

刷臉背後，卷積神經網絡的數學原理原來是這樣的

新機器視覺 前天

點擊下方

視覺/圖像重磅乾貨，第一時間送達



新機器視覺

最前沿的機器視覺與計算機視覺技術

206篇原創內容



公眾號

本文轉自|深度學習這件小事

計算機視覺技術在日常生活中有著非常普遍的應用：發朋友圈之前自動修圖、網上購物時刷臉支付.....在這一系列成功的應用背後，卷積神經網絡功不可沒。本文將介紹卷積神經網絡背後的數學原理。

在自動駕駛、醫療以及零售這些領域，計算機視覺讓我們完成了一些直到最近都被認為是不可能的事情。今天，自動駕駛汽車和無人商店聽起來不再那麼夢幻。事實上，我們每天都在使用計算機視覺技術——我們用自己的面孔解鎖手機，將圖片上傳到社交網絡之前進行自動修圖.....卷積神經網絡可能是這一巨大成功背後的關鍵組成模塊。這次，我們將要使用卷積神經網絡的思想來拓寬我們對神經網絡工作原理的理解。打個預防針，本文包含相當複雜的數學方程，但是，你也不必為自己不喜歡線性代數和微積分而沮喪。我的目標並不是讓你記住這些公式，而是為你提供一些關於底層原理的直覺認知。

簡介

過去我們接觸到了密集連接的神經網絡。那些神經網絡中，所有的神經元被分成了若干組，形成了連續的層。每個這樣的單元都與相鄰層的每一個單獨的神經元相連接。下圖所示的是這樣一個架構。

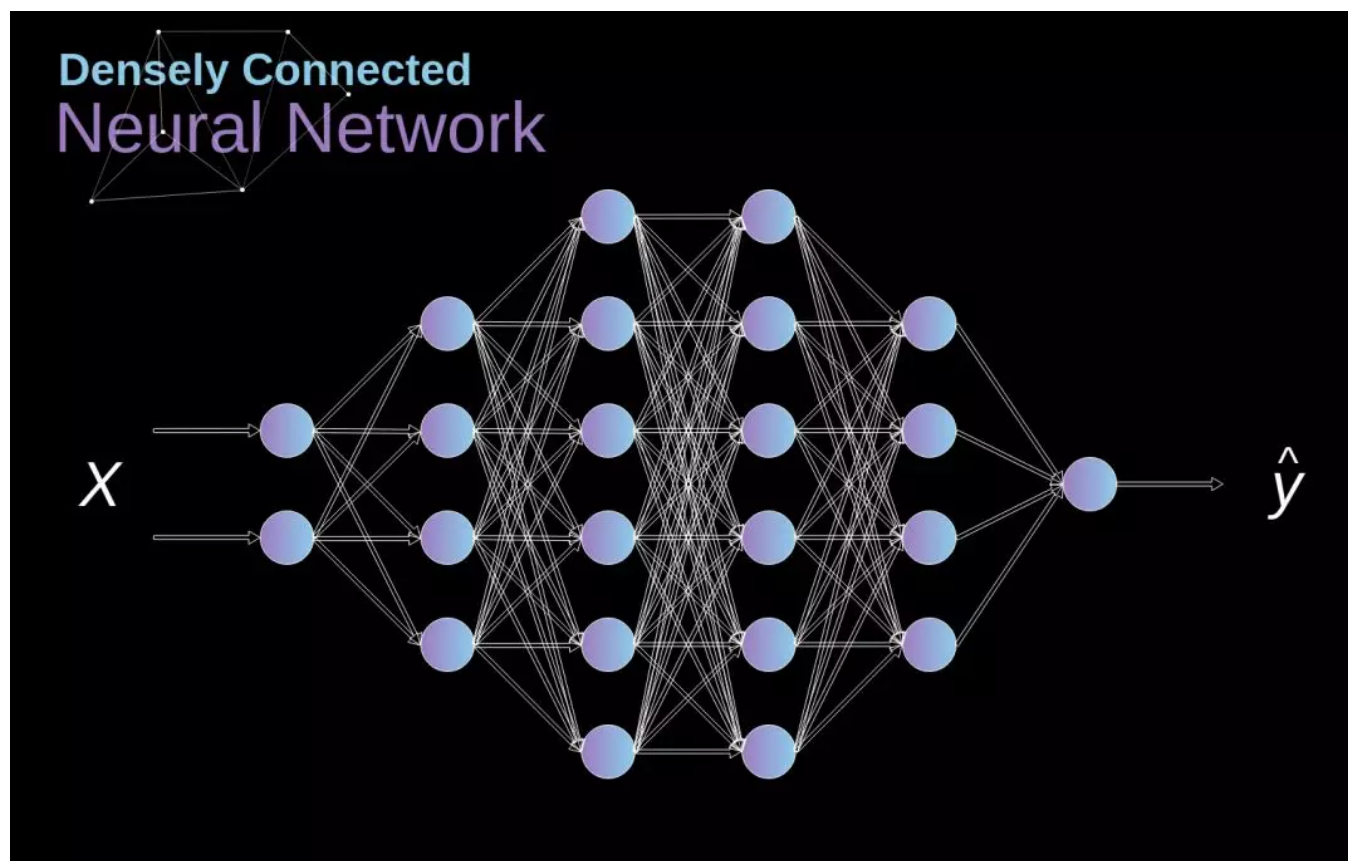


圖1：

當我們基於一個有限的固定特徵集合解決分類問題的時候，這種方法是很奏效的——例如，我們根據足球運動員在比賽中記錄的統計數據來預測他的位置。但是，當處理照片的時候，問題變得更加複雜。當然，我們可以把每個像素的亮度視作一個單獨的特徵，然後將它作為密集網絡的輸入傳遞進去。不幸的是，為了讓它能夠應付一張典型的智能手機照片，我們的網絡必須包含數千萬甚至上億的神經元。另一方面，雖然我們可以將照片縮小，

但是我們也會在這個過程中損失有價值的信息。所以我們馬上就會發現，傳統的策略是沒有用的——我們需要一種新的聰明的方法，來盡可能多的利用數據，但同時還要減少必需的計算量和參數。這就是CNN 發揮作用的時候了。

數字照片的數據結構

讓我們先花少許時間解釋一下數字圖像的存儲方式。大多數人可能意識到了，圖像實際上就是巨大的數字矩陣。每個數字代表的是一個單獨像素的亮度。在RGB 模型中，彩色圖片是由3 個這樣的矩陣組成的，每個矩陣對應著3 個顏色通道（紅、綠、藍）中的一個。在黑白圖像中，我們僅使用一個矩陣。每個矩陣都存儲著0 到255 的數值。這個數值範圍是圖像存儲信息的效率（256 個數值剛好對應一個字節）和人眼敏感度之間的折中（我們僅能區分同種顏色的幾種有限色度）。

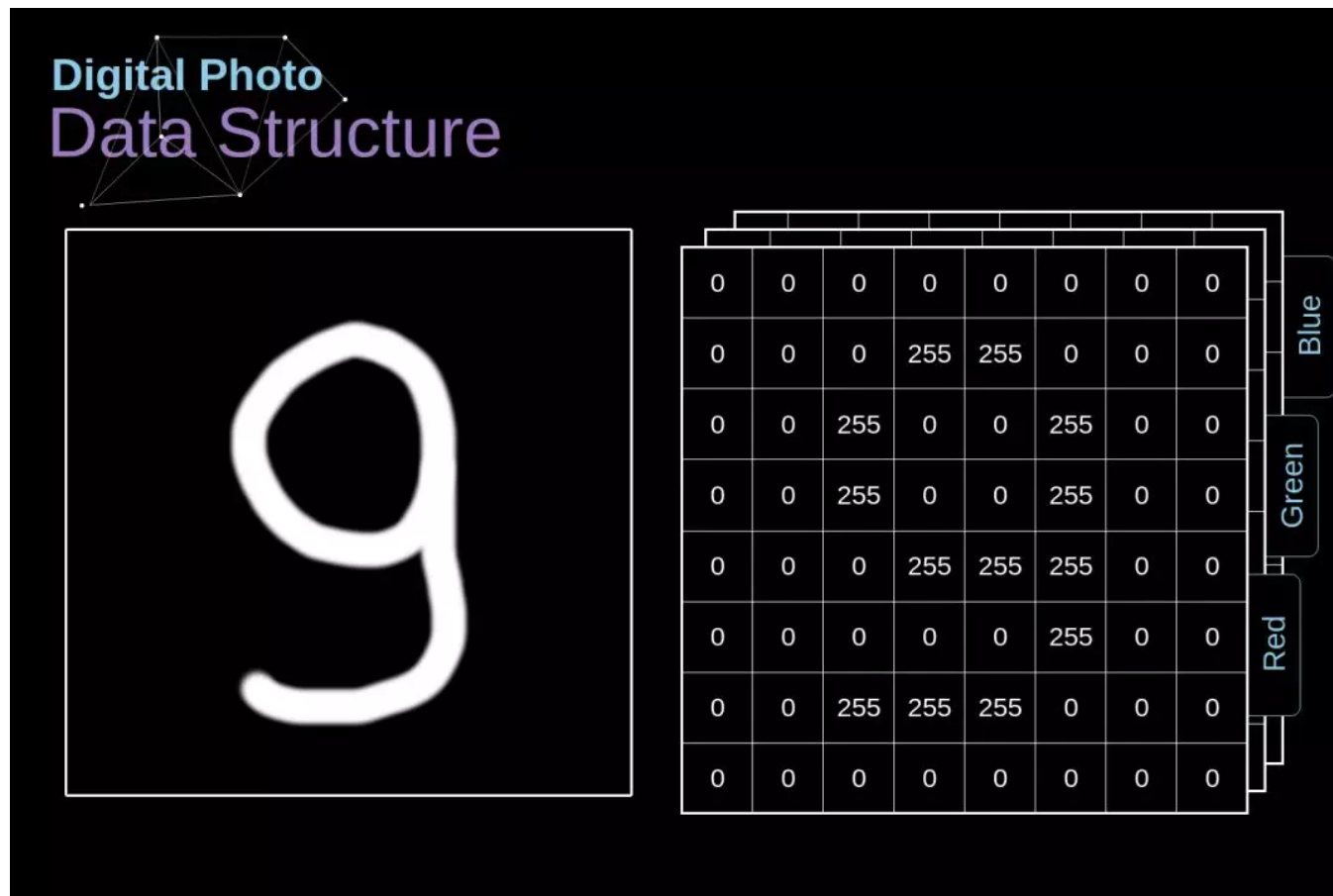


圖2. 數字圖像的數據結構

卷積

核卷積並不僅僅用在卷積神經網絡中，它也是很多其他計算機視覺算法的關鍵元素。這個過程是這樣的：我們有一個小的數字矩陣（稱作卷積核或濾波器），我們將它傳遞到我們的圖像上，然後基於濾波器的數值進行變換。後續的特徵圖的值要通過下面的公式計算，其中輸入圖像被記作 f ，我們的卷積核為 h 。計算結果的行列索引分別記為 m 和 n 。

$$G[m, n] = (f * h)[m, n] = \sum_j \sum_k h[j, k] f[m - j, n - k]$$

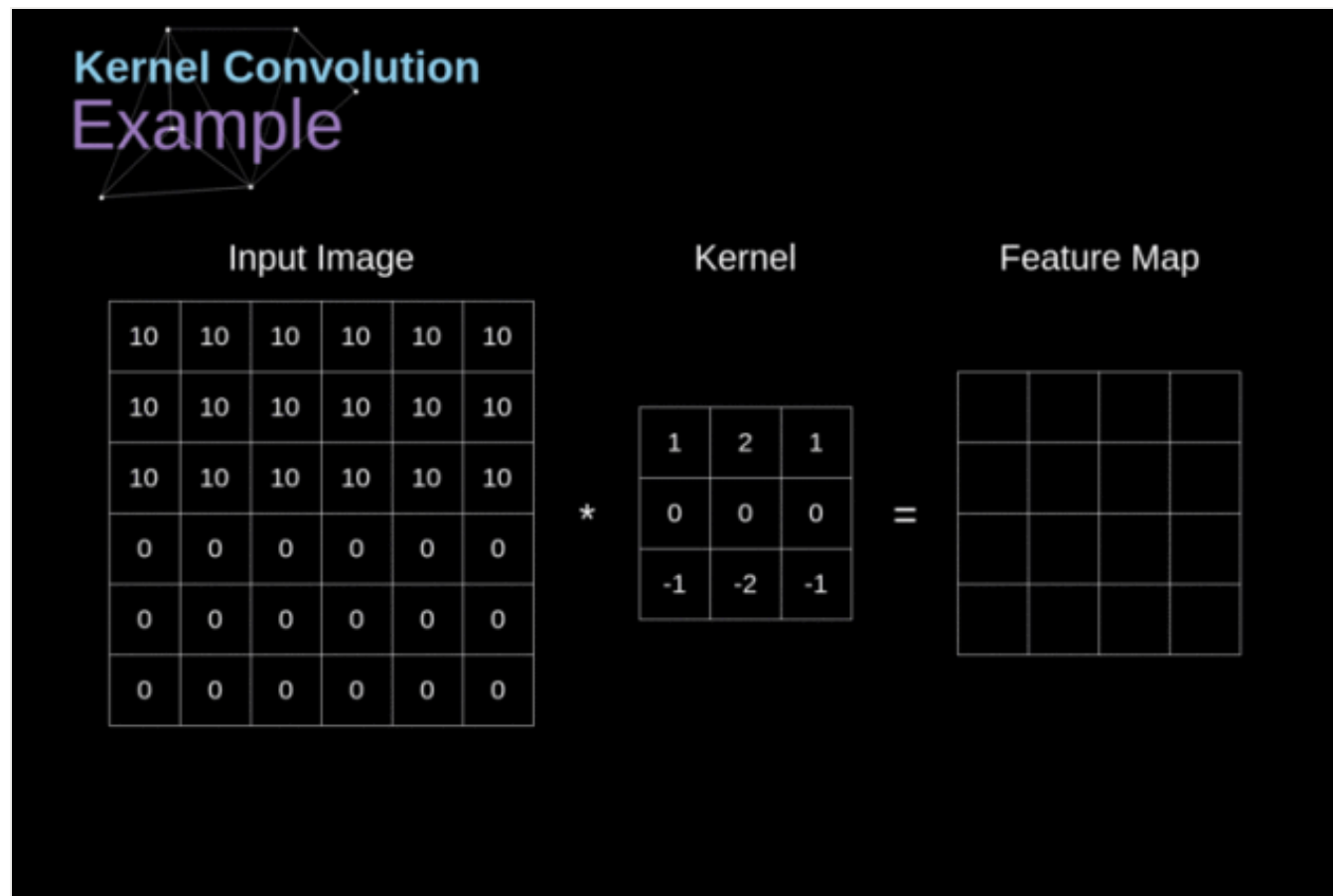


圖3. 核卷積的例子

在將我們的濾波器放在選中的像素上之後，我們將捲積核中的每一個數值和圖像中對應的數值成對相乘。最後將乘積的結果相加，然後把結果放在輸出特徵圖的正確位置上。我們在上邊的動畫中可以以一個微觀的形式看到這個運算的過程，但是更有趣的是我們在整幅圖像上執行這個運算得到的結果。圖4 展示了用數個濾波器做卷積的結果。



圖4. 用卷積核尋找邊緣

Valid 和 Same 的捲積

如圖3 所示，當我們在用3x3 的捲積核在6x6 的圖像上執行卷積時，我們得到了4x4 的特徵圖。這是因為在我們的圖像裡面，只有16 個獨特的位置來放置卷積核。由於我們的圖像的尺寸在每次卷積的時候都會收縮，在圖像完全消失之前，我們只能做有限次的捲積。此外，如果我們注意一下卷積

核是如何在圖像上移動的，我們會發現，邊緣的像素會比中央的像素影響更小。這樣的話我們會損失圖片中包含的一些信息，你可以在下圖看到，像素的位置是如何改變它對特徵圖的影響的。

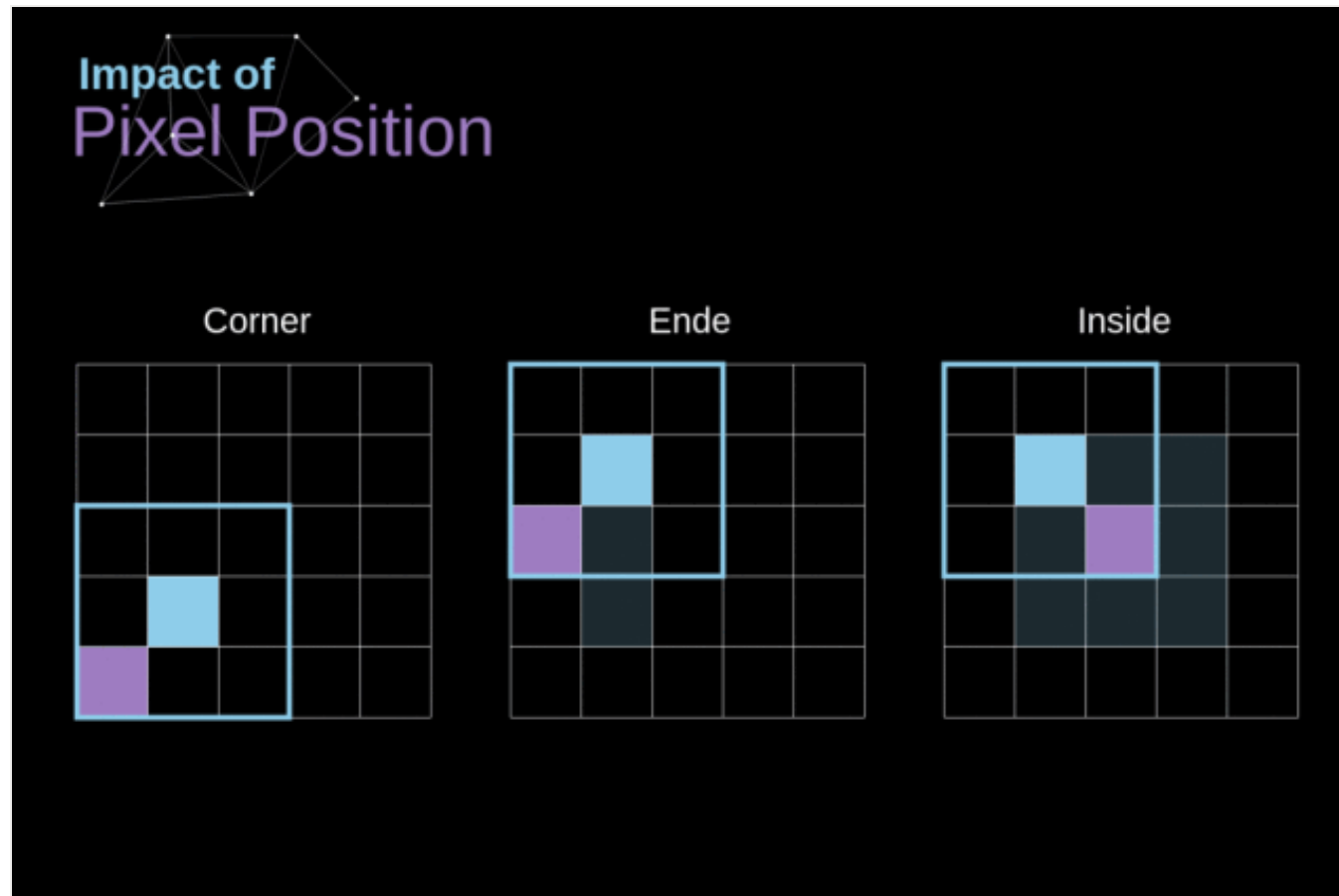


圖5. 像素位置的影響

為了解決這兩個問題，我們可以使用一個額外的邊界來填充圖像。例如，如果我們使用1 像素的填充，我們將圖像的尺寸增大到了8x8，這樣，3x3 的濾波器的輸出將會成為6x6。通常在實際中我們用0 來做額外的填充。根據我們是否使用填充，我們會進行兩種類型的捲積——Valid 和Same。

命名相當令人費解，所以在這裡解釋一下：valid 代表我們使用的是原始圖像，same 代表我們在圖像周圍使用了邊界，因此輸入和輸出的圖像大小相同。在第二種情況下，擴充的寬度應該滿足下面的方程，其中p 是padding（填充），f 是濾波器的維度（通常是奇數）。

$$p = \frac{f - 1}{2}$$

跨步卷積

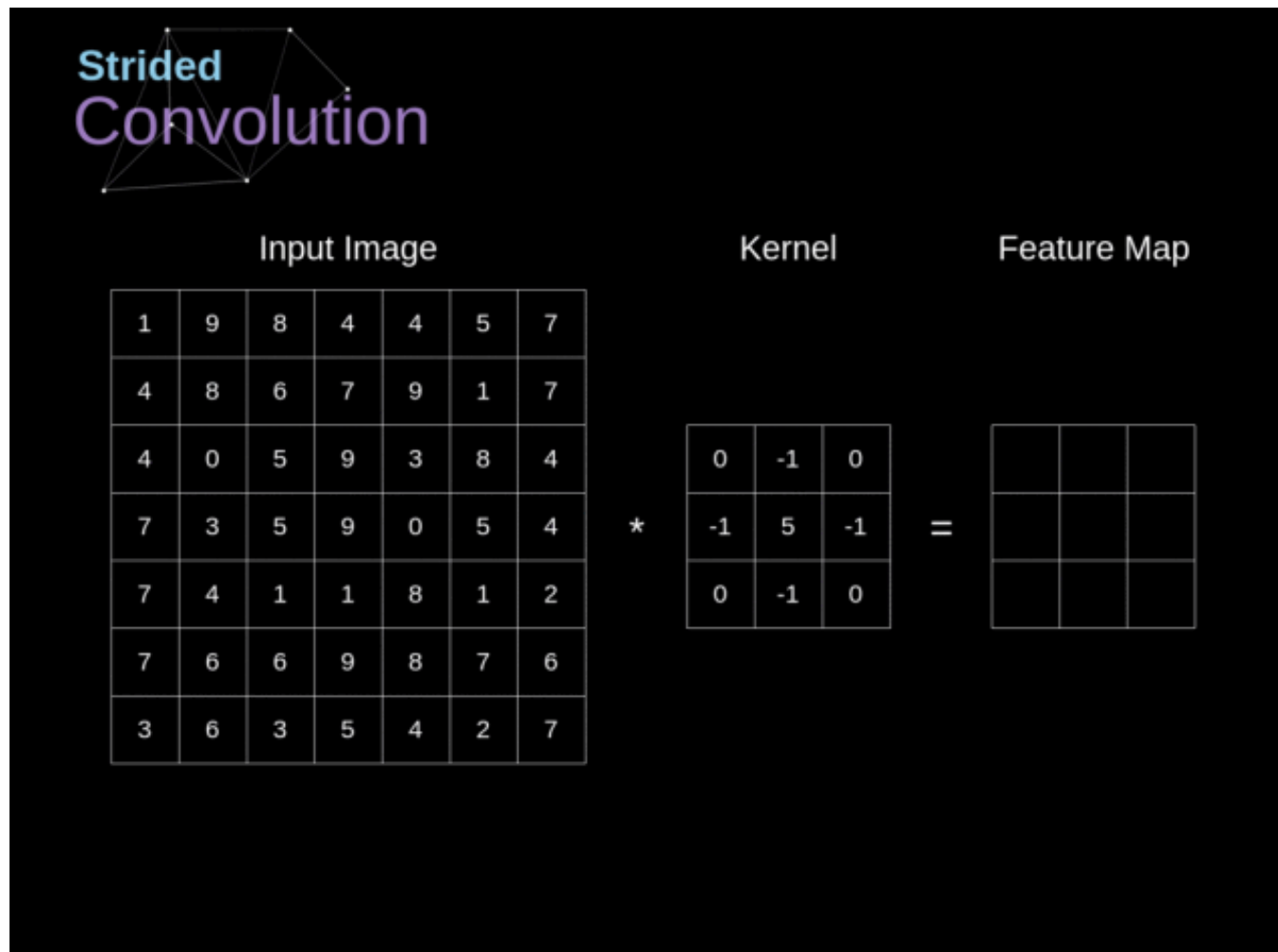



圖6. 跨步卷積的例子

在之前的例子中，我們總是將捲積核移動一個像素。但是，步長也可以看做是卷積層的一個參數。在圖6 中，我們可以看到，如果我們使用更大的步長，卷積會成為什麼樣子。在設計CNN 結構時，如果我們想讓接受域有更少的重疊或者想讓特徵圖有更小的空間維度，那麼我們可以決定增大步長。考慮到擴充和跨步，輸出矩陣的維度可以使用下面的公式計算：

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - f}{s} + 1 \right\rfloor$$

轉換到第三個維度

立體卷積是一個非常重要的概念，它不僅讓我們能夠處理彩色圖像，而且更重要的是，可以在一個單獨的層上使用多個濾波器。最重要的規則是，濾波器和你想在其上應用濾波器的圖像必須擁有相同的通道數。基本上，我們繼續使用和圖3類似的示例，儘管我們這次從第三個維度讓矩陣中的數值對相乘。如果我們想在同一張圖像上應用多個濾波器，我們會為每個濾波器獨立地計算卷積，然後將計算結果逐個堆疊，最後將他們組合成一個整體。得到的 

$$[n, n, n_c] * [f, f, n_c] = \left[\left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor, \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor, n_f \right]$$

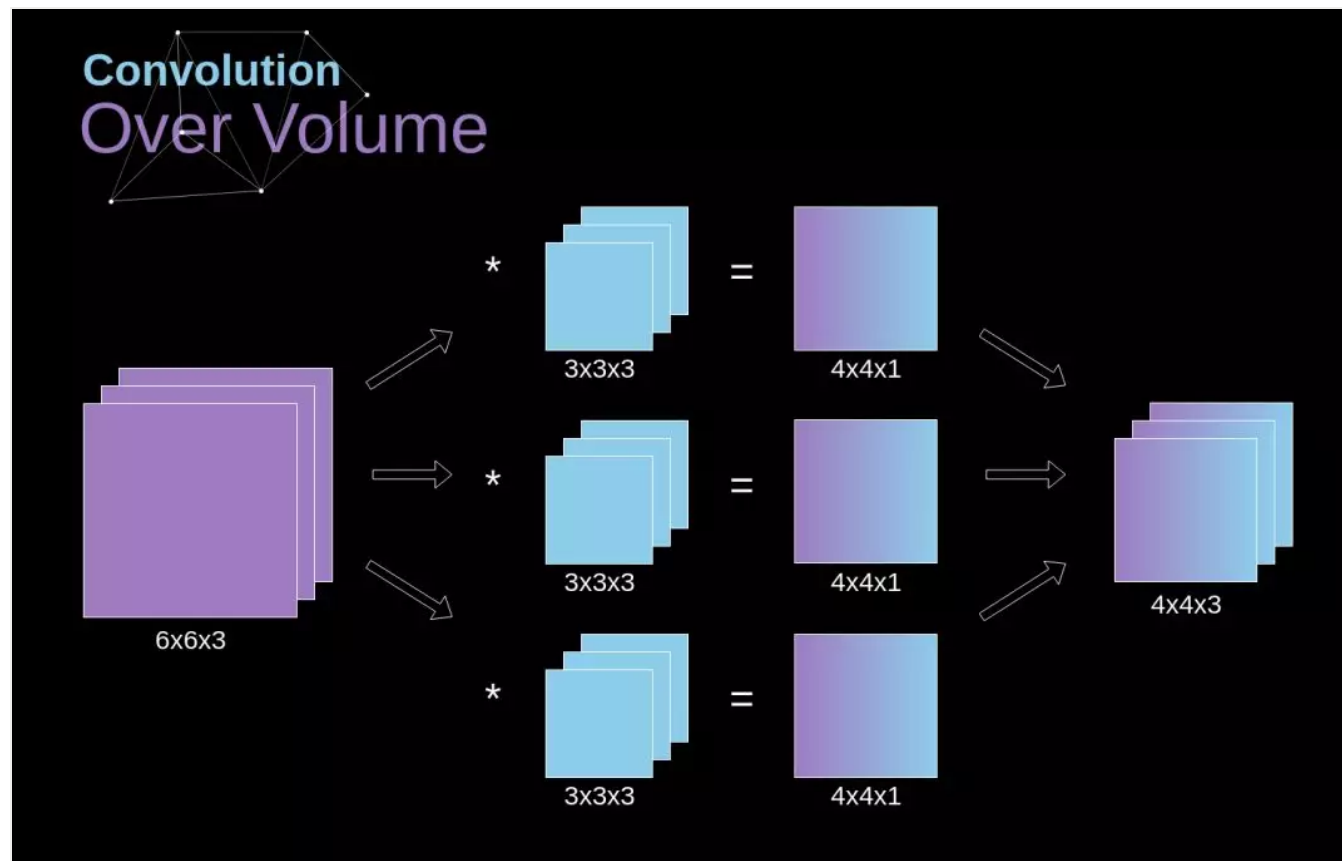


圖7. 立體卷積

卷積層

使用我們今天所學內容構造一個卷積層的時間到了。我們的方法幾乎與用在密集連接神經網絡上的方法相同，唯一的差別就是不使用簡單的矩陣相乘，這一次我們將會使用卷積。前向傳播包含兩個步驟。第一步是計算中間結果 Z ，它是由前一層的輸入數據與張量 W （包含濾波器）的捲積結果，加上偏置項 b 得到的。第二步是給我們的中間結果應用一個非線性的激活函數（我們的激活函數記作 g ）。矩陣方程的愛好者將在下面找到合適的數學公式。在下面的插圖中，你可以看見一個小型的可視化，它描述了我們方程中用到的張量的維度。

$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \quad \mathbf{A}^{[l]} = g^{[l]}(\mathbf{Z}^{[l]})$$

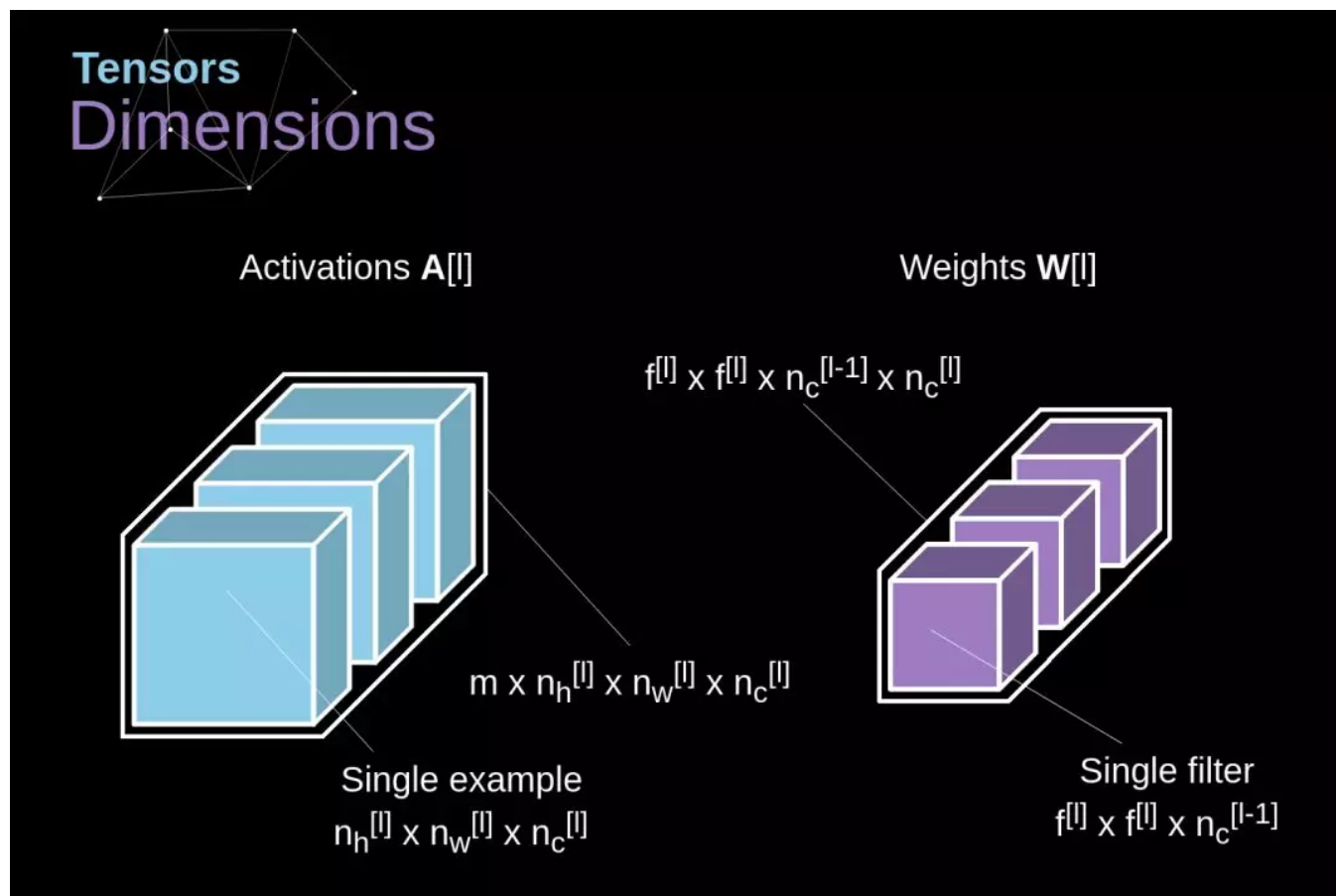


圖8. 張量維度

連接剪切和參數共享

在本文開始，由於需要學習的參數數量巨大，我提到密集連接神經網絡在處理圖像方面是很弱的。既然我們已經了解了關於卷積的所有內容，讓我們來考慮一下它是如何優化計算的吧。在下圖中，2D 卷積以一種稍微不同的方式進行了可視化——用數字1-9 標記的神經元組成接收後續像素亮度的輸入層，AD 這4 個單元代表的是計算得到的特徵圖元素。最後但同等重要的是，I-IV 是卷積核中的數值——它們必須被學習到。

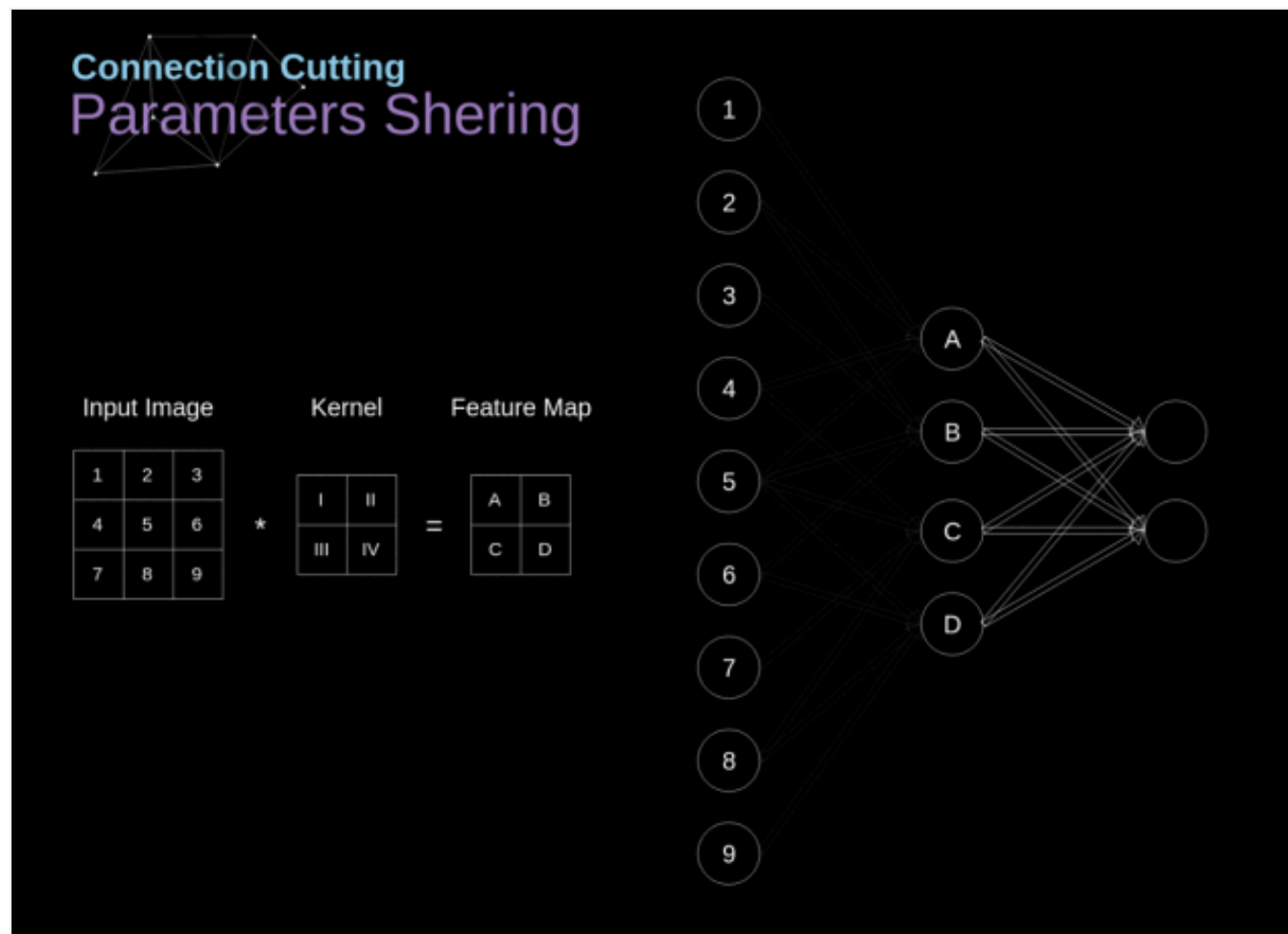


圖9. 連接剪切和參數共享

現在，讓我們聚焦於卷積層的兩個重要屬性。第一，你可以看到，連續兩層中，並不是所有的神經元都是彼此相連的。例如，單元1 僅僅會影響到A 的值。第二，我們發現，一些神經元會共享相同的權重。這兩個屬性都意味著我們要學習的參數數量要少很多。順便說一下，值得注意的是，濾波器中的每個值都會影響到特徵圖中的每個元素——這在反向傳播中是特別重要的。

卷積層反向傳播

任何一個曾經試圖從零編寫自己的神經網絡的人都知道，前向傳播遠遠不到成功的一半。真正有趣的是當你開始反向傳播的時候。現在，我們不必在反向傳播上花心思——深度學習框架都為我們做好了，但是我認為，了解背後發生的東西是很值得的。就像在密集連接神經網絡中一樣，我們的目標是在一個叫做梯度下降的過程中計算導數，然後使用它們來更新參數值。

在計算中我們會使用鍊式法則——這個我在之前的文章中提到過。我們想要評估參數的變化對結果特徵圖的影響，然後評估它對最終結果的影響。在開始進入細節之前，讓我們來統一下將會用到的數學符號——為了讓事情變得容易一些，我會放棄偏導數的完整符號，而會使用下面的簡寫符號。但是請記住，這個符號始終代表代價函數的偏導數。

$$\mathbf{dA}^{[l]} = \frac{\partial L}{\partial \mathbf{A}^{[l]}} \quad \mathbf{dZ}^{[l]} = \frac{\partial L}{\partial \mathbf{Z}^{[l]}} \quad \mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} \quad \mathbf{db}^{[l]} = \frac{\partial L}{\partial \mathbf{b}^{[l]}}$$

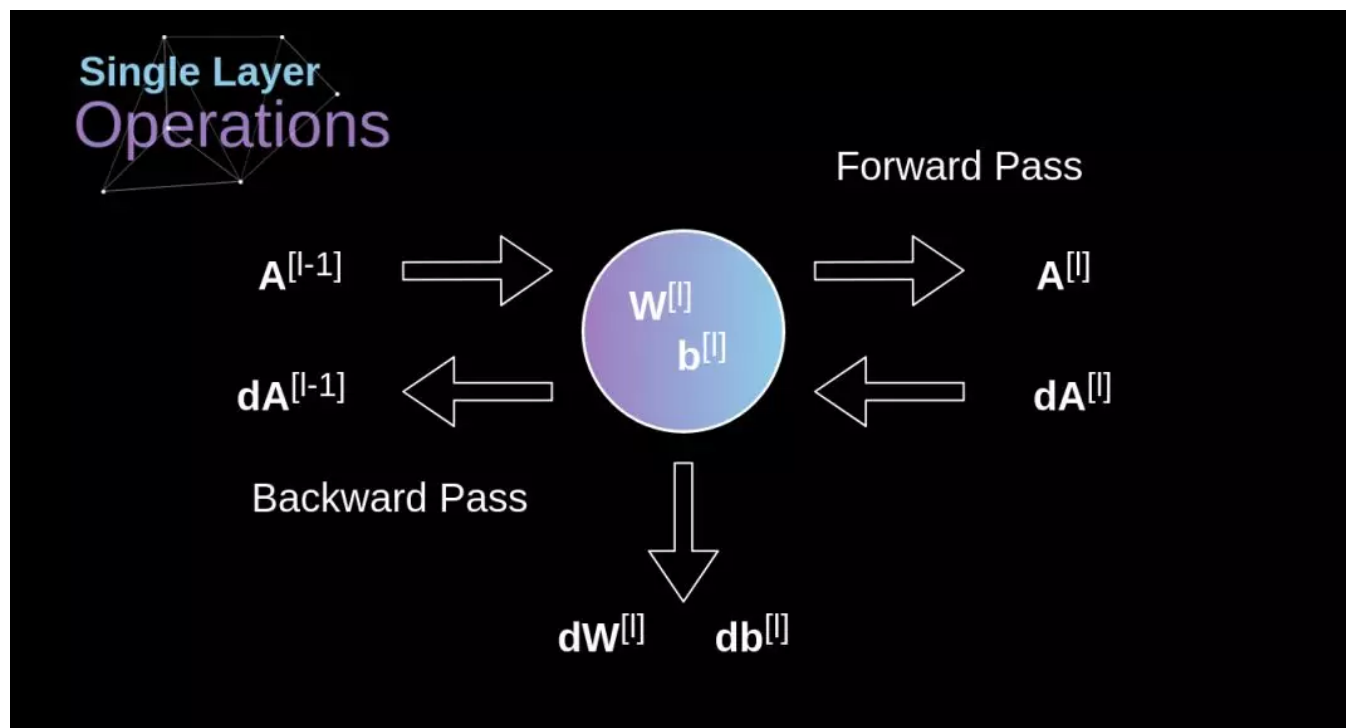


圖10. 一個卷積層在前向和反向傳播中的輸入和輸出數據

我們的任務是計算 $dW^{[l]}$ 和 $db^{[l]}$ ——它們是與當前層的參數相關的導數，還要計算 $dA^{[l-1]}$ ，它們會被傳遞到之前的層。如圖10所示，我們以 $dA^{[l]}$ 為輸入。當然，這些對應張量的維度都是相同的， dW 和 W ， db 和 b ，以及 dA 和 A 。第一步就是通過在我們的輸入張量上應用我們的激活函數的導數，得到中間值 $dZ^{[l]}$ 。根據鍊式法則，這個運算的結果在後面會被用到。

$$dZ^{[l]} = dA^{[l]} * g'(Z^{[l]})$$

現在，我們需要處理卷積神經網絡自身的反向傳播，為了達到這個目的，我們會使用一個叫做全卷積的矩陣運算——見下圖。請注意，我們在這裡使用的卷積核會提前旋轉180°。這個運算可以通過下面的公式描述，其中的濾波器記作 W ， $dZ[m,n]$ 是一個標量，它屬於從前一層得到的偏導數。

$$dA+ = \sum_{m=0}^{n_h} \sum_{n=0}^{n_w} \mathbf{W} \cdot d\mathbf{Z}[m, n]$$

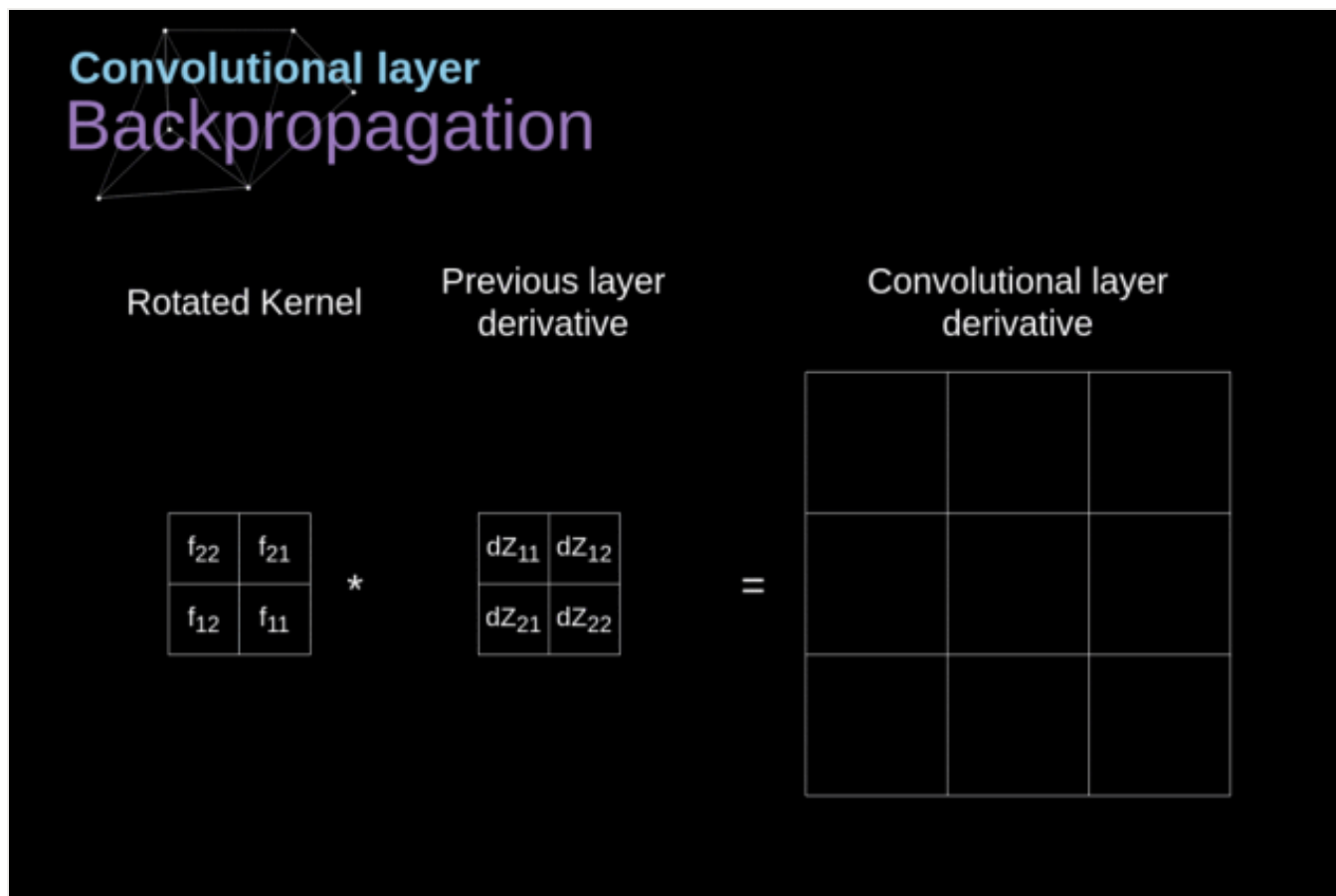


圖11. 全卷積

池化層

除了卷積層，CNN 通常會用到所謂的池化層。它們最早被用來減小張量的大小以及加速運算。這些層是比較簡單的——我們需要將我們的圖像分成不同的區域，然後在每一個部分上執行一些運算。例如，對Max Pool 層而言，我們會選擇每個區域的最大值，並將它放到對應的輸出區域。與卷積層的情況一樣，我們有兩個可用的超參數——濾波器大小和步長。最後但同樣重要的一點是，如果你對一個多通道的圖像執行池化操作，那麼每一個通道的池化應該單獨完成。

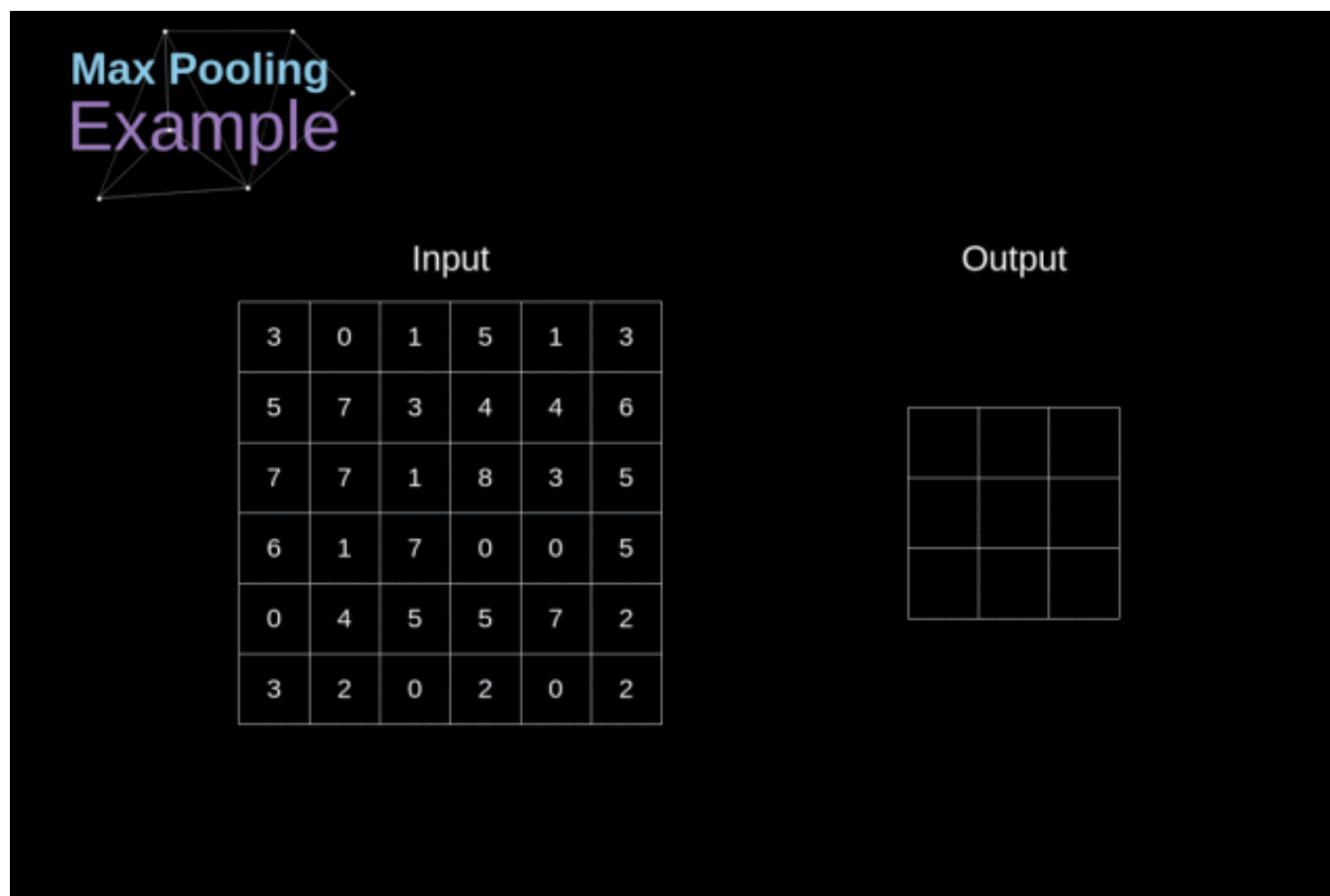


圖12. 最大池化（max pooling）的例子

池化層反向傳播

我們在這篇文章中只討論最大池化反向傳播，但是我們學到的規則是適用於所有類型的池化層的——只需要做微小的調整即可。因為在這種層中，我們沒有任何必須更新的參數，所以我們的任務就是合適地分配梯度。我們記得，在最大池化的前向傳播中，我們選擇的是每個區域的最大值，並將它傳遞到了下一層。所以在反向傳播中也是很清晰的，梯度不應該影響前向傳播中不包含的矩陣的元素。實際上，這是通過創建一個掩膜來完成的，這個掩膜記住了前一階段數值的位置，我們可以在後面轉移梯度的時候用到。

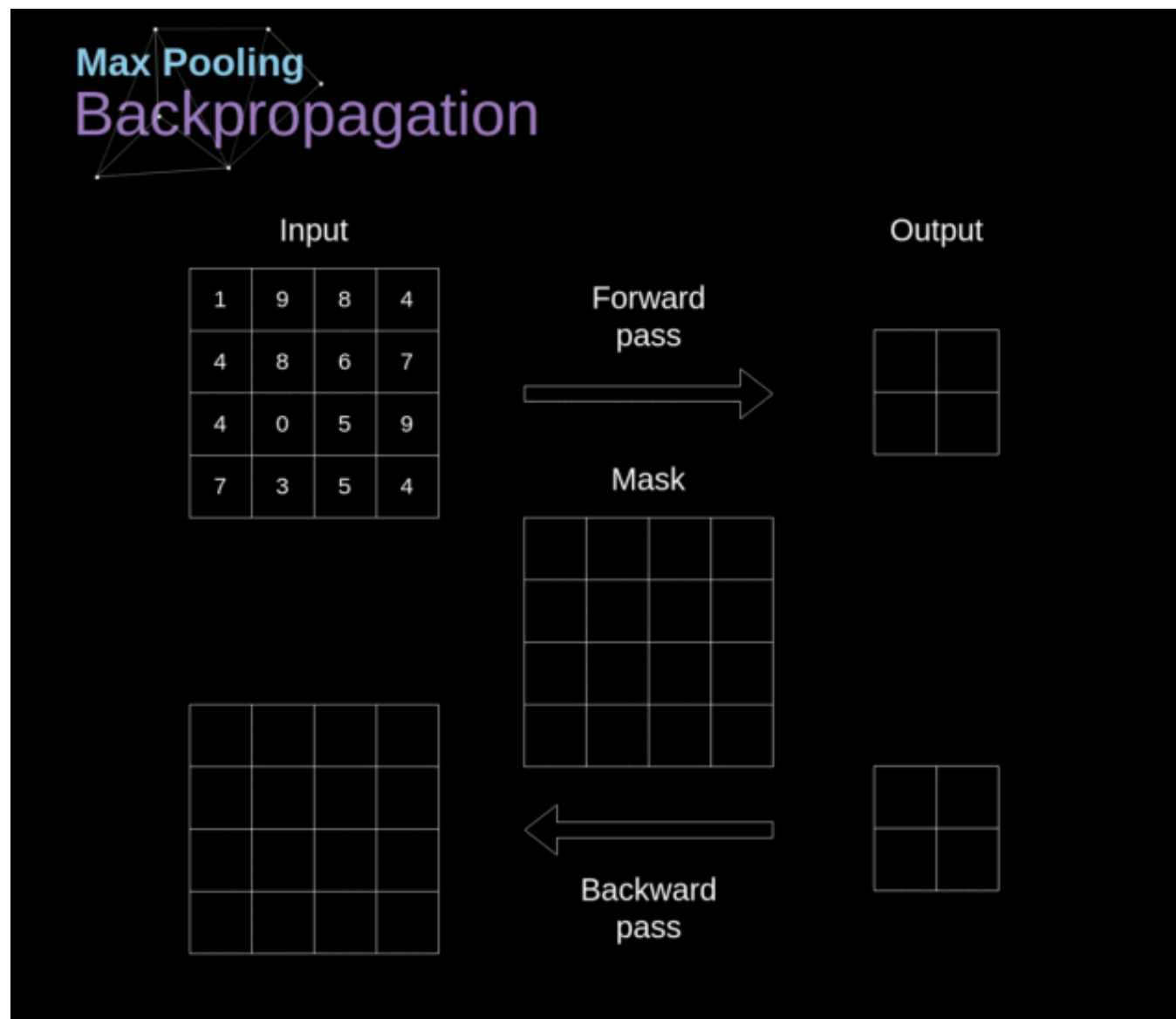


圖13. 最大池化反向傳播

原文鏈接: <https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9>

—版權聲明—

僅用於學術分享，版權屬於原作者。

若有侵權，請聯繫微信號: yiyang-sy 刪除或修改！

—THE END—

走进新机器视觉 · 拥抱机器视觉新时代

新机器视觉 —— 机器视觉领域服务平台
媒体论坛/智库咨询/投资孵化/技术服务

商务合作：
投稿咨询：
产品采购：

(微信号)

长按扫描右侧二维码关注“新机器视觉”公众号



喜歡此內容的人還喜歡

最通俗的神經網絡入門教程

新機器視覺



道理我都懂，但是神經網絡反向傳播時的梯度到底怎麼求？

小白學視覺



同樣都是調參，為什麼人家的神經網絡比我牛逼100 倍？

小白學視覺

