**节点是什么？**

# “节点”是什么

**节点这个词，都不陌生，互联网世界里，节点的概念一直存在，今天就来讲下什么是节点。**

**节点的定义**

**计算机网络节点：在数据通信中，一个物理网络节点可以是数据电路端接设备（DCE），如调制解调器、集线器、桥接器或交换机；也可以是一个数据终端设备（DTE），如数字手机，打印机或主机（例如路由器、工作站或服务器）。**

**指的是区块链网络中的计算机，包括手机、矿机、台式机和服务器等等。操作一个节点的人可以是普通的钱包用户、矿工和多个人协作。**

**例如比特币网络，是属于共有链，当我们在自己的已连接到互联网的电脑上运行bitcoind程序时，该电脑也就成为比特币网络中的一个节点。**

**对于像比特币这样的公有链，理论上来讲，你下载完整的区块链，参与交易和挖矿，才算是节点。然而，在现在的比特币里，矿工，完全节点，轻量节点，甚至普通用户，在不同的语境下都可能被称为节点。**

**对于公有链而言，是你想从链上获益就必须按照规则成为节点，而不是相反。而私有链和联盟链则反过来——你先成为节点，然后按照规则行事。**

**区块链节点的特点**

**一、具有一定的存储空间存储空间指电子存储空间，包括日常的TF卡、U盘、移动硬盘和计算机等。**

**二、连接网络，需要有存储空间的设备通过网线连接网络。**

**三、可视化操作终端手机、iPad、电脑是目前主流的可视化操作终端。**

**四、参与区块链要在连接网络的存储空间运行区块链相应程序，通过可视化惭怍终端进行交易。**

**区块链节点的前沿**

**国际上虽然有行业在应用区块链技术，但都称不上是真正的区块链，只能说是单纯运用了区块链的底层技术和原理。真正意义上的区块链应该是由大量个人或者家庭用户参与的区块链，每个个人或者家庭都是区块链的节点。目前主流的节点主要指计算机和矿机，对于大多数用户来讲电力和宽带成本很高，并不能适用于大部分节点。**

# 一个公式搞定深度学习模型中的参数数量计算

**我们知道随着神经网络层次的不断加深，参数数量也越来越多。但一个神经网络模型的参数量具体是多少可能我们并不是很清楚，所以今天我们来分享每一种网络模型的参数数量计算公式。**

**FFNN**

**首先我们定义三个参数：**

**i:输入大小**

**h:隐藏层大小**

**o:输出大小**

**即在前馈神经网络中参数数量为：**

**num\_params =各层之间参数+每层的偏差= (i×h + h×o)+(h + o)**

**例如输入大小3，隐藏层大小5，输出大小2：**

**则num\_params = （3×5 + 5×2）+（5 + 2） = 32**

**RNN**

**首先我们定义三个参数：**

**g:门的数量(RNN有1个门，GRU有3个，LSTM有4个)**

**h:隐藏单元大小**

**i:输出大小**

**每个门中的权重实际上是一个输入大小(h + i)和输出大小为h的FFNN。所以每个门都有h（h + i）+ h个参数。即在RNN中参数数量为：**

**num\_params= g ×[ h（h + i）+ h ]**

**例如具有2个隐藏单元和输入尺寸3的LSTM：**

**则num\_params = g ×[ h（h + i）+ h ]= 4 ×[2（2 + 3）+ 2] = 48**

**具有5个隐藏单元和输入大小为8（其输出连接在一起）的堆叠双向GRU +具有50个隐藏单元的LSTM的参数数量为：**

**其中双向GRU，5个隐藏单元，输入大小10**

**num\_params\_layer1 = 2 × g ×[ h(h + i)+ h ] = 2 ×3×[5(5 + 8)+ 5] = 420**

**LSTM有50个隐藏单位**

**num\_params\_layer2 = g ×[ h(h + i)+ h ]= 4×[50(50 + 10)+ 50]= 12200**

**则total\_params = 420 + 12200 = 12620**

**CNN**

**首先我们定义三个参数：**

**i:输入尺寸**

**f:卷积核的大小**

**o:输出大小**

**则每个滤波器对应的输出映射参数为**

**num\_params =权重+偏差= [ i×(f×f)×o ] + o**

**例如带有1 × 1滤波器的灰度图像，输出3个通道**

**参数数量为**

**num\_params = [ i×(f×f)×o ] + o= [1 ×(2 × 2)× 3] + 3= 15**

**例如具有2×2滤波器的RGB图像，1通道的输出。则每个输入像素通道都有1个滤波器。产生的卷积按元素叠加，并且向每个元素添加偏差项。**

**最终得到参数数量为**

**num\_params = [ i×(f×f)×o ] + o= [3 ×(2×2)×1] + 1 = 13**

**原文：**[**https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889**](https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889)

# 手动计算深度学习模型中的参数数量

## FFNN、RNN 和 CNN 模型中计算参数的 5 个简单示例

雷米·卡里姆

[雷米·卡里姆](https://medium.com/@remykarem?source=post_page-----8f1716241889--------------------------------)

宽为什么我们需要再次计算深度学习模型中的参数数量？我们没有。但在我们需要减少模型文件大小甚至减少模型推理时间的情况下，了解模型量化前后的参数数量会派上用场。（有关深度学习的有效方法和硬件，请参见[此处的](https://www.youtube.com/watch?v=eZdOkDtYMoo)视频。）

计数的数量*可训练*p一个深度学习模型rameters被认为是太微不足道了，因为你的代码已经可以为你做这个。但我想把我的笔记留在这里供我们偶尔参考。以下是我们将运行的模型：

1. [前馈神经网络 (FFNN)](https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889#9fe4)
2. [循环神经网络 (RNN)](https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889#192e)
3. [卷积神经网络 (CNN)](https://towardsdatascience.com/counting-no-of-parameters-in-deep-learning-models-by-hand-8f1716241889#5137)

同时，我将使用来自 Keras 的 API 构建模型，以方便原型设计和干净的代码，因此让我们在此处快速导入相关对象：

**from** keras.layers **import** Input, Dense, SimpleRNN, LSTM, GRU, Conv2D   
from keras.layers **import** Bidirectional   
from keras.models **import** Model

构建之后model，调用model.count\_params()以验证有多少参数是可训练的。

## 1. FFNN

* ***i*** , 输入大小
* ***h*** , 隐藏层的大小
* ***o*** , 输出大小

对于一个隐藏层，

*num\_params*  
*=*层之间的连接 + 每层的偏差  
= **( *i×h + h×o* ) *+* ( *h+o* )**

**例 1.1：输入大小 3，隐藏层大小 5，输出大小 2**



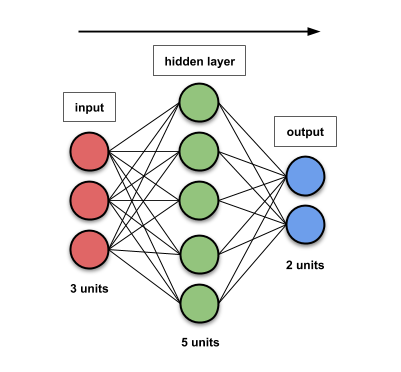


图 1.1：FFNN 输入大小为 3，隐藏层大小为 5，输出大小为 2。的单位。

* ***我***= 3
* ***小时***= 5
* ***○*** = 2

*num\_params*  
*=*层之间的连接数 + 每层的偏差  
= **(3×5 + 5×2) + (5+2)**  
**= 32**

输入=**输入**（（**无**，**3**））  
密集=**密集**（**5**）（输入）  
输出=**密集**（**2**）（密集）  
模型=**模型**（输入，输出）

**例 1.2：输入大小 50，隐藏层大小 [100,1,100]，输出大小 50**



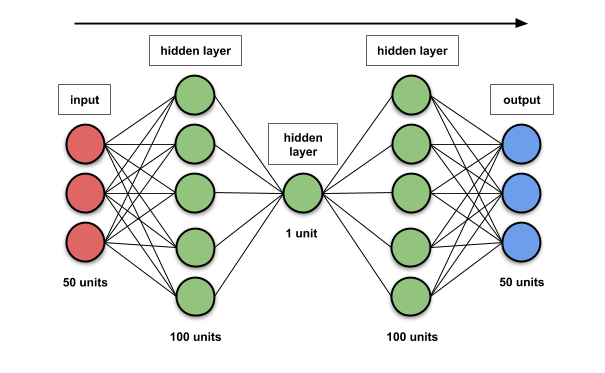


图 1.2：具有 3 个隐藏层的 FFNN。图形不反映实际编号。的单位。

* ***我***= 50
* ***h*** = 100, 1, 100
* ***Ø*** = 50

*num\_params*  
*= 层之间的连接数 + 每层的偏差*  
*=* **(50×100 + 100×1 + 1×100 + 100×50) + (100+1+100+50)**  
**= 10,451**

输入=**输入**（（**无**，**50**））  
密集=**密集**（**100**）（输入）  
密集=**密集**（**1**）（密集）  
密集=**密集**（**100**）（密集）  
输出=**密集**（**50**）（密集）  
模型=**模型**（输入输出）

## 2. RNN

* ***克***，没有。一个单元中的 FFNN（RNN 有 1 个，GRU 有 3 个，LSTM 有 4 个）
* ***h*** , 隐藏单元的大小
* ***i***，输入的维度/大小

由于每个 FFNN 都有***h* ( *h* + *i* ) + *h 个***参数，我们有

*num\_params 参数 =* ***G* × [*H*(*H*+*一世*) + *H*]**

**示例 2.1：具有 2 个隐藏单元和输入维度为 3 的 LSTM。**



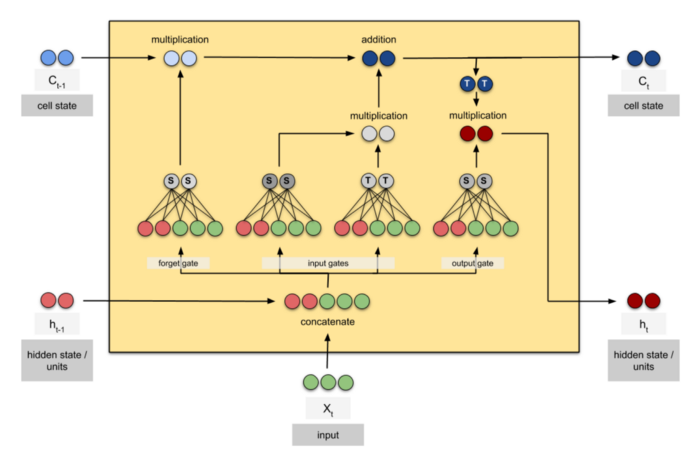


图 2.1：一个 LSTM 单元。取自[这里](https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45)。

* ***g*** = 4（LSTM 有 4 个 FFNN）
* ***小时***= 2
* ***我***= 3

*num\_params*  
*=* ***g* × [ *h* ( *h* + *i* ) + *h* ]**  
**= 4 × [2(2+3) + 2]**  
**= 48**

input = **Input** (( **None** , **3** ))   
lstm = **LSTM** ( **2** )(input)   
model = **Model** (input, lstm)

**示例 2.2：具有 5 个隐藏单元和输入大小为 8（其输出连接）的堆叠双向 GRU + 具有 50 个隐藏单元的 LSTM**



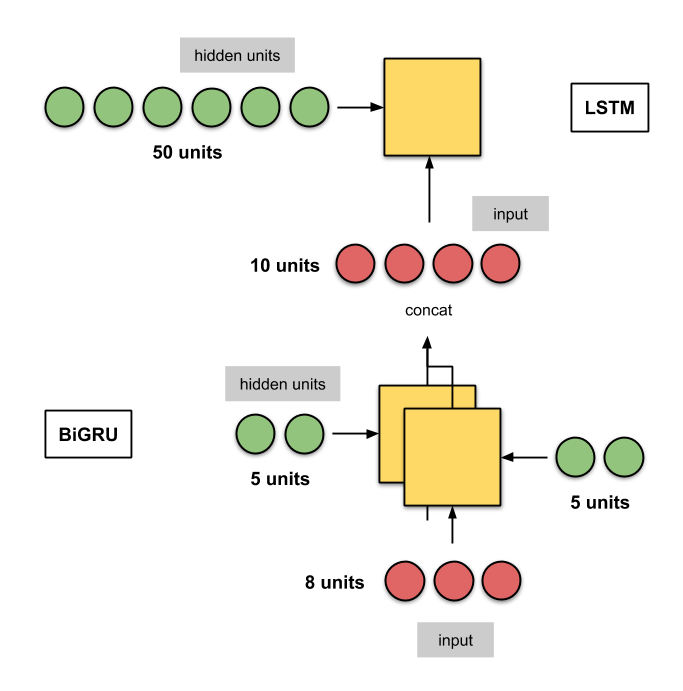


图 2.2：由 BiGRU 和 LSTM 层组成的堆叠 RNN。图形不反映实际编号。的单位。

具有 5 个隐藏单元和输入大小为 8 的双向 GRU

* ***g*** = 3（GRU 有 3 个 FFNN）
* ***小时***= 5
* ***我***= 8

*num\_params\_layer1*  
*=* **2 × *g* × [ *h* ( *h* + *i* ) + *h* ]** (第一项是 2 因为双向性) = **2 × 3 × [5(5+8) + 5]** = **420**

具有 50 个隐藏单元的 LSTM

* ***g*** = 4（LSTM 有 4 个 FFNN）
* ***小时***= 50
* ***i*** = 5+5（双向 GRU 连接的输出；GRU 的输出大小为 5，与隐藏单元的数量相同）

*num\_params\_layer2*  
*=* ***g* × [ *h* ( *h* + *i* ) + *h* ]**  
**= 4 × [50(50+10) + 50]**  
**= 12,200**

*total\_params* = 420 + 12,200 = 12,620

input = **Input** (( **None** , **8** ))   
layer1 = **Bidirectional** ( **GRU** ( **5** , return\_sequences= **True** ))(input)   
layer2 = **LSTM** ( **50** )(layer1)   
model = **Model** (input, layer2)

merge\_mode 默认是串联。

## CNN

对于一层，

* ***我***，没有。输入映射（或通道）
* ***f***，过滤器尺寸（只是长度）
* ***哦***，不。输出映射（或通道。这也由使用的过滤器数量定义）

每个输入映射都应用一个过滤器。

*num\_params*  
*= 权重 + 偏差*  
*=* **[ *i ×* ( *f×f* ) *× o* ] *+ o***

**例 3.1：灰度图像使用 2 *×2*过滤器，输出 3 个通道**



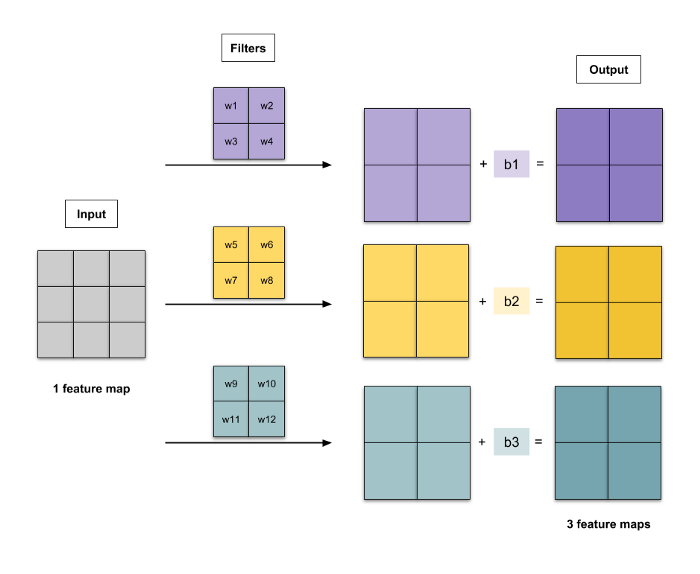


图 3.1：使用 2 *× 2*滤波器对灰度图像进行卷积以输出 3 个通道。这里有 15 个参数——12 个权重和 3 个偏差。

* ***i*** = 1（灰度只有 1 个通道）
* ***f*** = 2
* ***○*** = 3

*num\_params*  
*=* **[ *i ×* ( *f×f* ) *× o* ] *+ o***  
***=* [1 *×* (2 *×* 2) *×* 3] *+* 3**  
**= 15**

input = **Input** (( **None** , **None** , **1** ))   
conv2d = **Conv2D** (kernel\_size= **2** , filters= **3** )(input)   
model = **Model** (input, conv2d)

**例 3.2：RGB 图像带 2×2 过滤器，1 通道输出**

每个输入特征图有 1 个过滤器。生成的卷积被逐元素添加，并且偏置项被添加到每个元素。这给出了具有 1 个特征图的输出。



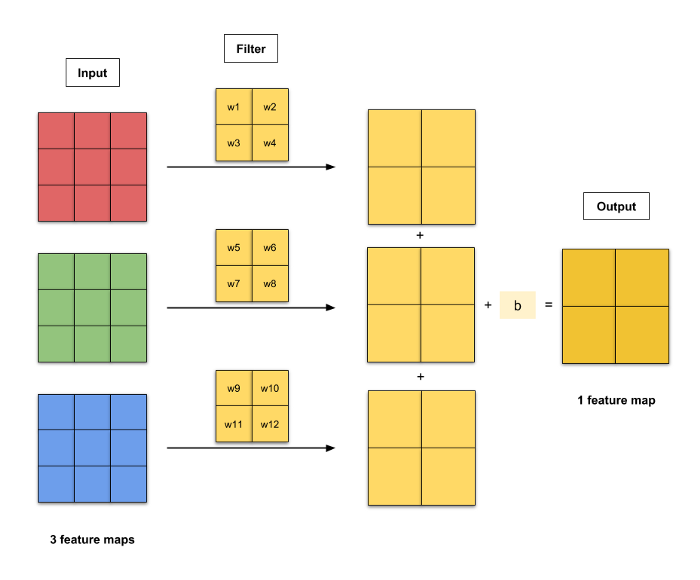


图 3.2：使用 2 *×2*滤波器卷积 RGB 图像以输出 1 个通道。这里有 13 个参数——12 个权重和 1 个偏差。

* ***i*** = 3（RGB 图像有 3 个通道）
* ***f*** = 2
* ***○*** = 1

*num\_params*  
*=* **[ *i ×* ( *f×f* ) *× o* ] *+ o***  
***=* [3 × (2×2) × 1] + 1** = **13**

input = **Input** (( **None** , **None** , **3** ))   
conv2d = **Conv2D** (kernel\_size= **2** , filters= **1** )(input)   
model = **Model** (input, conv2d)

**例 3.3：2 通道图像，2×2 滤波器，3 通道输出**

每个输入特征图有 3 个过滤器（紫色、黄色、青色）。生成的卷积被逐元素添加，并且偏置项被添加到每个元素。这给出了具有 3 个特征图的输出。



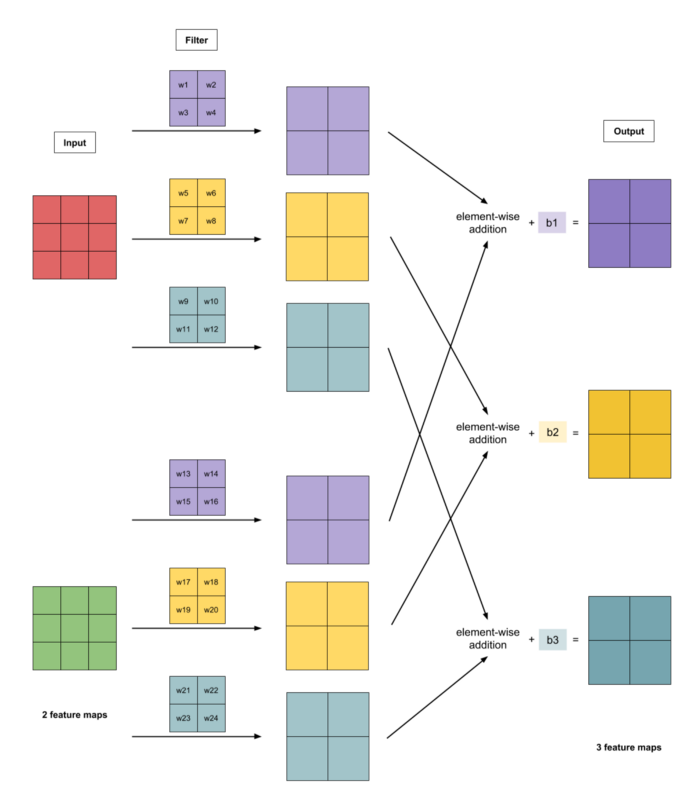


图 3.1：2 通道图像与 2 *×2*滤波器的卷积输出 3 通道。这里有 27 个参数——24 个权重和 3 个偏差。

* ***我***= 2
* ***f*** = 2
* ***○*** = 3

*num\_params*  
*=* **[ *i ×* ( *f×f* ) *× o* ] *+ o***  
***=* [2 × (2×2) × 3] + 3** = **27**

input = **Input** (( **None** , **None** , **2** ))   
conv2d = **Conv2D** (kernel\_size= **2** , filters= **3** )(input)   
model = **Model** (input, conv2d)

目前为止就这样了！如果您有任何反馈，请在下方留言！