

卷積神經網路 (三) ReLU 層



山與水你和我

體系結構小白，影像處理小白

16 人贊同了該文章

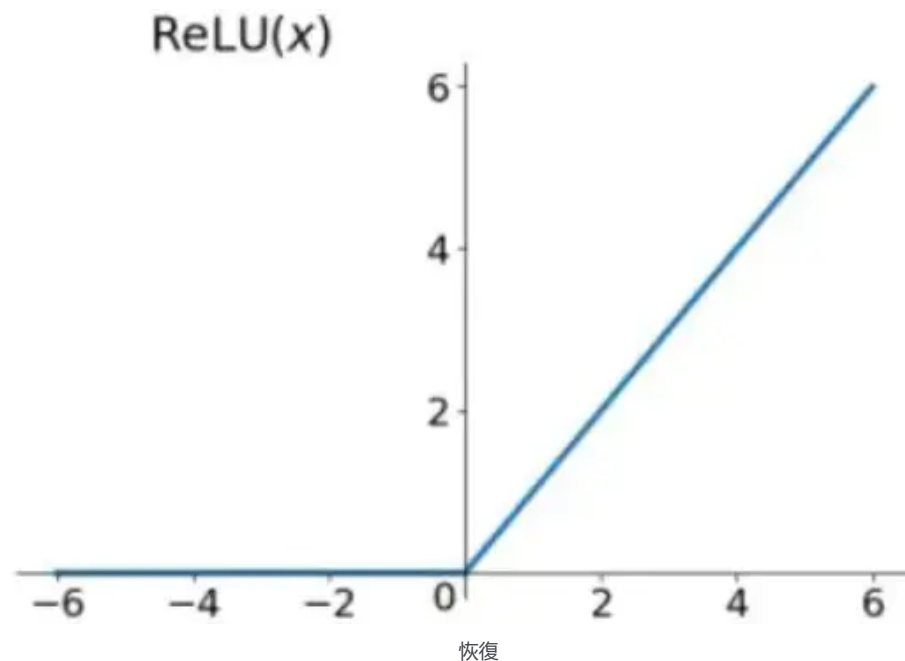
上一篇

[山與水你和我：卷積神經網路 \(二\) 從影像到tensor](#)

完成了從影像到tensor，可以輸入到任意的網路層。

CNN 卷積神經網路一般有Conv 卷積層、ReLU 活化函數層、MaxPool 池化層、Linear 全連接層等。在Pytorch 裡面，寫網路層只需要寫forward 前向過程，而backward 反向傳播和參數更新是自動完成的，Autograd 真是個好東西。本人沒有從自動求導開始寫起，那有點複雜，需要構造圖結構，水平有限只能老實寫簡單的backward。

首先從ReLU 層，最簡單的開始



登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。

立即登入/註冊



$$\text{relu}(x) = \begin{cases} x, & x > 0, \\ 0, & x \leq 0. \end{cases}$$

向前

前向傳播十分簡單，只需要遍歷這個batch 每個tensor, > 0 的保留, <=0 的置為 0

```
std::vector<tensor> ReLU::forward(const std::vector<tensor>& input) {
    // 获取图像信息
    const int batch_size = input.size();
    // 准备输出的 Tensor, 分配空间
    std::vector<tensor> output;
    output.reserve(batch_size);
    for(int b = 0; b < batch_size; ++b)
        output.emplace_back(new Tensor3D(input[b]->C, input[b]->H));
    // 获取每个 Tensor 的数据长度
    const int total_length = input[0]->get_length();
    for(int b = 0; b < batch_size; ++b) {
        data_type* const src_ptr = input[b]->data;    // 输入的第 b 张图像的特征
        data_type* const out_ptr = output[b]->data;   // 输出的第 b 张图像的特征
        // 只保留 > 0 的部分
        for(int i = 0; i < total_length; ++i)
            out_ptr[i] = src_ptr[i] >= 0 ? src_ptr[i] : 0;
    }
    return output;
}
```

落後

ReLU 層的反向十分簡單，因為只有> 0 的部分是有效的，其它置為0 的參與後面的計算也只會得到0，因此只有> 0 的部分有梯度，所以如果需要backward, 需要在forward過程記錄output，在 ReLU 的類別聲明中有

```
std::vector<tensor> output;
```

修改forward

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



```

const int batch_size = input.size();
// 如果是第一次经过这一层
if(output.empty()) {
    // 给输出分配空间
    this->output.reserve(batch_size);
    for(int b = 0; b < batch_size; ++b)
        this->output.emplace_back(new Tensor3D(input[0]->C, input[0]->H, input[0]->W));
}
// 只保留 > 0 的部分
const int total_length = input[0]->get_length();
for(int b = 0; b < batch_size; ++b) {
    data_type* const src_ptr = input[b]->data;
    data_type* const out_ptr = this->output[b]->data;
    for(int i = 0; i < total_length; ++i)
        out_ptr[i] = src_ptr[i] >= 0 ? src_ptr[i] : 0;
}
return this->output;
}

```

這裡有個小技巧，只有第一次經過forward的時候，會給output分配空間，之後都不必每次分配空間，返回tensor指針即可，可以減少消耗。但也存在一個問題，訓練時的batch_size不能更大，只能不變或更小；同理驗證和測試階段的batch_size ≤ 訓練階段的batch_size，邏輯導致的，問題不大，但可以更快。

記錄了每個batch的輸出，這樣一來backward就簡單了，output中≤ 0的地方梯度都為0，> 0的地方梯度保留delta，如下：

```

std::vector<tensor> ReLU::backward(std::vector<tensor>& delta) {
    // 获取信息
    const int batch_size = delta.size();
    // 从这一层的输出中，< 0 的部分过滤掉
    const int total_length = delta[0]->get_length();
    for(int b = 0; b < batch_size; ++b) {
        data_type* src_ptr = delta[b]->data;
        data_type* out_ptr = this->output[b]->data;
        for(int i = 0; i < total_length; ++i)
            src_ptr[i] = out_ptr[i] <= 0 ? 0 : src_ptr[i]; // 输出 > 0 的才有梯度从输出
    }
}

```

登入即可查看超5億專業優質內容

超5千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



這裡也有個細節，**就地修改**，直接在下一層回傳的梯度delta 上修改，作為回傳給上一層的梯度，這是因為relu 簡單，下一層回傳的梯度 δ^i 和回傳給上一層的梯度 δ^{i-1} 大小是一樣的。

驗證

假定輸入，第0 張影像的第3 個通道的內容如下：

-0.728	1.309	1.354	0.025	-0.478	3.618	-0.906
0.379	-1.002	0.213	-0.474	0.002	-0.468	-0.243
0.417	-0.965	-1.180	-0.805	0.425	0.152	-0.164
-0.901	-1.236	-2.054	-0.384	0.738	-0.525	1.012
-1.037	0.992	-0.992	0.780	0.690	-0.500	0.835
0.620	-0.532	0.751	-1.225	2.221	1.129	-1.075
-1.204	1.819	0.664	0.102	-2.001	1.673	-0.180

ReLU 層的輸入

經過ReLU 層過濾的輸出

0.000	<u>1.309</u>	<u>1.354</u>	<u>0.025</u>	0.000	<u>3.618</u>	0.000
<u>0.379</u>	0.000	<u>0.213</u>	0.000	<u>0.002</u>	0.000	0.000
<u>0.417</u>	0.000	0.000	0.000	<u>0.425</u>	<u>0.152</u>	0.000
0.000	0.000	0.000	0.000	<u>0.738</u>	0.000	<u>1.012</u>
0.000	<u>0.992</u>	0.000	<u>0.780</u>	<u>0.690</u>	0.000	<u>0.835</u>
<u>0.620</u>	0.000	<u>0.751</u>	0.000	<u>2.221</u>	<u>1.129</u>	0.000
0.000	<u>1.819</u>	<u>0.664</u>	<u>0.102</u>	0.000	<u>1.673</u>	0.000

ReLU 層的輸出

從ReLU 的下一層回傳的梯度假設為

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。



-0.190	-0.235	1.135	0.815	0.180	0.841	-0.338
0.763	0.005	1.016	-0.563	-0.845	0.487	-0.186
0.285	0.355	0.544	0.394	-0.531	0.596	-0.972
-0.661	-1.283	-0.959	1.424	1.559	0.021	0.755
0.489	-0.553	0.174	0.823	-0.606	-0.576	0.579
0.030	0.727	-0.397	-1.032	-1.972	1.712	-2.014

從下一層回傳的梯度


其中，只有劃紅線的地方（ReLU 輸出> 0），梯度會保留回傳給上一層，如下

0.000	-0.435	0.555	-0.282	0.000	-0.222	0.000
-0.190	0.000	1.135	0.000	0.180	0.000	0.000
0.763	0.000	0.000	0.000	-0.845	0.487	0.000
0.000	0.000	0.000	0.000	-0.531	0.000	-0.972
0.000	-1.283	0.000	1.424	1.559	0.000	0.755
0.489	0.000	0.174	0.000	-0.606	-0.576	0.000
0.000	0.727	-0.397	-1.032	0.000	1.712	0.000

回傳給上一層的梯度

OK，ReLU 層的前向與反向傳播完畢！相關測試程式碼在.....

程式碼

<https://github.com/hermosayhl/CNN>
號  github.com/hermosayhl/CNN

編輯於2022-02-19 19:09

[卷積神經網路 \(CNN\)](#) [深度學習 \(Deep Learning\)](#) [C++](#)

號  贊同16 號  新增評論 號  分享 號  喜歡 號  收藏 號  申請轉載 號...

寫下你的評論...

號  目錄

向前

收起

落後


驗證

程式碼

×

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。





還沒有評論，發表第一個評論吧

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

推薦閱讀

$$\text{Time} \sim O\left(\sum_{l=1}^D M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l\right)$$
$$\text{Space} \sim O\left(\sum_{l=1}^D K_l^2 \cdot C_{l-1} \cdot C_l + \sum_{l=1}^D M_l^2 \cdot C_l\right)$$

卷積神經網路的複雜度分析

麥可·袁

卷積神經網路的平移等變與平移不變

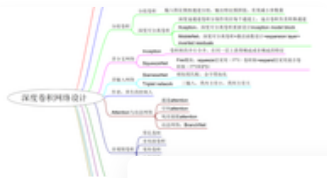
參數共享和局部連接是卷積神經網路最基本的兩個特性，而在平移等變和平移不變的性質上卻需要注意。對於卷積，參數共享的特殊形式使得神經網路層具有平移等變性 (equivariance)。例如，當處...

舒宇

卷積神經網路中的感受野計算 (譯)

本文翻譯自A guide to receptive field arithmetic for Convolutional Neural Networks (可能需要翻牆才能訪問)，方便自己學習和參考。若有侵權，也請告知。感受野 (receptive field) ...

祁崑崙



深度卷積神經

養生的控制...

登入即可查看超5億專業優質內容

超5 千萬創作者的優質提問、專業回答、深度文章和精彩影片盡在知乎。

