# 卷積神經網路（一）tensor 定義

山與水你和我
體系結構小白，影像處理小白



號 目錄 收起

基本設計

基本函數

共享指針



實現卷積神經網路的訓練和推斷，整個流程最基本的就是資料如何儲存。從影像到具體數據，數據在網路層與層之間儲存、流動，到最後輸出為機率，無處不用。在Pytorch 中，浮點資料是FloatTensor，可以像操作numpy 一樣做運算，十分方便。

目標：自訂一個張量Tensor。

## 基本設計

在訓練CNN 時，接受的資料類型基本上是$B \times C \times H \times W$，意思是這個batch 有$B$張圖像的內容，每張圖像的內容有$C$個通道，每個通道的特徵圖大小為$H \times W$，本人的設計為

```
std::vector<Tensor>
```

std::vector 的size() 代表了$B$。每個Tensor 的基本屬性有$C, H, W$，除此之外，還需要有具體內容，名為data，資料型態一般是float 或double；為了方便後期觀察和區分，可以有一個名字name，如下

```
using data_type = float;

class Tensor3D {
public:
    const int C, H, W;
    data_type* data;
    const std::string name;
};
```

data 是一個指針，是Tensor 內容的起始位址。

如果要存取該Tensor 第$ch$個通道中第$i$行第$j$列的數據，可以寫成

```
index = ch * H * W + i * W + j;
data[index];
```

這說明了資料的排列順序，從0 到$H*W$個數是第0 個通道的特徵圖內容，從$H*W$到$2*H*W$是第1 個頻道的特徵圖內容.....

根據上面，可以得到訓練時候的訪問模式

```
// 输入是 input, 是一个 std::vector<Tensor>
const int batch_size = input.size();
for(int b = 0; b < batch_size; ++b) {
    // 获取第 b 张图像特征的起始地址
    data_type* const batch_ptr = input[b]->data;
    for(int ch = 0;ch < input[b]->C; ++ch) {
        // 获取这张图象的第 ch 个通道的起始地址
        data_type* const ch_ptr = batch_ptr + ch * H * W;
        for(int i = 0;i < H; ++i) {
            // 获取这个通道第 i 行数据的起始地址
            data_type* const row_ptr = ch_ptr + i * W;
            for(int j = 0;j < W; ++j) {
                // 读取 or 修改这一行第 j 个数据
                // row_ptr[j]
            }
        }
    }
}
```

以上是特徵圖的存取方法，每張影像的特徵看起來是$C \times H \times W$的數據，但實際儲存是已經flatten 展平的一維數據。為了統一，全連接層的輸入可以看成$C \times 1 \times 1$的Tensor。

## 基本函數

numpy.array 或torch.Tensor 除了基本資料的存取之外，還有一系列函數，如找最大值找最小值之類的，在影像分類中，要找最大機率對應的類別作為分類結果，也是用的到的，如下：

```
// 找到这个一维向量的最大值
data_type Tensor3D::max() const {
    return this->data[argmax()];
}

// 找到这个一维向量最大值的位置
int Tensor3D::argmax() const {
    const int length = C * H * W;
    if(data == nullptr) return 0;
    data_type max_value = this->data[0];
    int max_index = 0;
    for(int i = 1;i < length; ++i)
        if(this->data[i] > max_value) {
            max_value = this->data[i];
            max_index = i;
        }
    return max_index;
}
```

後續使用的過程中也發現，在資料data 分配空間之後，一般都需要清零，加個清零函數

```
void Tensor3D::set_zero() {
    const int length = C * H * W;
    // for(int i = 0;i < length; ++i) data[i] = 0;
    std::memset(this->data, 0, sizeof(data_type) * length);
}
```

後續在使用的過程中逐漸加入新的函數，最終版本是

```
using data_type = float;
class Tensor3D {
public:
    const int C, H, W;
    data_type* data;
    const std::string name;
    // 形状 C x H x W, 分配内存
    Tensor3D(const int _C, const int _H, const int _W, const std::string _name="pipeli
        : C(_C), H(_H), W(_W), data(new data_type[_C * _H * _W]), name(std::move(_name
    // 形状 C x H x W, 分配内存
    Tensor3D(const std::tuple<int, int, int>& shape, const std::string _name="pipeline
```

```
        name(std::move(_name)) {}
    // 形状 length x 1 x 1, 此时length = C, 全连接层用得到
    Tensor3D(const int length, const std::string _name="pipeline")
        : C(length), H(1), W(1), data(new data_type[length]), name(std::move(_name)) {
    // 从图像指针中读取内容, 加载到 Tensor.data 中
    void read_from_opencv_mat(const uchar* const img_ptr);
    // 清零
    void set_zero();
    // 找最大值
    data_type max() const;
    int argmax() const;
    // 找最小值
    data_type min() const;
    int argmin() const;
    // 从 tensor 恢复成图像
    cv::Mat opecv_mat(const int CH=3) const;
    // 获取这个 Tensor 的内容长度
    int get_length() const;
    // 获取这个 Tensor 的形状
    std::tuple<int, int, int> get_shape() const;
    // 打印这个 Tensor 的形状
    void print_shape() const;
    // 打印这个 Tensor 在第 _C 个通道的内容
    void print(const int _C=0) const;
    ~Tensor3D() noexcept;
};
```

## 共享指針

Tensor 佔的空間還是比較大，在C++ 中如果作為回傳值，就會呼叫拷貝函數，局部的Tensor 也會呼叫析構函數，為了減少消耗，最終的張量表現形式是

```
using tensor = std::shared_ptr<Tensor3D>;
```

在卷積神經網路中不斷傳遞的是**共享指標的陣列**std::vector<tensor>，也只有當引用計數為0 時Tensor 才會釋放空間。

張量定義完畢！下一步是從映像到Tensor 的建構過程

11 贊同 0 評論 文章

發佈於2022-02-19 12:43

C++ 電腦視覺 卷積神經網路（CNN）

寫下你的評論...

**9 則評論** 預設 最新

**波格**
決定追番，期待大佬後續輸出，關於自己實驗cnn向前推理這塊
2022-02-25 號 回覆 號 喜歡

**山與水你和我** 作者
推理?可以看看這篇文章所在專欄，其實我程式碼裡有inference， 只是我最近比較忙，沒時間寫推理的部分
2022-02-25 號 回覆 號 喜歡

**連線成功**
額。。。。。。你這叫3D矩陣可能好一點。有點侮辱張量這個名字
2023-04-21 號 回覆 號 喜歡

**山與水你和我** 作者
確實不該叫張量，demo 而已
2023-04-21 號 回覆 號 喜歡

**喜洋洋**
膜拜作者，請問一下，有3D卷積的C++研究嗎？能否推薦一下，謝謝
2022-07-27 號 回覆 號 喜歡

**山與水你和我** 作者
你好，本人沒接觸過3D卷積，這種只能去看開源庫了。我寫的只是一個最簡單的實現，沒有加速
2022-07-28 號 回覆 號 喜歡

**李澤昊**
c++11後加入了RVO（return value optimization），回傳一個局部物件時不會呼叫析構函數，拷貝建構函數。 shared ptr反而會增加效能開銷。
2022-02-26 號 回覆 號 喜歡

如果每次網路forward 都回傳局部對象，計算一次，出了作用域就銷毀，有點浪費，所以我選擇回傳指針

2022-02-26 號 回覆 號 喜歡

**山與水你和我** 作者

確實，以前在《深度探索C++物件模型》看過回傳值優化。只是，我後續實現的時候，有類似buffer 一樣的Tensor，同樣也是一些函數的返回值，一直使用到程式結束，我就乾脆返回指針得了🤣

2022-02-26 號 回覆 號 喜歡

## 文章被以下專欄收錄

**電腦視覺基礎**
最簡單的捲積神經網路、ReLU、BN、目標偵測等

**帶你學習AI•深度學習應用之路**
創作內容基於深度學習的理論學習與應用開發技術分享

## 推薦閱讀

**張量（Tensor）：神經網路的基本資料結構**

範赫

**【演算法理論】經典AD-Census:（1）代價計算**

ADCensus演算法原始碼位址：https://github.com/ethan-li-coding/AD-CensusAD-Census演算法來自於中國學者Xing Mei等在ICCV2011發表的論文《On Building an Accurate Stereo...

李迎松 發表於立體視覺理...

**【滴滴】使用TensoFlow進行張量計算**

參考了xinyu chen的知乎文章淺談張量分解（二）：張量分解的數學基礎，用TensorFlow實現張量計算的相關定義。 import tensorflow as tf import numpy as np sess=tf.Session()1. Kronecker ...

王庫辛


**基於深度學習的雙選擇性衰落通道的通道估計**

heu御林軍