AI-Deep Learning 2023(Final) Exam time (9:20-11:20)

1. (30%) Given an LSTM model as shown in Figure 1, point out (i.e., write down the symbols) which is the input gate, output gate, and forget gate, respectively (10%). Then explain the function of each gate (10%). Suppose that we have an application that needs to predict an output y for a sequence of three inputs (x1, x2, x3) (i.e., three time steps); draw an unfolded figure of LSTM for this application. (10%)

給定如圖 1 所示的 LSTM 模型，指出（即記下符號）分別是輸入門、輸出門和忘記門 （10%）。

然後解釋每個門的功能（10%）。

假設我們有一個應用程式需要預測三個輸入序列（x1、x2、x3）（即三個時間步長）的輸出 y;為此應用程式繪製 LSTM 的展開圖。(10%)

**LSTM (長短期記憶)**

LSTM網絡設計用於記住長期依賴，讓模型可以長期的計算每個輸入的影響程度，並由一系列單元組成，每個單元包含三個門：遺忘門、輸入門和輸出門。

1. **遺忘門ft**: 決定從前一個單元狀態中應該丟棄或保留的信息。使用Sigmoid函數輸出0到1之間的值。
2. **輸入門it**: 決定將哪些新信息存儲在單元狀態中。結合輸入數據和之前的隱藏狀態hidden state來更新單元狀態。
3. **輸出門ot**: 控制輸出，決定單元狀態的哪一部分應作為下一時間步的隱藏狀態輸出。
4. AT：以前稱為gate gate,，是一種 tanh激活函數，用於將輸出入資料控制在 -1 到 +1 之間。

**門控循環單元（GRU，Gated Recurrent Unit）：**

GRU是LSTM的簡化版本，結合**~~遺忘門（Forget Gate）~~**~~和輸入門~~**~~（Input Gate）~~成為一個更新門，**並將**記憶單元和隱藏狀態合併**。

GRU有兩個門：**更新門和重置門**。

更新門（Update Gate）： 決定需要**保留多少過去訊息**。

重置門（Reset Gate）： 決定應該**忘記多少過去訊息**。

優勢： GRU的計算效率比LSTM更高。

主要差異：LSTM有三個門（遺忘門、輸入門、輸出門），而GRU有兩個（重置門、更新門）。LSTM更複雜，可以捕捉更多複雜的模式，而GRU更簡單，訓練速度更快。

一張含有 螢幕擷取畫面, 圖表, Rectangle, 設計 的圖片

自動產生的描述

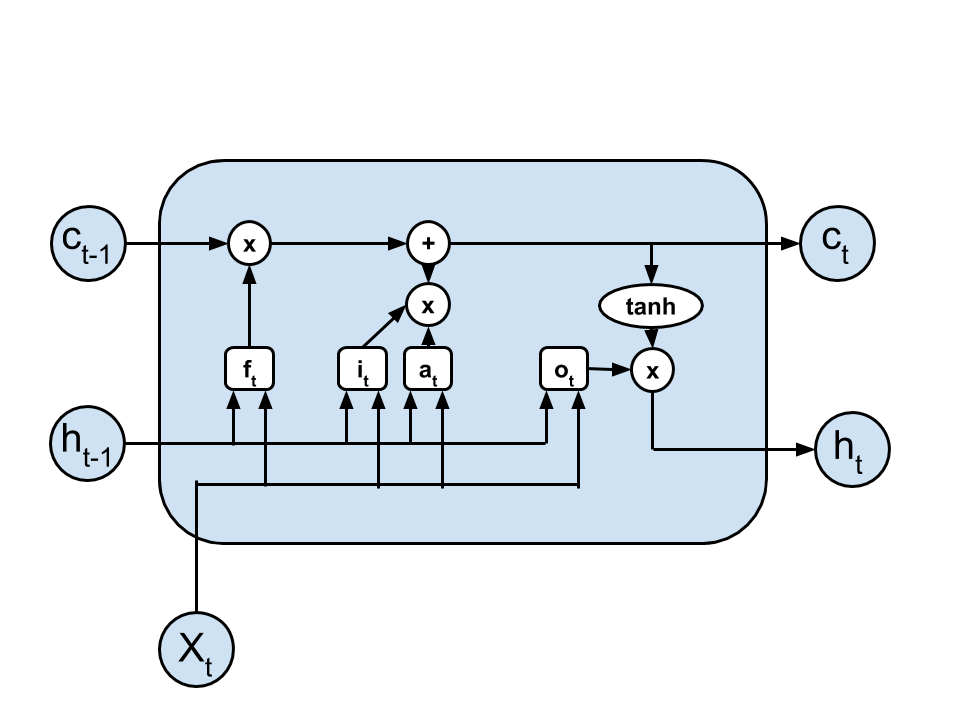


Figure 1. LSTM

1. (10%) What is the problem with a simple (or vanilla) RNN? How to solve it?

可能會導致gradient vanishing or grading exploding problems梯度消失或分級爆炸問題。使用「梯度裁剪」求解梯度爆炸問題，使用LSTM或GRU求解梯度消失問題。

梯度消失和爆炸

• 梯度爆炸問題使RNN陷入不穩定狀態

由於 RNN 的長期不收斂被導致資訊量過多的狀態。

• 當長期梯度以指數速度快速變為零時，就會出現梯度消失問題，且模型暫時無法從遙遠的事件中學習。

**總結**

**基礎RNN面臨梯度消失和梯度爆炸問題，這使得模型難以學習長期依賴。LSTM和GRU等改進模型通過引入門機制有效地解決了這些問題。此外，梯度截斷Gradient Clipping、雙向RNN** (**Bidirectional RNNs)、注意力機制(Attention Mechanisms)和transformer等技術進一步提升了模型捕捉長期依賴的能力和訓練的穩定性。**

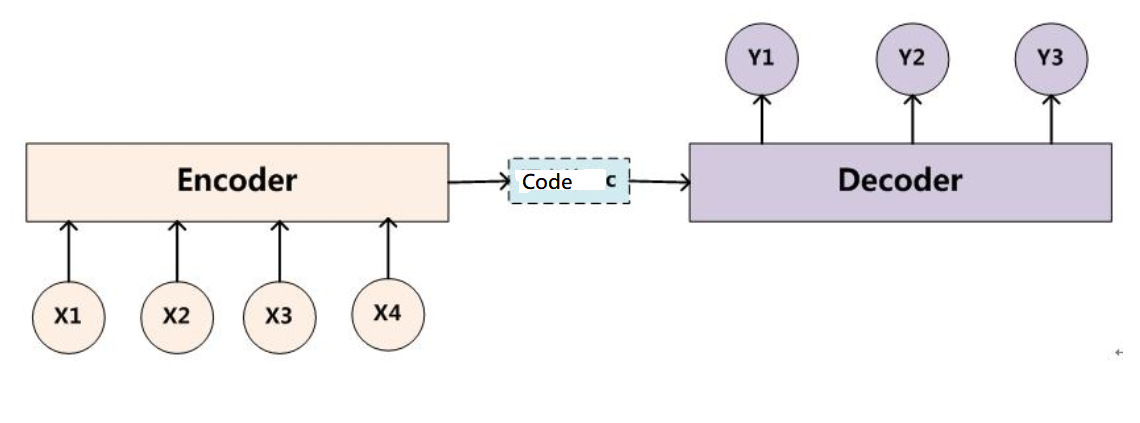
|  |
| --- |
| **梯度消失問題（Vanishing Gradient Problem）：** |
| 在基礎RNN中，反向傳播過程中會計算損失函數相對於權重的梯度，這些梯度用來更新權重，**當梯度通過多個時間步長向後傳播時，梯度會縮小。**  這種縮小會導致梯度變得非常小，最終接近於零。這就是所謂的梯度消失問題，使得模型難以學習長期依賴，因為與較早時間步長相關的權重無法有效更新。 |
| **梯度爆炸問題（Exploding Gradient Problem）：** |
| 梯度在反向傳播過程中也可能會指數級地增大。這會**導致權重更新過大，從而使網絡不穩定**。雖然梯度爆炸問題相對較少見，但它會使**訓練過程變得困難並導致模型無法收斂**。 |
| 長期依賴問題（Long-term Dependency Problem）：  **由於梯度消失問題**，**基礎RNN難以捕捉長期依賴關係**。這使得它們**在處理需要長期上下文的任務時效果不佳**。 |
| 解決方案 |
| **梯度截斷（Gradient Clipping）：**  技術：  為了**解決梯度爆炸問題，可以使用梯度截斷技術**。這種技術**涉及設置梯度的閾值，如果梯度超過這個閾值，則將其縮放到閾值範圍內**。  一張含有 文字, 傢俱, 螢幕擷取畫面, 椅子 的圖片  自動產生的描述 |
| **長短期記憶網絡（LSTM，Long Short-Term Memory）：**  **LSTM網絡設計用來避免長期依賴問題**。  **記憶單元**和**輸入門、遺忘門和輸出門**。  記憶單元：能夠長時間存儲訊息，  **遺忘門（Forget Gate）**： 決定應該**丟棄多少訊息**。  **輸入門（Input Gate）**： 決定應該**存儲多少新訊息**。  **輸出門（Output Gate）**： 決定應該**輸出多少訊息**。  優勢： 這種結構**允許LSTM有效捕捉長期依賴，並減輕梯度消失問題**。 |
| **門控循環單元（GRU，Gated Recurrent Unit）：**  結構：  GRU是LSTM的簡化版本，結合**~~遺忘門（Forget Gate）~~**~~和輸入門~~**~~（Input Gate）~~成為一個更新門，**並將**記憶單元和隱藏狀態合併**。  它包含兩個門：**更新門和重置門**。  運作：  更新門（Update Gate）： 決定需要**保留多少過去訊息**。  重置門（Reset Gate）： 決定應該**忘記多少過去訊息**。  優勢： GRU的計算效率比LSTM更高，在許多任務中表現相似。 |
| **雙向RNN（Bidirectional RNNs）：**  這些**網絡同時處理前向和後向數據，捕捉來自過去和未來的上下文訊息**。 |
| **注意力機制（Attention Mechanisms）：**  注意力機制**允許模型在預測每個輸出元素時關注輸入序列的特定部分，提高了長期依賴處理能力**。 |
| **變壓器（Transformers）：**  變壓器**完全依賴於注意力機制**，摒棄了循環結構，對於資訊的長期依賴性佳和處理速度快。 |

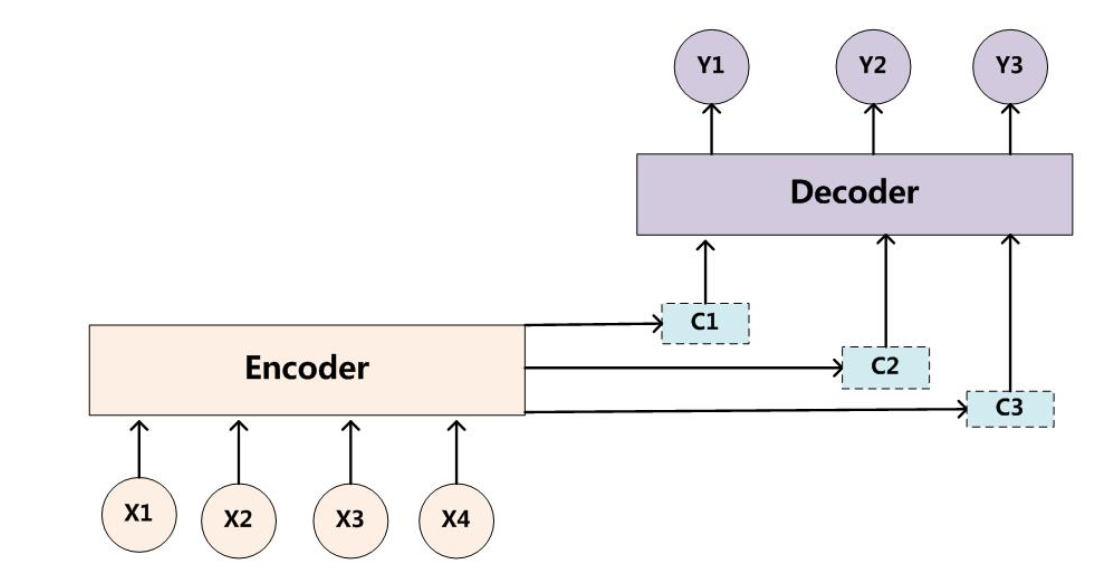
1. (10%) The following diagrams show a sequence-to-sequence translation application of an RNN with or without using attention. Point out which (Fig.2(**a)** or Fig.2(**b**) is the RNN with attention model and which is not. Explain what is the difference between them. (10%)下圖顯示了 RNN 的序列間翻譯應用程式，無論是否使用注意力。指出哪個（圖2（a）或圖2（b）是具有注意力模型的RNN，哪個不是。 解釋它們之間的區別

[Attention-Model.pdf](file:///D:\GitHub\NTUST\112-2\MI5126701%20%E4%BA%BA%E5%B7%A5%E6%99%BA%E6%85%A7%E8%88%87%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92\W12%200507\Attention-Model.pdf)

**注意力機制（**attention model**）：**

注意力機制**允許模型關注輸入序列的特定部分，讓模型得以對句子(序列)中每個字對個別字的相關性和上下文影響做判斷**。

1. Fig. 2(a)



√Fig. 2(b)

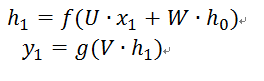
4. (20%) A simple RNN with an initial hidden state of h0=[0.0, 0.0], U=[0.5,0.6], V=[1.0,2.0], Hidden layer bias=[1.0,-1.0], Output bias=[0.1]

W = [0.1,0.2]

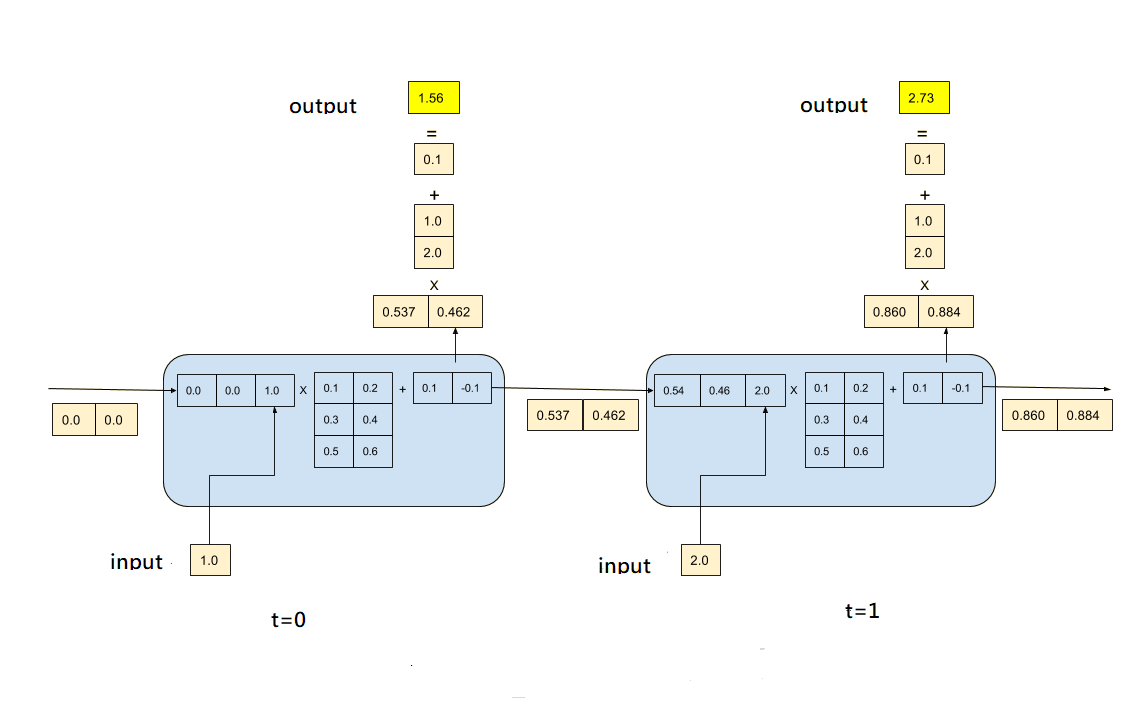
    [0.3,0.4] 會在code forwardrnn

**Given the input vector [2, 3], calculates the corresponding output of y.**

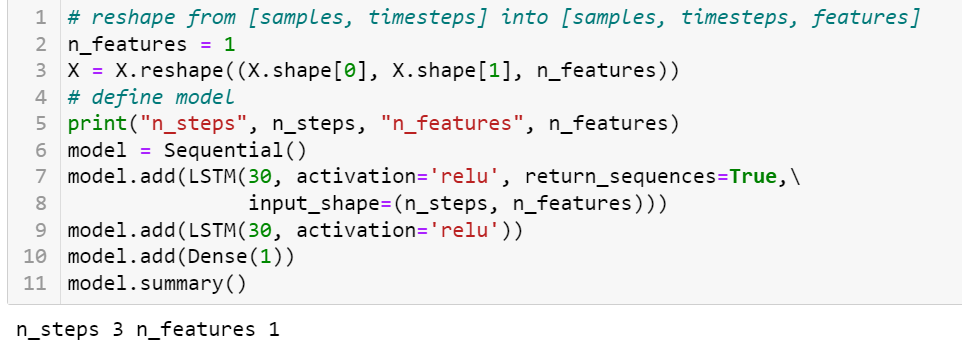
*For your reference, note that the activation function f is tanh().*

**

* 1. **將h0和input\_vector(x1)合成一個向量**
  2. **乘上W、U權重，做matrixs矩陣的乘法計算，橫乘直**
  3. **加上hidden layer bias**
  4. **Tanh，成為下一層h1的hidden layer，並同時繼續運算**
  5. **乘上V權重**
  6. **加上output bias 得到outputy1**



5. (20%)The following piece of code defines a deep LSTM model for a time series prediction. It uses three time steps of historical time series values to predict the value of the next time step. (20%)



Answer the following questions:

1. How many parameters are there in the first LSTM layer?
2. How many parameters are there in the second LSTM layer?

Hint: The input of the first layer is just a real number, and the input of the second layer is the hidden vector of the first layer, which has a dimension of 30.

Layer parameters公式

4 gates \* **units(記憶單元數量) \* 【units** **+ (layer1: n\_feature)**  +1】

**+ (layer2: iuput上層輸出參數)**

第一個 LSTM 層 (lstm\_2) parameters

4 \* 【units \* (units + **n\_feature** + 1))】 = 4 \* (30 \* (30 + 1 + 1)) = 3840

第二個 LSTM 層 (lstm\_3) parameters

4 \* 【units \* (units + **iuput\_dim** + 1))】 = 4 \* (30 \* (30 + 30 + 1)) = 7320

全連接層 (dense\_1)公式

Parameter = (input\_dim \* **neuron**) + **neurons**

**(model.add.dense(A))**

**全連階層參數量受上一層輸出units和密集曾自身神經元數影響**

Parameter : (30\*1)+1=31

總參數數量 = 3840 + 7320 + 31 = 11191

*一張含有 文字, 圖表, 螢幕擷取畫面, 字型 的圖片

自動產生的描述*

Bf 是 bias

1. (10%) Transformer is the foundation of the many recent large pre-trained language models, such as BERT and ChatGPT. The concept of self-attention is the core of the Transformer. Based on the following figure, please explain the relationships among query, key, value and the resulting representation Z. Please answer this question by considering only one-head attention. *Specifically, please answer how Z's first row is derived from v1 and v2.*

在 Transformer 模型中，自注意力機制的核心概念如下：

1. **Query (Q)**：用於搜尋相關訊息的查詢向量矩陣。
2. **Key (K)**：用於配對查詢的鍵向量矩陣。
3. **Value (V)**：包含實際訊息的值向量矩陣。
4. **表示 Z**：自注意力機制的輸出，是基於注意力得分的值的加權和。

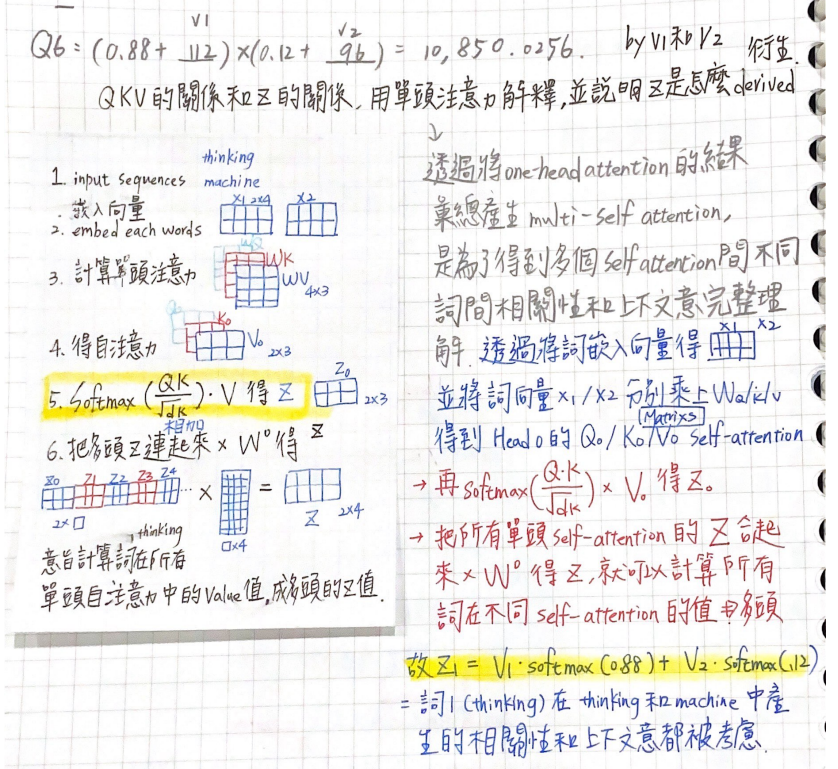
一張含有 文字, 螢幕擷取畫面, 圖表, 設計 的圖片

自動產生的描述

一張含有 Rectangle, 正方形, 行, 螢幕擷取畫面 的圖片

自動產生的描述

The answer is : Z1= 0.88 \* v1 +0.12 \* v2

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

自動產生的描述一張含有 文字, 字型, 文件, 螢幕擷取畫面 的圖片

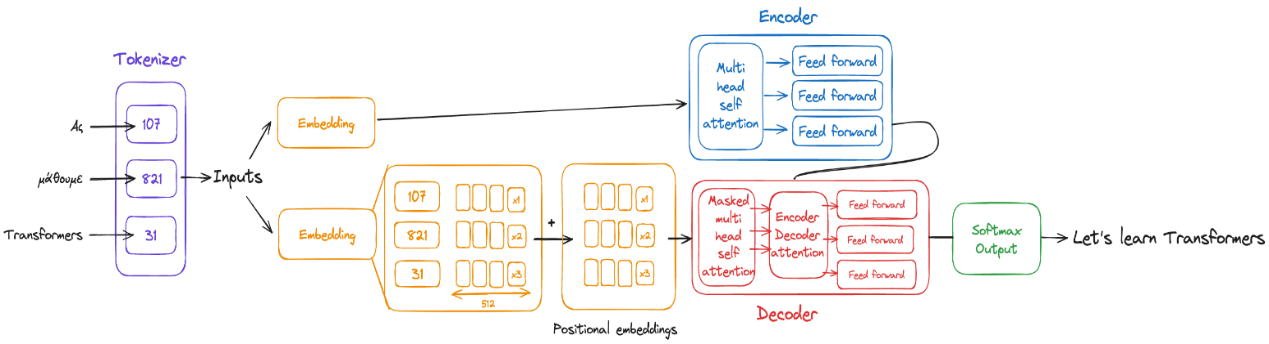
自動產生的描述一張含有 文字, 筆跡, 字型, 數字 的圖片

自動產生的描述一張含有 文字, 字型, 筆跡, 文件 的圖片

自動產生的描述一張含有 文字, 字型, 數字, 信 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述



老師可能會給我們三個字，叫我們計算hidden layer

Transformer——一種利用注意力來提高模型訓練速度的模型

<https://zjuturtle.com/2020/01/25/transformer/>

其中透過編碼器encoder和解碼器decoder來進行翻譯的動作，將輸入的句子語言進行編碼，並透過解碼器解碼成另一個語言一張含有 文字, 螢幕擷取畫面, 行, 字型 的圖片

自動產生的描述編碼器的輸入首先流經自注意力層self attentions layer，該層幫助編碼器在**對特定單字進行編碼時**查看輸入句子中的其他單字(看需要注意的前後文)。

**自注意力層的輸出被饋送到前饋神經網路FFD**。完全相同的前饋網路獨立應用於每個位置。

解碼器decoder具有這兩個層，但它們之間是一個注意力層，幫助解碼器專注於輸入句子的相關部分（類似於 seq2seq 模型中註意力的作用）。

題目也有GAN

[Information, entropy, cross entropy, KL-Divergence.pdf](file:///D:\GitHub\NTUST\112-2\MI5126701%20%E4%BA%BA%E5%B7%A5%E6%99%BA%E6%85%A7%E8%88%87%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92\W4%200311\Information,%20entropy,%20cross%20entropy,%20KL-Divergence.pdf)

<https://jianjiesun.medium.com/dl-ml%E7%AD%86%E8%A8%98-%E4%B8%89-kl-divergence-cross-entropy-4b48810e0e90>

[stable disfussion](https://blog.luckylucy.live/2023/08/28/Aigc_Stable%20Diffusion%E5%8E%9F%E7%90%86/)

一張含有 文字, 圖表, 工程製圖, 方案 的圖片

自動產生的描述

[RNN](https://arbu00.blogspot.com/2017/05/3-rnn-recurrent-neural-networks.html)

### 生成對抗網絡（簡稱GAN）

深度神經網路架構，包括兩個互相競爭的網絡：**生成器（Generator）和判別器（Discriminator）**，GAN 通過對抗訓練，來讓模型生成與真實數據分佈相似數據的神經網絡架構。它由兩個主要組件構成：

1. **生成器（G）：**

生成器接受一個隨機噪聲向量作為輸入，並生成**很像真實的假數據**。目標是生成能夠騙過判別器的假數據，使判別器無法區分這些數據是真是假。

1. **判別器（D）：**

判別器接受一個數據樣本（可以是真實數據，也可以是生成數據）作為輸入，並區分輸入數據是真是假。目標是**讓模型判斷真假數據的準確度最大化**。

GAN 的訓練過程是生成器G和判別器D兩者的對抗過程：

1. **初始化權重：** 隨機初始化G和D的權重。
2. **訓練判別器：** 使用一批真實數據和生成數據來訓練判別器，使其能夠正確區分真實數據和假數據。
3. **訓練生成器：** 只更新生成器G的權重，使生成器生成的數據能夠騙過判別器。
4. **反覆進行：** 重複上述步驟，在多個迭代中交替訓練生成器和判別器，**直到生成器能夠生成非常逼真的數據**。

* GAN 透過生成器Generator和判別器**Discriminator**的對抗訓練，學習並生成與真實數據分佈相似的數據，並在
* **生成逼真的圖像**
* **藝術風格轉換**
* **數據增強**
* **圖像修復和超分辨率**

等多個領域展現了其強大的應用潛力。

### 成本函數和最佳化

* **生成器的損失函數：**

LG=−log⁡(D(G(z)))L\_G = -\log(D(G(z)))LG​=−log(D(G(z)))

其中 DDD 是判別器，GGG 是生成器，zzz 是隨機噪聲向量。

* **判別器的損失函數：**

LD=−(log⁡(D(x))+log⁡(1−D(G(z))))L\_D = -(\log(D(x)) + \log(1 - D(G(z))))LD​=−(log(D(x))+log(1−D(G(z))))

其中 xxx 是真實數據。