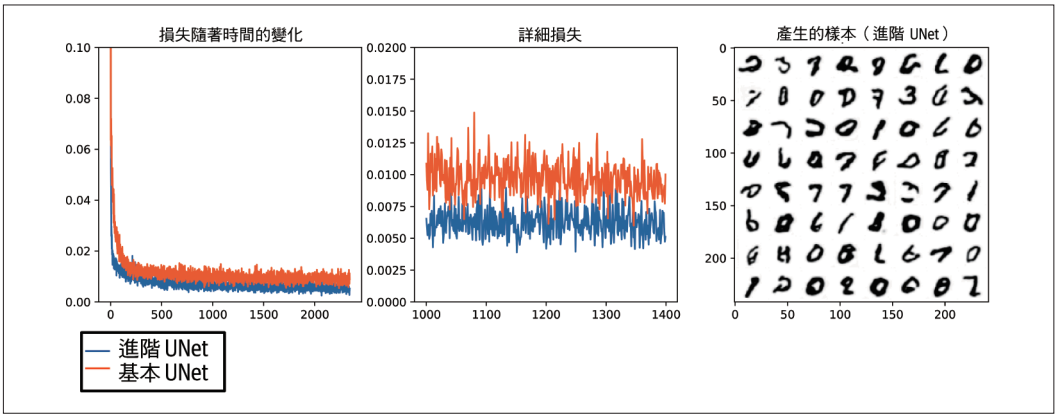


圖 4-9　diffusers UNet 的損失和產生的結果，UNet 改善了基本架構

碁峰 www.gotop.com.tw

替代架構

最近有幾個擴散模型的替代架構被提出（圖 4-10）。

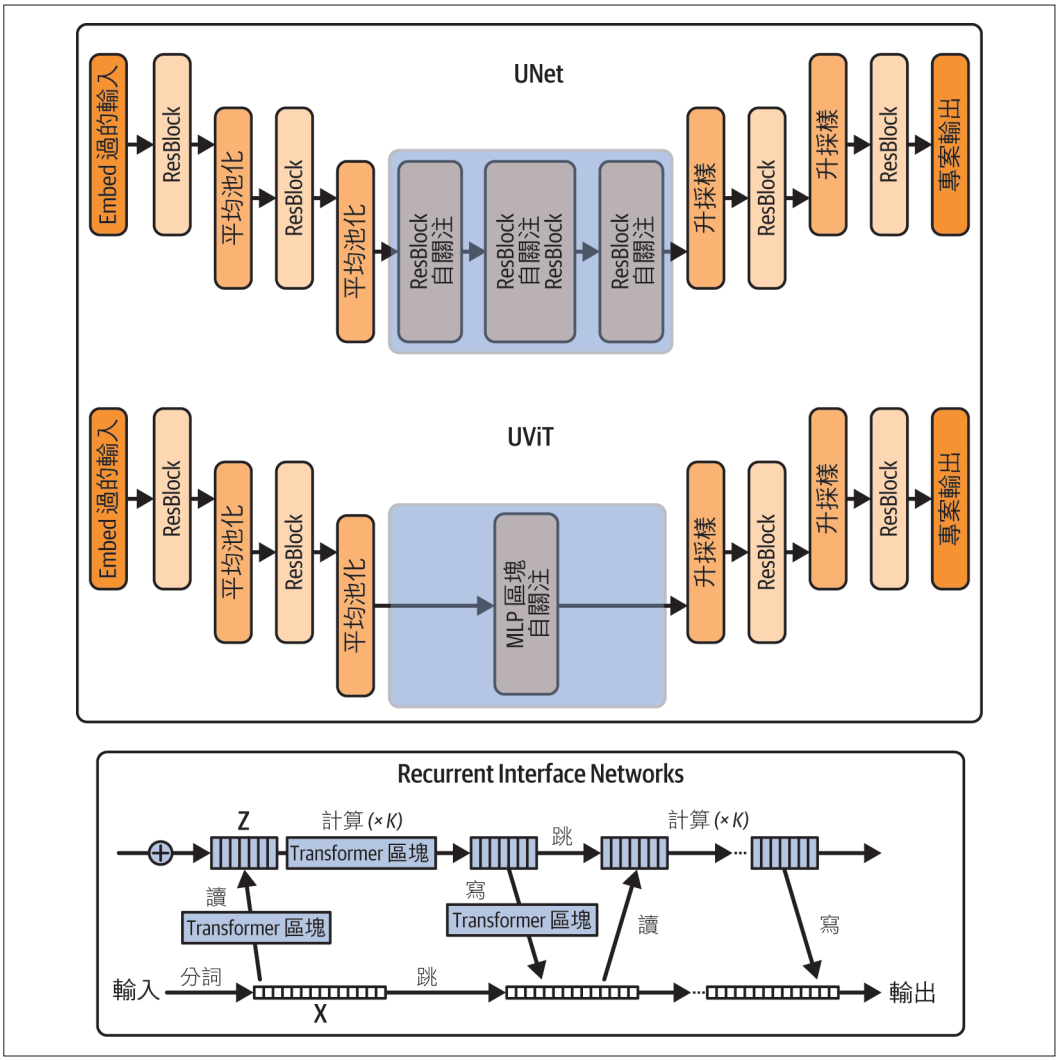


圖 4-10　比較 UNet 與 UViT 和 RIN

碁峰 www.gotop.com.tw

以下是這些架構的簡介：

Transformer

Diffusion Transformers 論文（*http://arxiv.org/abs/2212.09748*）指出，基於 transformer 的架構可以用來訓練擴散模型，並且有卓越的效果。然而，transformer 架構在處理極高解析度時的運算與記憶體需求仍然是一項挑戰。

UViT

Simple Diffusion 論文提出的 UViT 架構試圖結合兩者的優勢，將 UNet 的中間層換成一大疊的 transformer 區塊。該論文的關鍵觀點在於，將大部分的計算集中在 UNet 的低解析度區塊上可以更有效地訓練高解析度的擴散模型。他們使用額外的預處理技術來處理極高解析度，這種技術稱為 wavelet transform，可降低輸入圖像的空間解析度，同時透過額外的通道來盡量保留資訊，進一步減少處理高空間解析度的計算量。

Recurrent Interface Networks (RINs)

RIN 論文（*http://arxiv.org/abs/2212.11972*）採取類似的做法，先將高解析度的輸入對映到更容易管理的低維潛表徵中，然後用一疊 transformer 區塊來處理它們，最後解碼成圖像。這篇論文也引入遞迴的概念，也就是將先前的處理步驟的資訊傳給模型，這有助於改善擴散模型的迭代改進程序。

Flux、Stable Diffusion 3、PixArt-Σ，以及文字轉圖像的 Sora 都是高品質的擴散 transformer 模型。基於 transformer 的方法能否完全取代 UNet 成為擴散模型的首選架構，或是像 UViT 和 RIN 這樣的混合方法能否成為最有效的解決方案，仍有待觀察。

深入探討：擴散目標

我們已經討論擴散模型如何處理有雜訊的輸入，及如何學習移除雜訊了。你可能認為，網路的預測目標當然是圖像的無雜訊版本，我們稱之為 `x0`。然而，在程式碼中，我們比較的是「模型的預測」與「用來建立有雜訊的版本的單位變異數雜訊」（通常稱為 epsilon 目標，`eps`），雖然兩者在數學裡看起來相同，因為一旦知道雜訊和時步，即可推導出 `x0`，反之亦然，但預測目標的選擇會對不同時步的損失大小產生一些微妙的影響，從而影響模型最擅長移除哪些雜訊程度。

碁峰 www.gotop.com.tw

對模型來說，預測雜訊比直接預測目標資料更容易。因為雜訊在每一個時步中遵循已知的分布，預測兩個時步之間的差異通常比預測目標資料的絕對值更簡單。

為了幫助理解，我們可以將不同的目標在不同時步中的情況視覺化。如圖 4-11 所示，輸入圖像和隨機雜訊是相同的（前兩列），但是在第三列的含雜訊圖像，則隨著時步的不同，而被加入不同程度的雜訊。

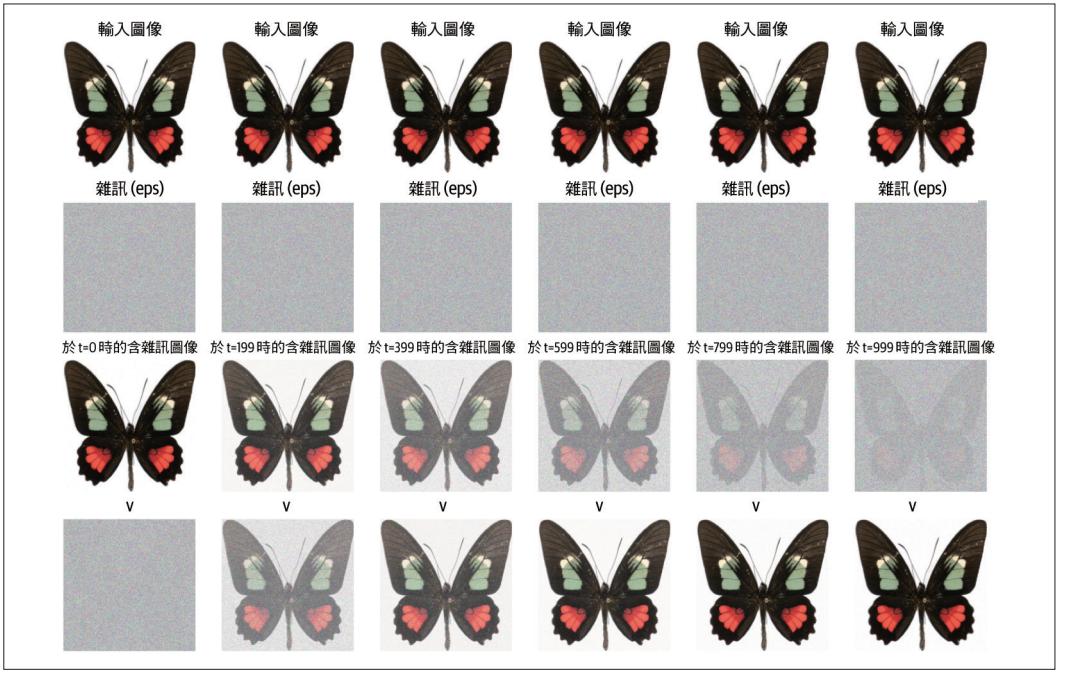


圖 4-11　比較 eps、x0 和 v 目標：eps 試著預測每一個時步中加入的雜訊，x0 預測移除雜訊後的圖像，而 v 則是兩者的混合

在雜訊量極低的情況下，以 x0 作為目標非常簡單（含雜訊的圖像幾乎與輸入圖像相同），但準確預測雜訊是幾乎不可能做到的事情。同理，在雜訊量極高的情況下，以 eps 作為目標很簡單（含雜訊的圖像幾乎等於所加入的純雜訊），但準確預測移除雜訊後的圖像是幾乎不可能做到的事情。如果使用 x0 作為目標，那麼訓練過程會給較低的雜訊水準較小的權重。

碁峰 www.gotop.com.tw