# Page 1

在提供適當的歸屬後，Google特此授權僅為新聞或學術作品之用而複製本文中的表格和圖表。  
  
注意力是你所需要的一切  
  
Ashish Vaswani\*   
Google Brain   
avaswani@google.com  
  
Noam Shazeer\*   
Google Brain   
noam@google.com  
  
Niki Parmar\*   
Google Research   
nikip@google.com  
  
Jakob Uszkoreit\*   
Google Research   
usz@google.com  
  
Llion Jones\*   
Google Research   
llion@google.com  
  
Aidan N. Gomez\* †   
多倫多大學   
aidan@cs.toronto.edu  
  
Lukasz Kaiser\*   
Google Brain   
lukaszkaiser@google.com  
  
Illia Polosukhin\* †   
illia.polosukhin@gmail.com  
  
摘要  
  
主流的序列轉換模型基於複雜的循環或卷積神經網絡，這些網絡包括編碼器和解碼器。表現最好的模型還通過注意力機制將編碼器和解碼器連接起來。我們提出了一種新的簡單網絡架構，稱為Transformer，完全基於注意力機制，完全摒棄了循環和卷積。在兩個機器翻譯任務上的實驗表明，這些模型在質量上優於現有模型，同時具有更高的並行性，並且訓練時間顯著減少。我們的模型在WMT 2014英語到德語翻譯任務中達到了28.4的BLEU分數，比現有的最佳結果（包括集成模型）提高了超過2個BLEU分。在WMT 2014英語到法語翻譯任務中，我們的模型在八個GPU上訓練3.5天後，創造了新的單模型最先進的BLEU分數41.8，訓練成本僅為文獻中最佳模型的一小部分。我們通過成功應用於英語句法分析任務，無論是大規模還是有限的訓練數據，證明了Transformer在其他任務中的良好泛化能力。  
  
\*同等貢獻。列出順序是隨機的。Jakob提出用自注意力取代RNN，並開始評估這一想法的工作。Ashish與Illia一起設計並實現了第一個Transformer模型，並在這項工作的每個方面都至關重要。Noam提出了縮放點積注意力、多頭注意力和無參數位置表示，並成為每個細節方面的另一個參與者。Niki在我們的原始代碼庫和tensor2tensor中設計、實現、調整和評估了無數模型變體。Llion也嘗試了新穎的模型變體，負責我們的初始代碼庫和高效的推理和可視化。Lukasz和Aidan花費了無數天幫助我們改進模型並實現tensor2tensor，取代了我們早期的代碼庫，大大改進了結果並大幅加速了我們的研究。  
  
†在Google Brain工作期間完成的工作。  
‡在Google Research工作期間完成的工作。  
  
第31屆神經信息處理系統會議（NIPS 2017），美國加利福尼亞州長灘。

# Page 2

1 引言  
  
循環神經網絡，特別是長短期記憶（LSTM）[13] 和門控循環神經網絡 [7]，已被確立為序列建模和轉換問題（如語言建模和機器翻譯）中的最先進方法 [3, 25, 5]。自此以來，許多努力繼續推動循環語言模型和編碼器-解碼器架構的邊界 [38, 24, 15]。  
  
循環模型通常沿著輸入和輸出序列的符號位置進行計算。將位置與計算時間步驟對齊，它們生成一系列隱藏狀態 ht，作為前一隱藏狀態 ht−1 和位置 t 的輸入的函數。這種固有的順序性排除了在訓練示例中的並行化，這在序列長度較長時變得至關重要，因為內存限制限制了跨示例的批處理。最近的工作通過分解技巧 [21] 和條件計算 [32] 在計算效率上取得了顯著改進，同時在後者的情況下也提高了模型性能。然而，順序計算的基本限制仍然存在。  
  
注意力機制已成為各種任務中引人注目的序列建模和轉換模型的組成部分，允許在不考慮輸入或輸出序列中距離的情況下建模依賴關係 [2, 19]。然而，除了少數情況 [27]，這些注意力機制通常與循環網絡結合使用。  
  
在這項工作中，我們提出了Transformer，一種完全依賴注意力機制來在輸入和輸出之間繪製全局依賴關係的模型架構，摒棄了循環。Transformer允許顯著更多的並行化，並且在八個P100 GPU上訓練僅十二小時後即可達到翻譯質量的新水平。  
  
2 背景  
  
減少順序計算的目標也構成了擴展神經GPU [16]、ByteNet [18] 和 ConvS2S [9] 的基礎，這些模型都使用卷積神經網絡作為基本構建塊，並行計算所有輸入和輸出位置的隱藏表示。在這些模型中，將兩個任意輸入或輸出位置的信號關聯起來所需的操作數量隨位置之間的距離增長，對於ConvS2S是線性增長，對於ByteNet是對數增長。這使得學習遠距離位置之間的依賴關係變得更加困難 [12]。在Transformer中，這被減少到恆定數量的操作，儘管由於平均注意力加權位置的影響，導致有效分辨率降低，我們在第3.2節中描述的多頭注意力機制來對抗這一影響。  
  
自注意力，有時稱為內部注意力，是一種注意力機制，將單個序列的不同位置關聯起來，以計算序列的表示。自注意力已成功應用於各種任務，包括閱讀理解、抽象摘要、文本推理和學習任務無關的句子表示 [4, 27, 28, 22]。  
  
端到端記憶網絡基於一種循環注意力機制，而不是序列對齊的循環，並且已被證明在單語言問答和語言建模任務中表現良好 [34]。  
  
據我們所知，Transformer是第一個完全依賴自注意力來計算其輸入和輸出表示的轉換模型，而不使用序列對齊的RNN或卷積。在接下來的部分中，我們將描述Transformer，激勵自注意力並討論其相對於 [17, 18] 和 [9] 等模型的優勢。  
  
3 模型架構  
  
大多數競爭性的神經序列轉換模型具有編碼器-解碼器結構 [5, 2, 35]。在這裡，編碼器將符號表示的輸入序列 (x1, ..., xn) 映射到連續表示序列 z = (z1, ..., zn)。給定 z，解碼器然後逐個元素生成符號的輸出序列 (y1, ..., ym)。在每一步，模型是自回歸的 [10]，在生成下一個符號時消耗先前生成的符號作為額外輸入。

# Page 3

圖1：Transformer - 模型架構。  
  
Transformer 使用堆疊的自注意力機制和逐點的全連接層來構建編碼器和解碼器，分別顯示在圖1的左半部分和右半部分。  
  
3.1 編碼器和解碼器堆疊  
  
編碼器：編碼器由 \(N = 6\) 個相同的層堆疊而成。每一層有兩個子層。第一個是多頭自注意力機制，第二個是簡單的逐點全連接前饋網絡。我們在每個子層周圍使用殘差連接 [11]，然後進行層歸一化 [1]。也就是說，每個子層的輸出是 LayerNorm(x + Sublayer(x))，其中 Sublayer(x) 是子層本身實現的函數。為了促進這些殘差連接，模型中的所有子層以及嵌入層都產生維度為 \(d\_{model} = 512\) 的輸出。  
  
解碼器：解碼器也由 \(N = 6\) 個相同的層堆疊而成。除了每個編碼器層中的兩個子層外，解碼器還插入了一個第三子層，該子層對編碼器堆疊的輸出進行多頭注意力。與編碼器類似，我們在每個子層周圍使用殘差連接，然後進行層歸一化。我們還修改了解碼器堆疊中的自注意力子層，以防止位置關注到後續位置。這種掩蔽結合輸出嵌入偏移一個位置的事實，確保位置 i 的預測只能依賴於小於 i 的已知輸出。  
  
3.2 注意力  
  
注意力函數可以描述為將查詢和一組鍵值對映射到輸出，其中查詢、鍵、值和輸出都是向量。輸出計算為加權和。

# Page 4

圖2：（左）縮放點積注意力。（右）多頭注意力由多個並行運行的注意力層組成。  
  
每個值分配的權重是通過查詢與相應鍵的兼容性函數計算得出的。  
  
3.2.1 縮放點積注意力  
  
我們稱我們的特定注意力為“縮放點積注意力”（圖2）。輸入由維度為dk的查詢和鍵以及維度為dv的值組成。我們計算查詢與所有鍵的點積，將每個點積除以√dk，並應用softmax函數以獲得值的權重。  
  
在實踐中，我們同時對一組查詢計算注意力函數；將它們打包成一個矩陣Q。鍵和值也被打包成矩陣K和V。我們計算輸出矩陣為：  
  
Attention(Q, K, V) = softmax( QK^T / √dk )V (1)  
  
最常用的兩種注意力函數是加性注意力[2]和點積（乘法）注意力。點積注意力與我們的算法相同，除了縮放因子1/√dk。加性注意力使用具有單個隱藏層的前饋網絡計算兼容性函數。雖然兩者在理論上的複雜性相似，但在實踐中，點積注意力更快且更節省空間，因為它可以使用高度優化的矩陣乘法代碼實現。  
  
雖然對於小值的dk，兩種機制的表現相似，但對於較大的dk值，加性注意力在沒有縮放的情況下優於點積注意力[3]。我們懷疑對於較大的dk值，點積的幅度會變大，將softmax函數推入梯度極小的區域。為了抵消這一影響，我們將點積縮放為1/√dk。  
  
3.2.2 多頭注意力  
  
我們發現，與其使用dmodel維度的鍵、值和查詢執行單一注意力函數，不如將查詢、鍵和值線性投影h次，分別投影到dk、dk和dv維度的不同學習線性投影上。在這些投影版本的查詢、鍵和值上，我們並行執行注意力函數，產生dv維度的輸出。  
  
為了說明為什麼點積會變大，假設q和k的組件是均值為0、方差為1的獨立隨機變量。那麼它們的點積q ⋅ k = ∑i=1^dk qi ki，其均值為0，方差為dk。

# Page 5

輸出值。這些值被串聯並再次投影，產生最終值，如圖2所示。  
  
多頭注意力機制允許模型同時關注來自不同表示子空間的不同位置的信息。使用單個注意力頭時，平均會抑制這一點。  
  
$$  
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}\_1, ..., \text{head}\_h)W^O  
$$  
  
其中 $\text{head}\_i = \text{Attention}(QW\_i^Q, KW\_i^K, VW\_i^V)$  
  
投影矩陣為 $W\_i^Q \in \mathbb{R}^{d\_{model} \times d\_k}$, $W\_i^K \in \mathbb{R}^{d\_{model} \times d\_k}$, $W\_i^V \in \mathbb{R}^{d\_{model} \times d\_v}$,  
和 $W^O \in \mathbb{R}^{hd\_v \times d\_{model}}$。  
  
在這項工作中，我們使用 $h = 8$ 個並行注意力層或頭。對於每個頭，我們使用  
$d\_k = d\_v = d\_{model}/h = 64$。由於每個頭的維度減少，總計算成本與全維度的單頭注意力相似。  
  
#### 3.2.3 注意力機制在我們模型中的應用  
  
Transformer在三種不同的方式中使用多頭注意力機制：  
  
- 在“編碼器-解碼器注意力”層中，查詢來自前一個解碼器層，記憶鍵和值來自編碼器的輸出。這允許解碼器中的每個位置關注輸入序列中的所有位置。這模仿了序列到序列模型中的典型編碼器-解碼器注意力機制，如[38, 2, 9]。  
- 編碼器包含自注意力層。在自注意力層中，所有的鍵、值和查詢都來自同一個地方，在這種情況下，來自編碼器中前一層的輸出。編碼器中的每個位置可以關注編碼器前一層的所有位置。  
- 同樣，解碼器中的自注意力層允許解碼器中的每個位置關注解碼器中所有位置，直到並包括該位置。我們需要防止解碼器中的左向信息流，以保持自回歸屬性。我們在縮放點積注意力中實現這一點，通過屏蔽（設置為 $-\infty$）softmax輸入中對應於非法連接的所有值。見圖2。  
  
### 3.3 位置前饋網絡  
  
除了注意力子層，我們的編碼器和解碼器中的每一層都包含一個完全連接的前饋網絡，該網絡分別且相同地應用於每個位置。這包括兩個線性變換，中間有一個ReLU激活。  
  
$$  
\text{FFN}(x) = \max(0, xW\_1 + b\_1)W\_2 + b\_2  
$$  
  
雖然線性變換在不同位置是相同的，但它們在不同層中使用不同的參數。另一種描述方式是兩個核大小為1的卷積。輸入和輸出的維度為 $d\_{model} = 512$，內層的維度為 $d\_{ff} = 2048$。  
  
### 3.4 嵌入和Softmax  
  
與其他序列轉換模型類似，我們使用學習的嵌入將輸入標記和輸出標記轉換為維度為 $d\_{model}$ 的向量。我們還使用通常的學習線性變換和softmax函數將解碼器輸出轉換為預測的下一個標記概率。在我們的模型中，我們在兩個嵌入層和預softmax線性變換之間共享相同的權重矩陣，類似於[30]。在嵌入層中，我們將這些權重乘以 $\sqrt{d\_{model}}$。  
  
\[  
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}\_1, ..., \text{head}\_h)W^O  
\]  
  
其中 $\text{head}\_i = \text{Attention}(QW\_i^Q, KW\_i^K, VW\_i^V)$  
  
投影矩陣為 $W\_i^Q \in \mathbb{R}^{d\_{model} \times d\_k}$, $W\_i^K \in \mathbb{R}^{d\_{model} \times d\_k}$, $W\_i^V \in \mathbb{R}^{d\_{model} \times d\_v}$,  
和 $W^O \in \mathbb{R}^{hd\_v \times d\_{model}}$。  
  
在這項工作中，我們使用 $h = 8$ 個並行注意力層或頭。對於每個頭，我們使用  
$d\_k = d\_v = d\_{model}/h = 64$。由於每個頭的維度減少，總計算成本與全維度的單頭注意力相似。  
  
#### 3.2.3 注意力機制在我們模型中的應用  
  
Transformer在三種不同的方式中使用多頭注意力機制：  
  
- 在“編碼器-解碼器注意力”層中，查詢來自前一個解碼器層，記憶鍵和值來自編碼器的輸出。這允許解碼器中的每個位置關注輸入序列中的所有位置。這模仿了序列到序列模型中的典型編碼器-解碼器注意力機制，如[38, 2, 9]。  
- 編碼器包含自注意力層。在自注意力層中，所有的鍵、值和查詢都來自同一個地方，在這種情況下，來自編碼器中前一層的輸出。編碼器中的每個位置可以關注編碼器前一層的所有位置。  
- 同樣，解碼器中的自注意力層允許解碼器中的每個位置關注解碼器中所有位置，直到並包括該位置。我們需要防止解碼器中的左向信息流，以保持自回歸屬性。我們在縮放點積注意力中實現這一點，通過屏蔽（設置為 $-\infty$）softmax輸入中對應於非法連接的所有值。見圖2。  
  
### 3.3 位置前饋網絡  
  
除了注意力子層，我們的編碼器和解碼器中的每一層都包含一個完全連接的前饋網絡，該網絡分別且相同地應用於每個位置。這包括兩個線性變換，中間有一個ReLU激活。  
  
$$  
\text{FFN}(x) = \max(0, xW\_1 + b\_1)W\_2 + b\_2  
$$  
  
雖然線性變換在不同位置是相同的，但它們在不同層中使用不同的參數。另一種描述方式是兩個核大小為1的卷積。輸入和輸出的維度為 $d\_{model} = 512$，內層的維度為 $d\_{ff} = 2048$。  
  
### 3.4 嵌入和Softmax  
  
與其他序列轉換模型類似，我們使用學習的嵌入將輸入標記和輸出標記轉換為維度為 $d\_{model}$ 的向量。我們還使用通常的學習線性變換和softmax函數將解碼器輸出轉換為預測的下一個標記概率。在我們的模型中，我們在兩個嵌入層和預softmax線性變換之間共享相同的權重矩陣，類似於[30]。在嵌入層中，我們將這些權重乘以 $\sqrt{d\_{model}}$。

# Page 6

表 1：不同層類型的最大路徑長度、每層複雜度和最少順序操作數。n 是序列長度，d 是表示維度，k 是卷積的核大小，r 是受限自注意力中的鄰域大小。  
  
| 層類型 | 每層複雜度 | 順序操作數 | 最大路徑長度 |  
|-----------------------------|----------------------|-----------------------|---------------------|  
| 自注意力 | O(n² · d) | O(1) | O(1) |  
| 循環層 | O(n · d²) | O(n) | O(n) |  
| 卷積層 | O(k · n · d²) | O(1) | O(logₖ(n)) |  
| 受限自注意力 | O(r · n · d) | O(1) | O(n/r) |  
  
3.5 位置編碼  
  
由於我們的模型不包含循環和卷積，為了使模型利用序列的順序，我們必須注入一些關於序列中標記相對或絕對位置的信息。為此，我們在編碼器和解碼器堆疊的底部將“位置編碼”添加到輸入嵌入中。位置編碼與嵌入具有相同的維度 d\_model，以便兩者可以相加。有許多位置編碼的選擇，包括學習的和固定的[9]。  
  
在這項工作中，我們使用不同頻率的正弦和餘弦函數：  
  
\[ PE\_{(pos, 2i)} = \sin(pos / 10000^{2i/d\_{model}}) \]  
\[ PE\_{(pos, 2i+1)} = \cos(pos / 10000^{2i/d\_{model}}) \]  
  
其中 pos 是位置，i 是維度。也就是說，位置編碼的每個維度對應一個正弦波。波長形成從 2π 到 10000·2π 的幾何級數。我們選擇這個函數是因為我們假設它可以讓模型容易地學會通過相對位置進行注意，因為對於任何固定的偏移量 k，\( PE\_{(pos+k)} \) 可以表示為 \( PE\_{pos} \) 的線性函數。  
  
我們還嘗試使用學習的位置嵌入[9]，發現兩個版本產生的結果幾乎相同（見表 3 行 (E)）。我們選擇正弦版本是因為它可能允許模型外推到比訓練期間遇到的序列長度更長的序列。  
  
4 為什麼選擇自注意力  
  
在本節中，我們將自注意力層與常用於將一個變長序列的符號表示 \((x\_1, ..., x\_n)\) 映射到另一個相同長度序列 \((z\_1, ..., z\_n)\) 的循環層和卷積層進行比較，其中 \( x\_i, z\_i \in \mathbb{R}^d \)，例如典型序列轉導編碼器或解碼器中的隱藏層。為了激勵我們使用自注意力，我們考慮三個需求。  
  
一是每層的總計算複雜度。另一個是可以並行化的計算量，通過所需的最少順序操作數來衡量。  
  
第三是網絡中長距依賴關係之間的路徑長度。學習長距依賴關係是許多序列轉導任務中的一個關鍵挑戰。影響學習這些依賴關係能力的一個關鍵因素是網絡中前向和後向信號必須穿越的路徑長度。輸入和輸出序列中任何位置組合之間的這些路徑越短，學習長距依賴關係就越容易[12]。因此，我們還比較了由不同層類型組成的網絡中任何兩個輸入和輸出位置之間的最大路徑長度。  
  
如表 1 所示，自注意力層通過恆定數量的順序操作連接所有位置，而循環層需要 \( O(n) \) 順序操作。在計算複雜度方面，自注意力層比循環層更快，當序列

# Page 7

長度 n 小於表示維度 d，這在使用最先進模型進行機器翻譯的句子表示中是最常見的情況，例如 word-piece [38] 和 byte-pair [31] 表示。為了提高處理非常長序列任務的計算性能，可以將自注意力限制在僅考慮輸入序列中以相應輸出位置為中心的大小為 r 的鄰域。這將使最大路徑長度增加到 O(n/r)。我們計劃在未來的工作中進一步研究這種方法。  
  
單個卷積層的核寬度 k < n 無法連接所有輸入和輸出位置對。在連續核的情況下，這需要堆疊 O(n/k) 個卷積層，或者在膨脹卷積的情況下需要 O(logk(n)) 個卷積層 [18]，從而增加了網絡中任意兩個位置之間最長路徑的長度。卷積層通常比循環層更昂貴，成本高出 k 倍。然而，可分離卷積 [6] 大大降低了複雜性，達到 O(k · n · d + n · dk)。即使 k = n，可分離卷積的複雜性也等同於自注意力層和逐點前饋層的組合，這是我們在模型中採用的方法。  
  
作為附帶好處，自注意力可以產生更易解釋的模型。我們檢查了模型的注意力分佈，並在附錄中展示和討論了示例。不僅個別注意力頭明顯學會了執行不同的任務，許多還表現出與句子的句法和語義結構相關的行為。  
  
5 訓練  
  
本節描述了我們模型的訓練方案。  
  
5.1 訓練數據和批處理  
  
我們在標準的 WMT 2014 英德數據集上進行了訓練，該數據集包含約 450 萬對句子。句子使用 byte-pair 編碼 [31]，其共享的源-目標詞彙表約為 37000 個標記。對於英法語，我們使用了顯著更大的 WMT 2014 英法數據集，該數據集包含 3600 萬句子，並將標記分為 32000 個 word-piece 詞彙表 [38]。句子對按大致序列長度分批。每個訓練批次包含一組句子對，約包含 25000 個源標記和 25000 個目標標記。  
  
5.2 硬件和時間表  
  
我們在一台配備 8 個 NVIDIA P100 GPU 的機器上訓練了我們的模型。對於使用本文描述的超參數的基礎模型，每個訓練步驟大約需要 0.4 秒。我們訓練基礎模型共 100,000 步或 12 小時。對於我們的大模型（表 3 底部描述），每步時間為 1.0 秒。大模型訓練了 300,000 步（3.5 天）。  
  
5.3 優化器  
  
我們使用了 Adam 優化器 [20]，其參數為 β1 = 0.9，β2 = 0.98 和 ϵ = 10−9。我們在訓練過程中根據以下公式變化學習率：  
  
$$  
lrate = d\_{model}^{-0.5} \cdot \min(step\\_num^{-0.5}, step\\_num \cdot warmup\\_steps^{-1.5})  
$$  
  
(3)  
  
這對應於在前 warmup\_steps 訓練步驟中線性增加學習率，之後按步數的平方根倒數成比例地減少學習率。我們使用了 warmup\_steps = 4000。  
  
5.4 正則化  
  
我們在訓練過程中使用了三種類型的正則化：

# Page 8

表2：Transformer在英語到德語和英語到法語的newstest2014測試中，以較低的訓練成本實現了比先前最先進模型更好的BLEU分數。  
  
| 模型 | BLEU | 訓練成本 (FLOPs) |  
|-------------------------------|---------------|-------------------|  
| | EN-DE | EN-FR | EN-DE | EN-FR |  
| ByteNet [18] | 23.75 | | 1.0 \* 10^20 |  
| Deep-Att + PosUnk [39] | | 39.2 | 1.0 \* 10^20 |  
| GNMT + RL [38] | 24.6 | 39.92 | 2.3 \* 10^19 | 1.4 \* 10^20 |  
| ConvS2S [9] | 25.16 | 40.46 | 9.6 \* 10^18 | 1.5 \* 10^19 |  
| MoE [23] | 26.03 | 40.56 | 2.0 \* 10^19 | 1.2 \* 10^20 |  
| Deep-Att + PosUnk Ensemble [39]| | 40.4 | 8.0 \* 10^20 |  
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | 1.8 \* 10^19 | 1.1 \* 10^21 |  
| ConvS2S Ensemble [9] | 26.36 | 41.29 | 7.7 \* 10^19 | 1.2 \* 10^21 |  
| Transformer (base model) | 27.3 | 38.1 | 3.3 \* 10^18 |  
| Transformer (big) | 28.4 | 41.8 | 2.3 \* 10^19 |  
  
殘差Dropout 我們對每個子層的輸出應用dropout [33]，在其被添加到子層輸入並進行歸一化之前。此外，我們在編碼器和解碼器堆棧中的嵌入和位置編碼的總和上應用dropout。對於基礎模型，我們使用的dropout率為Pdrop = 0.1。  
  
標籤平滑 在訓練過程中，我們使用了值為εls = 0.1的標籤平滑[36]。這會影響困惑度，因為模型學會了更加不確定，但提高了準確性和BLEU分數。  
  
6 結果  
  
6.1 機器翻譯  
  
在WMT 2014英語到德語的翻譯任務中，大型Transformer模型（表2中的Transformer (big)）比先前報告的最佳模型（包括集成模型）高出超過2.0 BLEU，創造了新的最先進BLEU分數28.4。該模型的配置列在表3的底部。訓練在8個P100 GPU上花費了3.5天。即使是我們的基礎模型也超過了所有先前發表的模型和集成模型，且訓練成本僅為任何競爭模型的一小部分。  
  
在WMT 2014英語到法語的翻譯任務中，我們的大型模型實現了41.0的BLEU分數，超過了所有先前發表的單一模型，訓練成本不到先前最先進模型的1/4。英語到法語的Transformer (big)模型使用的dropout率為Pdrop = 0.1，而不是0.3。  
  
對於基礎模型，我們使用了通過平均最後5個檢查點獲得的單一模型，這些檢查點每10分鐘寫一次。對於大型模型，我們平均了最後20個檢查點。我們使用了束搜索，束大小為4，長度懲罰α = 0.6 [38]。這些超參數是在開發集上實驗後選擇的。我們在推理過程中將最大輸出長度設置為輸入長度+50，但在可能的情況下提前終止[38]。  
  
表2總結了我們的結果，並將我們的翻譯質量和訓練成本與文獻中的其他模型架構進行了比較。我們通過將訓練時間、使用的GPU數量和每個GPU的持續單精度浮點容量估計值相乘來估算訓練模型所用的浮點運算次數。  
  
6.2 模型變體  
  
為了評估Transformer不同組件的重要性，我們以不同方式變化了我們的基礎模型，測量了英語到德語翻譯性能的變化。  
  
5 我們使用了2.8、3.7、6.0和9.5 TFLOPS的K80、K40、M40和P100的值。

# Page 9

表3：Transformer架構的變體。未列出的值與基礎模型相同。所有指標均基於英德翻譯開發集newstest2013。列出的困惑度是根據我們的字節對編碼按每個詞片計算的，不應與每個詞的困惑度進行比較。  
  
\[  
\begin{array}{cccccccccccc}  
N & d\_{\text{model}} & d\_{\text{ff}} & h & d\_k & d\_v & P\_{\text{drop}} & c\_{\text{ls}} & \text{訓練步驟} & \text{PPL (dev)} & \text{BLEU (dev)} & \text{參數} \times 10^6 \\  
\hline  
\text{基礎} & 6 & 512 & 2048 & 8 & 64 & 64 & 0.1 & 0.1 & 100K & 4.92 & 25.8 & 65 \\  
\hline  
(A) & 1 & 512 & 512 & 5.29 & 24.9 & \\  
& 4 & 128 & 128 & 5.00 & 25.5 & \\  
& 16 & 32 & 32 & 4.91 & 25.8 & \\  
& 32 & 16 & 16 & 5.01 & 25.4 & \\  
\hline  
(B) & 16 & 5.16 & 25.1 & 58 \\  
& 32 & 5.01 & 25.4 & 60 \\  
\hline  
(C) & 2 & 6.11 & 23.7 & 36 \\  
& 4 & 5.19 & 25.3 & 50 \\  
& 8 & 4.88 & 25.5 & 80 \\  
\hline  
& 256 & 32 & 32 & 5.85 & 24.5 & 28 \\  
& 1024 & 128 & 128 & 4.66 & 26.0 & 48 \\  
\hline  
& 1024 & 5.12 & 25.4 & 53 \\  
& 4096 & 4.75 & 26.2 & 90 \\  
\hline  
(D) & 0.0 & 5.77 & 24.6 & \\  
& 0.2 & 4.95 & 25.5 & \\  
& 0.4 & 4.67 & 25.3 & \\  
& 0.6 & 5.47 & 25.7 & \\  
\hline  
(E) & \text{位置嵌入代替正弦函數} & 4.92 & 25.7 & \\  
\hline  
\text{大} & 6 & 1024 & 4096 & 16 & 0.3 & 300K & 4.33 & 26.4 & 213 \\  
\end{array}  
\]  
  
開發集newstest2013。我們使用了上一節描述的束搜索，但沒有進行檢查點平均。我們在表3中展示了這些結果。  
  
在表3的(A)行中，我們改變了注意力頭的數量以及注意力鍵和值的維度，保持計算量不變，如第3.2.2節所述。雖然單頭注意力比最佳設置低0.9 BLEU，但頭數過多時質量也會下降。  
  
在表3的(B)行中，我們觀察到減少注意力鍵的大小 \(d\_k\) 會損害模型質量。這表明確定兼容性並不容易，可能需要比點積更複雜的兼容性函數。我們在(C)和(D)行中進一步觀察到，正如預期的那樣，更大的模型效果更好，並且丟棄法在避免過擬合方面非常有幫助。在(E)行中，我們用學習的位置嵌入代替了正弦位置編碼[9]，並觀察到與基礎模型幾乎相同的結果。  
  
6.3 英語句法分析  
  
為了評估Transformer是否能夠推廣到其他任務，我們在英語句法分析上進行了實驗。這項任務提出了特定的挑戰：輸出受到強烈的結構約束，並且顯著長於輸入。此外，RNN序列到序列模型在小數據環境中未能達到最先進的結果[37]。  
  
我們在Penn Treebank的Wall Street Journal (WSJ)部分上訓練了一個4層的Transformer，\(d\_{\text{model}} = 1024\)，大約有40K的訓練句子。我們還在半監督設置中進行了訓練，使用了來自高置信度和BerkleyParser語料庫的大約1M句子[37]。我們在僅WSJ設置中使用了16K詞彙量，在半監督設置中使用了32K詞彙量。  
  
我們只進行了少量實驗來選擇丟棄率、注意力和殘差（第5.4節）、學習率和束大小在第22節開發集上，但所有其他參數均保持不變，與英德基礎翻譯模型相同。在推理過程中，我們

# Page 10

表 4：Transformer 在英語句法分析中的泛化能力（結果基於 WSJ 第 23 節）  
  
| 解析器 | 訓練 | WSJ 23 F1 |  
|---------------------------------------|-------------------------|-----------|  
| Vinyals & Kaiser 等人 (2014) [37] | 僅 WSJ，判別式 | 88.3 |  
| Petrov 等人 (2006) [29] | 僅 WSJ，判別式 | 90.4 |  
| Zhu 等人 (2013) [40] | 僅 WSJ，判別式 | 90.4 |  
| Dyer 等人 (2016) [8] | 僅 WSJ，判別式 | 91.7 |  
| Transformer (4 層) | 僅 WSJ，判別式 | 91.3 |  
| Zhu 等人 (2013) [40] | 半監督式 | 91.3 |  
| Huang & Harper (2009) [14] | 半監督式 | 91.3 |  
| McClosky 等人 (2006) [26] | 半監督式 | 92.1 |  
| Vinyals & Kaiser 等人 (2014) [37] | 半監督式 | 92.1 |  
| Transformer (4 層) | 半監督式 | 92.7 |  
| Luong 等人 (2015) [23] | 多任務 | 93.0 |  
| Dyer 等人 (2016) [8] | 生成式 | 93.3 |  
  
我們將最大輸出長度增加到輸入長度 + 300。我們在僅 WSJ 和半監督式設置中都使用了 21 的 beam size 和 α = 0.3。  
  
表 4 中的結果顯示，儘管缺乏針對特定任務的調整，我們的模型表現出乎意料地好，除了 Recurrent Neural Network Grammar [8] 之外，所有先前報告的模型都被超越。  
  
與 RNN 序列到序列模型 [37] 相比，即使僅在 WSJ 訓練集的 40K 句子上訓練，Transformer 也超越了 Berkeley-Parser [29]。  
  
7 結論  
  
在這項工作中，我們介紹了 Transformer，這是第一個完全基於注意力的序列轉換模型，用多頭自注意力取代了編碼器-解碼器架構中最常用的循環層。  
  
對於翻譯任務，Transformer 的訓練速度顯著快於基於循環或卷積層的架構。在 WMT 2014 英語到德語和 WMT 2014 英語到法語的翻譯任務中，我們達到了新的最先進水平。在前一個任務中，我們的最佳模型超越了所有先前報告的集成模型。  
  
我們對基於注意力的模型的未來感到興奮，並計劃將它們應用於其他任務。我們計劃將 Transformer 擴展到涉及文本以外的輸入和輸出模態的問題，並研究局部、受限的注意力機制，以有效處理如圖像、音頻和視頻等大型輸入和輸出。使生成過程不那麼順序化是我們的另一個研究目標。  
  
我們用來訓練和評估模型的代碼可在 https://github.com/tensorflow/tensor2tensor 獲得。  
  
致謝 我們感謝 Nal Kalchbrenner 和 Stephan Gouws 的寶貴意見、修正和啟發。  
  
參考文獻  
  
[1] Jimmy Lei Ba, Jamie Ryan Kiros, 和 Geoffrey E Hinton. 層歸一化. arXiv preprint arXiv:1607.06450, 2016.  
  
[2] Dzmitry Bahdanau, Kyunghyun Cho, 和 Yoshua Bengio. 通過聯合學習對齊和翻譯的神經機器翻譯. CoRR, abs/1409.0473, 2014.  
  
[3] Denny Britz, Anna Goldie, Minh-Thang Luong, 和 Quoc V. Le. 神經機器翻譯架構的大規模探索. CoRR, abs/1703.03906, 2017.  
  
[4] Jianpeng Cheng, Li Dong, 和 Mirella Lapata. 用於機器閱讀的長短期記憶網絡. arXiv preprint arXiv:1601.06733, 2016.

# Page 11

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, 和 Yoshua Bengio. 使用RNN編碼器-解碼器進行統計機器翻譯的短語表示學習。CoRR, abs/1406.1078, 2014年。  
  
[6] Francois Chollet. Xception: 使用深度可分離卷積的深度學習。arXiv預印本 arXiv:1610.02357, 2016年。  
  
[7] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, 和 Yoshua Bengio. 閘控循環神經網絡在序列建模中的實證評估。CoRR, abs/1412.3555, 2014年。  
  
[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, 和 Noah A. Smith. 循環神經網絡語法。在NAACL會議論文集, 2016年。  
  
[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, 和 Yann N. Dauphin. 卷積序列到序列學習。arXiv預印本 arXiv:1705.03122v2, 2017年。  
  
[10] Alex Graves. 使用循環神經網絡生成序列。arXiv預印本 arXiv:1308.0850, 2013年。  
  
[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, 和 Jian Sun. 用於圖像識別的深度殘差學習。在IEEE計算機視覺與模式識別會議論文集, 頁770–778, 2016年。  
  
[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, 和 Jürgen Schmidhuber. 循環網絡中的梯度流動：學習長期依賴的困難, 2001年。  
  
[13] Sepp Hochreiter 和 Jürgen Schmidhuber. 長短期記憶。神經計算, 9(8):1735–1780, 1997年。  
  
[14] 黃中強 和 Mary Harper. 使用潛在註釋跨語言自訓練PCFG語法。在2009年自然語言處理實證方法會議論文集, 頁832–841。ACL, 2009年8月。  
  
[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, 和 Yonghui Wu. 探索語言建模的極限。arXiv預印本 arXiv:1602.02410, 2016年。  
  
[16] Lukasz Kaiser 和 Samy Bengio. 主動記憶能否取代注意力？在神經信息處理系統進展 (NIPS), 2016年。  
  
[17] Lukasz Kaiser 和 Ilya Sutskever. 神經GPU學習算法。在國際學習表徵會議 (ICLR), 2016年。  
  
[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, 和 Koray Kavukcuoglu. 線性時間的神經機器翻譯。arXiv預印本 arXiv:1610.10099v2, 2017年。  
  
[19] Yoon Kim, Carl Denton, Luong Hoang, 和 Alexander M. Rush. 結構化注意網絡。在國際學習表徵會議, 2017年。  
  
[20] Diederik Kingma 和 Jimmy Ba. Adam: 一種隨機優化方法。在ICLR, 2015年。  
  
[21] Oleksii Kuchaiev 和 Boris Ginsburg. LSTM網絡的因式分解技巧。arXiv預印本 arXiv:1703.10722, 2017年。  
  
[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, 和 Yoshua Bengio. 結構化自注意句子嵌入。arXiv預印本 arXiv:1703.03130, 2017年。  
  
[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, 和 Lukasz Kaiser. 多任務序列到序列學習。arXiv預印本 arXiv:1511.06114, 2015年。  
  
[24] Minh-Thang Luong, Hieu Pham, 和 Christopher D Manning. 基於注意力的神經機器翻譯的有效方法。arXiv預印本 arXiv:1508.04025, 2015年。

# Page 12

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, 和 Beatrice Santorini. 建立一個大型註釋的英語語料庫：賓州樹庫。計算語言學, 19(2):313–330, 1993年。  
  
[26] David McClosky, Eugene Charniak, 和 Mark Johnson. 有效的自我訓練解析方法。在北美計算語言學協會人類語言技術會議主會議論文集, 頁152–159。ACL, 2006年6月。  
  
[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, 和 Jakob Uszkoreit. 一個可分解的注意力模型。在自然語言處理中的經驗方法, 2016年。  
  
[28] Romain Paulus, Caiming Xiong, 和 Richard Socher. 用於抽象總結的深度強化模型。arXiv預印本 arXiv:1705.04304, 2017年。  
  
[29] Slav Petrov, Leon Barrett, Romain Thibaux, 和 Dan Klein. 學習準確、緊湊且可解釋的樹註釋。在第21屆國際計算語言學會議和第44屆ACL年會論文集, 頁433–440。ACL, 2006年7月。  
  
[30] Ofir Press 和 Lior Wolf. 使用輸出嵌入來改進語言模型。arXiv預印本 arXiv:1608.05859, 2016年。  
  
[31] Rico Sennrich, Barry Haddow, 和 Alexandra Birch. 使用子詞單元進行罕見詞的神經機器翻譯。arXiv預印本 arXiv:1508.07909, 2015年。  
  
[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, 和 Jeff Dean. 極大規模的神經網絡：稀疏門控專家混合層。arXiv預印本 arXiv:1701.06538, 2017年。  
  
[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, 和 Ruslan Salakhutdinov. Dropout：防止神經網絡過擬合的一種簡單方法。機器學習研究期刊, 15(1):1929–1958, 2014年。  
  
[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, 和 Rob Fergus. 端到端記憶網絡。在C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, 和 R. Garnett編輯的《神經信息處理系統進展28》中, 頁2440–2448。Curran Associates, Inc., 2015年。  
  
[35] Ilya Sutskever, Oriol Vinyals, 和 Quoc V V Le. 使用神經網絡進行序列到序列學習。在《神經信息處理系統進展》中, 頁3104–3112, 2014年。  
  
[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, 和 Zbigniew Wojna. 重新思考計算機視覺的Inception架構。CoRR, abs/1512.00567, 2015年。  
  
[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, 和 Hinton. 將語法視為外語。在《神經信息處理系統進展》中, 2015年。  
  
[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, 等。谷歌的神經機器翻譯系統：彌合人類和機器翻譯之間的差距。arXiv預印本 arXiv:1609.08144, 2016年。  
  
[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, 和 Wei Xu. 具有快速前向連接的深度循環模型用於神經機器翻譯。CoRR, abs/1606.04199, 2016年。  
  
[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, 和 Jingbo Zhu. 快速且準確的移位-歸約成分解析。在第51屆ACL年會（長篇論文卷1）論文集, 頁434–443。ACL, 2013年8月。

# Page 13

注意力可視化  
  
圖3：在第6層中的第5層編碼器自注意力機制中，關注長距離依賴關係的示例。許多注意力頭關注動詞「making」的遠距依賴，完成短語「making...more difficult」。此處僅顯示單詞「making」的注意力。不同顏色代表不同的注意力頭。最佳觀看效果為彩色。

# Page 14

法律永遠不會完美，但這個應用應該是公正的，這正是我們所缺少的，這是我的看法。  
<EOS>  
<pad>  
<pad>  
<pad>  
  
法律永遠不會完美，但這個應用應該是公正的，這正是我們所缺少的，這是我的看法。  
<EOS>  
<pad>  
<pad>  
<pad>  
  
圖4：兩個注意力頭，也在第6層的第5層，顯然參與了解決照應問題。上圖：頭5的完整注意力。下圖：僅從單詞“其”的注意力頭5和6的隔離注意力。注意，這個單詞的注意力非常集中。  
  
14

# Page 15

法律永遠不會完美，但在我看來，這個應用程式應該正是我們所缺少的。  
  
法律永遠不會完美，但在我看來，這個應用程式應該正是我們所缺少的。  
  
圖5：許多注意力頭表現出與句子結構相關的行為。我們在上面給出了兩個這樣的例子，來自第5層中的兩個不同的編碼器自注意力頭。這些頭顯然學會了執行不同的任務。  
  
15