卷積神經網路原理及其C++/Opencv實作(8)—手寫數位影像辨識

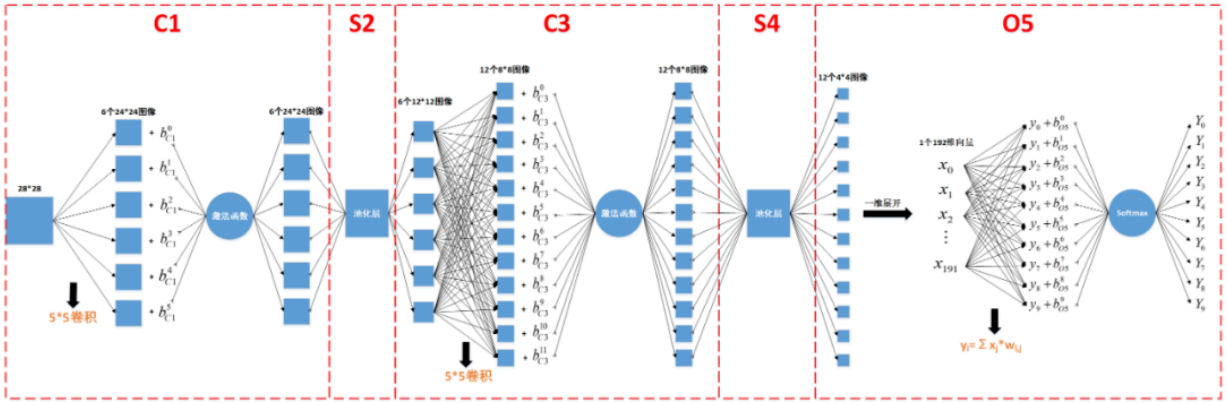
原創 sdff20201029 萌萌噠程序猴 2021-04-03 18:09

本文是本系列的第8篇文章，也是終結篇章。在本文中我們主要講5層卷積神經網路參數更新和訓練的程式碼實現，以及如何使用5層卷積神經網路來實現0~9的手寫數位影像的辨識。

首先還是列出本系列其它博文的超鏈接，方便讀者跳轉查閱：

1. 卷積神經網路原理及其C++/Opencv實作(1)
2. 卷積神經網路原理及其C++/Opencv實作(2)
3. 卷積神經網路原理及其C++/Opencv實作(3)
4. 卷積神經網路原理及其C++/Opencv實作(4)—誤反向傳播法
5. 卷積神經網路原理及其C++/Opencv實作(5)—參數更新
6. 卷積神經網路原理及其C++/Opencv實作(6)—前向傳播程式碼實現
7. 卷積神經網路原理及其C++/Opencv實作(7)—誤反向傳播程式碼實現

下面我們還是分別講5層網路其餘部分的程式碼實作吧~



1. 訓練過程中參數的更新

(1) O5層參數更新

本層需要更新的參數為192*10個權重值，以及10個偏移值。更新公式如下，其中α為學習率，Y為Softmax函數的輸出，t為標籤，x為Affine層的輸入，0≤i<10，0 ≤j < 192。

本層的參數更新程式碼實作如下：

```
void update_full_para(vector<Mat> inputData, CNNOpts opts, OutLayer &O)
{
  int outSize_r = inputData[0].rows;
  int outSize_c = inputData[0].cols;
  Mat OinData(1, outSize_r*outSize_c*inputData.size(), CV_32FC1);
  for (int i = 0; i < inputData.size(); i++)   //12通道
  {
    for (int r = 0; r < outSize_r; r++)  //4
    {
      for (int c = 0; c < outSize_c; c++)   //4
      {
        //把本层输入的12个4*4图像展开成长度为192的一维向量
        OinData.ptr<float>(0)[i*outSize_r*outSize_c + r*outSize_c + c] = inpu
      }
```

```
    }
  }

  for (int j = 0; j < O.outputNum; j++)  //10通道
  {
    for (int i = 0; i < O.inputNum; i++)  //192通道
    {
      //w = w - α*dE/dw
      O.wData.ptr<float>(j)[i] = O.wData.ptr<float>(j)[i] - opts.alpha*O.d.pt
    }
    //b = b - α*dE/db
    O.basicData.ptr<float>(0)[j] = O.basicData.ptr<float>(0)[j] - opts.alpha*
  }
}
```

(2) C3層參數更新

本層需要更新的參數為6*12個5*5卷積核，以及12個偏移值。更新公式如下，其中α為學習率，k為本層的捲積核，b為本層的偏置，Y_{S2}為S2層的輸出，d_{C3}為C3層的局部梯度，sum為求矩陣中所有元素和的操作，0≤i<12，0 ≤j <6 。d_{C3}的計算可參考上篇部落格：

卷積神經網路原理及其C++/Opencv實作(7)—誤反向傳播程式碼實現

本層的參數更新程式碼實作如下：

```
void update_cov_para(vector<Mat> inputData, CNNOpts opts, CovLayer &C)
{
  for (int i = 0; i < C.outChannels; i++)   //6通道
  {
    for (int j = 0; j < C.inChannels; j++)   //1通道
    {
      Mat Cdk = correlation(C.d[i], inputData[j], valid);  //计算YS2*dC3
      Cdk = Cdk*(-opts.alpha);   //矩阵乘以系数-α.dE/dk
      C.mapData[j][i] = C.mapData[j][i] + Cdk;   //计算k = k - α.dE/dk
    }

    float d_sum = (float)cv::sum(C.d[i])[0];   //计算sum(dC3)，这里有6个24*24的
    C.basicData.ptr<float>(0)[i] = C.basicData.ptr<float>(0)[i] - opts.alpha*
  }
}
```

(3) C1層參數更新

本層需要更新的參數為6個5*5卷積核，以及6個偏移值。更新公式如下，其中α為學習率，k為本層的捲積核，b為本層的偏置，I_{C1}為C1層的28*28輸入影像（也即5層網路

的一張28*28輸入影像），d_{C1}為C1層的局部梯度，sum為求矩陣中所有元素和的操作，$0 \le i < 6$ 。d_{C1}的計算也可參考上篇部落格文章。

卷積神經網路原理及其C++/Opencv實作(7)—誤反向傳播程式碼實現

由於本層的參數更新程式碼操作與C3層一樣，只是輸入、輸出參數不一樣而已，因此本層的參數更新也可以呼叫上述update_cov_para函數來實現。

(4) 所有參數的更新

綜上，C1、C3、O5層的參數更新程式碼如下，其中inputdata為5層網路的單張28*28手寫數位影像。

```
void cnnapplygrads(CNN &cnn, CNNOpts opts, Mat inputData) // 更新权重
{
  vector<Mat> input_tmp;
  input_tmp.push_back(inputData);

  update_cov_para(input_tmp, opts, cnn.C1);

  update_cov_para(cnn.S2.y, opts, cnn.C3);

```

```
  update_full_para(cnn.S4.y, opts, cnn.O5);
}
```

2. 訓練過程中參數的清零

由於訓練是多輪迭代的過程，且訓練時會有參數累積的操作，下一輪訓練開始前需要將參數清除，否則累加操作會出問題。

```
//清零卷积层的参数
void clear_cov_mid_para(CovLayer &C)
{
  int row = C.d[0].rows;
  int col = C.d[0].cols;
  for (int j = 0; j < C.outChannels; j++)
  {
    for (int r = 0; r < row; r++)
    {
      for (int c = 0; c < col; c++)
      {
        C.d[j].ptr<float>(r)[c] = 0.0;
        C.v[j].ptr<float>(r)[c] = 0.0;
        C.y[j].ptr<float>(r)[c] = 0.0;
      }
    }
  }
}

//清零池化层的参数
void clear_pool_mid_para(PoolLayer &S)
{
```

```
  int row = S.d[0].rows;
  int col = S.d[0].cols;
  for (int j = 0; j < S.outChannels; j++)
  {
    for (int r = 0; r < row; r++)
    {
      for (int c = 0; c < col; c++)
      {
        S.d[j].ptr<float>(r)[c] = 0.0;
        S.y[j].ptr<float>(r)[c] = 0.0;
      }
    }
  }
}

//清零输出层的参数
void clear_out_mid_para(OutLayer &O)
{
  for (int j = 0; j < O.outputNum; j++)
  {
    O.d.ptr<float>(0)[j] = 0.0;
    O.v.ptr<float>(0)[j] = 0.0;
    O.y.ptr<float>(0)[j] = 0.0;
  }
}

//调用上述函数实现5层网络的参数清零
void cnnclear(CNN &cnn)
{
  clear_cov_mid_para(cnn.C1);
```

```
53    clear_pool_mid_para(cnn.S2);
54    clear_cov_mid_para(cnn.C3);
55    clear_pool_mid_para(cnn.S4);
56    clear_out_mid_para(cnn.O5);
57  }
```

2. 手寫數位影像的讀取

從網路上下載的手寫數位影像，是gz壓縮文件，需要將其解壓縮：

解壓縮gz檔案之後得到以下4個對應文件，其中train-images.idx3-ubyte為訓練資料文件，train-labels.idx1-ubyte為訓練資料的標籤文件，t10k-images.idx3-ubyte為測試資料文件， t10k-labels.idx1-ubyte為測試資料的標籤檔。

(1) 訓練資料檔與測試資料檔的格式如下圖所示：

文件格式：該區域的4個位元組數據組成一個int數據，如果該int數據為2051，表示該文件是圖像文件，如果是2049表示該文件是文字文件。因此對於訓練資料和測試資料文件，本區域的值為2051。

總圖像數：該區域的4個位元組數據組成一個int數據，該int數據為文件中包含的圖像總數。

圖像行數：該區域的4個位元組數據組成一個int數據，該int數據為每張圖像的行數。

圖像列數：該區域的4個位元組數據組成一個int數據，該int數據為每張圖像的列數。

要注意的是，如果運行程式的處理器是英特爾處理器，需要把讀到的4個位元組數據按相反順序排序，再組成int數據，例如首先我們讀取到的int數據由byte0、byte1、byte2、byte3這4個位元組資料組成（<<為左移運算）：

d=(byte3 << 24)) + (byte2 << 16) + (byte1 << 8) + byte0

那就需要把4個數據依照相反順序排序，重新組成int數據，這個重組的int數據才是我們想要的數據：

d'=(byte0 << 24)) + (byte1 << 16) + (byte2 << 8) + byte3

根據上述格式，訓練資料檔案與測試資料檔案的讀取程式碼如下，我們將同一個檔案中的多張影像都讀成Opencv的Mat格式，然後將多個Mat格式影像儲存進vector陣列中：

```
//将int数据中的4个字节数据按相反顺序重新排列，重组成一个int数据
int ReverseInt(int i)
{
  unsigned char ch1, ch2, ch3, ch4;
  ch1 = i & 0xff;
  ch2 = (i >> 8) & 0xff;
  ch3 = (i >> 16) & 0xff;
  ch4 = (i >> 24) & 0xff;

  return ((int)(ch1 << 24)) + ((int)(ch2 << 16)) + ((int)(ch3 << 8)) + (int)(
}
```

```
vector<Mat> read_Img_to_Mat(const char* filename)
{
  FILE  *fp = NULL;
  fp = fopen(filename, "rb");
  if (fp == NULL)
    printf("open file failed\n");
  assert(fp);

  int magic_number = 0;
  int number_of_images = 0;
  int n_rows = 0;
  int n_cols = 0;

  fread(&magic_number, sizeof(int), 1, fp);   //从文件中读取sizeof(int) 个字符到
  magic_number = ReverseInt(magic_number);

  fread(&number_of_images, sizeof(int), 1, fp);   //获取训练或测试image的个数nu
  number_of_images = ReverseInt(number_of_images);

  fread(&n_rows, sizeof(int), 1, fp);   //获取训练或测试图像的高度Heigh
  n_rows = ReverseInt(n_rows);

  fread(&n_cols, sizeof(int), 1, fp);   //获取训练或测试图像的宽度Width
  n_cols = ReverseInt(n_cols);


  //获取第i幅图像，保存到vec中
  int i, r, c;

  int img_size = n_rows*n_cols;
```

```
  vector<Mat> img_list;
  for (i = 0; i < number_of_images; ++i)
  {
    Mat tmp(n_rows, n_cols, CV_8UC1);
    fread(tmp.data, sizeof(uchar), img_size, fp);  //读取一张图像
    tmp.convertTo(tmp, CV_32F);   //将图像转换为float数据
    tmp = tmp / 255.0;   //将数据转换成0~1的数据
    img_list.push_back(tmp.clone());
  }

  fclose(fp);
  return img_list;
}
```

(2) 標籤文件的格式如下圖所示：

文件格式：該區域的4個位元組數據組成一個int數據，如果該int數據為2051，表示該文件是圖像文件，如果是2049表示該文件是文字文件。標籤文件屬於文字文件，因此本區域的值為2049。

總圖像數：該區域的4個位元組數據組成一個int數據，該int數據為文件中包含的圖像總數。

如果執行程式的處理器為英特爾處理器，同樣需要把讀到的4個位元組資料依相反順序排序，再重組成int資料。

每張影像表示的數字為0~9中的一個數字，因此影像標籤就是0~9之中的一個數字，且該數字與影像表示的數字相對應。

由於卷積神經網路使用的是"one-hot"碼，因此我們需要把0~9的標籤數字轉換成"one-hot"碼:

0 --> 1 0 0 0 0 0 0 0 0 0
1 --> 0 1 0 0 0 0 0 0 0 0
2 --> 0 0 1 0 0 0 0 0 0 0
3 --> 0 0 0 1 0 0 0 0 0 0
4 --> 0 0 0 0 1 0 0 0 0 0
5 --> 0 0 0 0 0 1 0 0 0 0
6 --> 0 0 0 0 0 0 1 0 0 0
7 --> 0 0 0 0 0 0 0 1 0 0
8 --> 0 0 0 0 0 0 0 0 1 0
9 --> 0 0 0 0 0 0 0 0 0 1

根據上述格式，標籤檔案的讀取代碼如下，我們將同一個標籤檔案中的每個標籤數字轉換成"one-hot"碼，然後再將"one-hot"碼儲存到一個1行10列的Mat結構當中，再將Mat儲存到vector中：

```
vector<Mat> read_Lable_to_Mat(const char* filename)
{
  FILE  *fp = NULL;
  fp = fopen(filename, "rb");
  if (fp == NULL)
    printf("open file failed\n");
  assert(fp);

  int magic_number = 0;
  int number_of_labels = 0;
```

```
    int label_long = 10;


    fread(&magic_number, sizeof(int), 1, fp);   //从文件中读取sizeof(magic_numbe
    magic_number = ReverseInt(magic_number);

    fread(&number_of_labels, sizeof(int), 1, fp);   //获取训练或测试image的个数nu
    number_of_labels = ReverseInt(number_of_labels);

    int i, l;

    vector<Mat> label_list;

    for (i = 0; i < number_of_labels; ++i)
    {

      Mat tmp = Mat::zeros(1, label_long, CV_32FC1);
      unsigned char temp = 0;
      fread(&temp, sizeof(unsigned char), 1, fp);
      tmp.ptr<float>(0)[(int)temp] = 1.0;  //将0~9的数字转换成one-hot码
      label_list.push_back(tmp.clone());
    }

    fclose(fp);
    return label_list;
}
```

3 . 訓練過程的實現代碼

```
void cnntrain(CNN &cnn, vector<Mat> inputData, vector<Mat> outputData, CNNOpt
{
  // 学习训练误差曲线，记录交叉熵误差函数的值
  cnn.L = Mat(1, trainNum, CV_32FC1).clone();
  for (int e = 0;  e < opts.numepochs; e++)   //opts.numepochs表示需要训练次数
  {
    for (int n = 0; n < trainNum; n++)   //trainNum表示由多少张图片，训练完这些图
    {
      //学习率递减0.03~0.001
      opts.alpha = 0.03 - 0.029*n / (trainNum - 1);

      cnnff(cnn, inputData[n]);   // 前向传播
      cnnbp(cnn, outputData[n]); // 后向传播
      cnnapplygrads(cnn, opts, inputData[n]); // 更新参数

      // 计算交叉熵误差函数的值
      float l = 0.0;
      for (int i = 0; i < cnn.O5.outputNum; i++)
      {
        l = l - outputData[n].ptr<float>(0)[i] * log(cnn.O5.y.ptr<float>(0)[i
      }
      cnn.L.ptr<float>(0)[n] = l;

      cnnclear(cnn);   //清零参数

      printf("n=%d, f=%f, α=%f\n", n, cnn.L.ptr<float>(0)[n], opts.alpha);
    }
  }

```

```
}
```

4.對手寫數位影像分類的實現代碼

```
//1行n列的向量
int vecmaxIndex(Mat vec)  //返回向量最大数的序号
{
  int veclength = vec.cols;
  float maxnum = -1.0;
  int maxIndex = 0;

  float *p = vec.ptr<float>(0);
  for(int i=0; i < veclength; i++)
  {
    if(maxnum < p[i])
    {
      maxnum = p[i];
      maxIndex = i;
    }
  }
  return maxIndex;
}

//测试函数
float cnntest(CNN cnn, vector<Mat> inputData, vector<Mat> outputData)
{
  int incorrectnum = 0;  //错误预测的数目
  for (int i = 0; i < inputData.size(); i++)  //inputData.size()为测试图像的总
```

```
  {
    cnnff(cnn, inputData[i]);   //前向传播
    //检查神经网络输出的最大概率的序号是否等于标签中1值的序号，如果等于则表示分类成功
    if (vecmaxIndex(cnn.O5.y) != vecmaxIndex(outputData[i]))
    {
      incorrectnum++;
      printf("i = %d, 识别失败\n", i);
    }
    else
    {
      printf("i = %d, 识别成功\n", i);
    }
    cnnclear(cnn);
  }
  printf("incorrectnum=%d\n", incorrectnum);
  printf("inputData.size()=%d\n", inputData.size());
  return (float)incorrectnum / (float)inputData.size();
}
```

5．總體測試的實作程式碼

下列函數就是5層網路的測試程式碼，在mian函數中呼叫。

```
void minst_cnn_test(void)
{
  vector<Mat> traindata_list;
  vector<Mat> traindata_label;
  vector<Mat> testdata_list;
  vector<Mat> testdata_label;
```

```
7
8     //读取训练数据标签
9     traindata_label = read_Lable_to_Mat("Minst/train-labels.idx1-ubyte");
10    //读取训练数据
11    traindata_list = read_Img_to_Mat("Minst/train-images.idx3-ubyte");
12    //读取测试数据标签
13    testdata_label = read_Lable_to_Mat("Minst/t10k-labels.idx1-ubyte");
14    //读取测试数据
15    testdata_list = read_Img_to_Mat("Minst/t10k-images.idx3-ubyte");
16
17    int train_num = traindata_list.size();
18    int test_num = testdata_list.size();
19    int outSize = testdata_label[0].cols;
20
21    int row = traindata_list[0].rows;
22    int col = traindata_list[0].cols;
23
24    CNNOpts opts;
25    opts.numepochs = 1;
26    opts.alpha = 0.03;   //学习率初始值
27    int trainNum = 60000;
28
29    CNN cnn;
30    cnnsetup(cnn, row, col, outSize);   //cnn初始化
31    cnntrain(cnn, traindata_list, traindata_label, opts, train_num);  //训练
32
33    float success = cnntest(cnn, testdata_list, testdata_label);   //分类
34    printf("success=%f\n", 1 - success);   //打印分类的成功率
35 }
```

執行以上函數對5層網路進行手寫數位影像的訓練與分類測試，得到的結果如下，對10000張影像進行分類，分類失敗170張，準確率達98.3%，還是相當高的。

本系列的基於VS2015與Opencv3.4.1的完整程式碼工程，讀者可在以下網址下載：
https://download.csdn.net/download/shandianfengfan/16392246

好了，本系列的文章就更新到這裡啦，有人可能會說我重複造輪子沒有意義，我倒不這麼認為，因為這是一個學習的過程，自己去實現一遍會加深自己的理解。在深度理解之後，再去使用別人現成的深度學習框架，也會順手得多。接下來的文章我們就不自己實現網路了，而是使用別人現成的深度學習框架，我們把主要精力放在網路的建構與訓練模型的建構上面。

歡迎掃碼追蹤以下微信公眾號，接下來會不定時更新更加精彩的內容噢～

人工智慧 27　深度學習 26　機器學習 33　C++ 70　Opencv 50

人工智慧 目錄

上一篇
卷積神經網路原理及其C++/Opencv實作(7)—誤反向傳播程式碼實現

下一篇
win10+VS2017+Opencv3.4.1+libtorch開發環境建置(1)

閱讀原文

喜歡此內容的人還喜歡