网络资源是无限的

# tiny-cnn执行过程分析(MNIST)

2016-01-31 17:42 3565人阅读 评论(2) 收藏 举报

分类： Caffe（19） Deep Learning（8） Neural Network（12）

在http://blog.csdn.net/fengbingchun/article/details/50573841中以MNIST为例对tiny-cnn的使用进行了介绍，下面对其执行过程进行分析：

支持两种损失函数：(1)、mean squared error(均方差)；(2)、cross entropy(交叉熵)。在MNIST中使用的是mean squared error，代码段：

[cpp]

```cpp
// mean-squared-error loss function for regression
class mse {
public:
    static float_t f(float_t y, float_t t) {
        return (y - t) * (y - t) / 2;
    }

    static float_t df(float_t y, float_t t) {
        return y - t;
    }
};
```

支持六种激活函数：(1)、tanh；(2)、sigmoid；(3)、softmax；(4)、rectifiedlinear(relu)；(5)、leaky relu；(6)、identity。MNIST中使用的是tanh，代码段：

[cpp]

```cpp
class tan_h : public function {
public:
    float_t f(const vec_t& v, size_t i) const override {
        const float_t ep = std::exp(v[i]);
        const float_t em = std::exp(-v[i]);
        return (ep - em) / (ep + em);
    }

    // fast approximation of tanh (improve 2-3% speed in LeNet-5)
    /*float_t f(float_t x) const {
        const float_t x2 = x * x;
        x *= 1.0 + x2 * (0.1653 + x2 * 0.0097);
        return x / std::sqrt(1.0 + x * x);// invsqrt(static_cast<float>(1.0 + x * x));
    }*/

    float_t df(float_t y) const override { return 1.0 - sqr(y); }
    std::pair<float_t, float_t> scale() const override { return std::make_pair(-0.8, 0.8)
```

设计CNN结构，用于MNIST，与LeNet-5结构相似，去除了F6层：

输入层Input：图像大小32*32，神经元数量32*32=1024，代码段：

- HTML (3)
- Image Recognition (8)
- Image Processing (18)
- Image Registration (13)
- ImageMagick (3)
- Java (5)
- Linux (20)
- Log (2)
- Makefile (2)
- Mathematical Knowledge (6)
- Multi-thread (4)
- Matlab (33)
- MFC (8)
- MinGW (3)
- Mac (1)
- Neural Network (13)
- OCR (9)
- Office (2)
- OpenCL (2)
- OpenSSL (7)
- OpenCV (86)
- OpenGL (2)
- OpenGL ES (3)
- OpenMP (3)
- Photoshop (1)
- Python (4)
- Qt (1)
- SIMD (14)
- Software Development (4)
- System architecture (2)
- Skia (1)
- SVN (1)
- Software Testing (4)
- Shell (2)
- Socket (3)
- Target Detection (2)
- Target Tracking (2)
- VC6 (6)
- VS2008 (16)
- VS2010 (4)
- VS2013 (3)
- vigra (2)
- VLC (5)
- VLFeat (1)
- wxWidgets (1)
- Watermark (4)
- Windows7 (6)
- Windows Core Programming (9)
- XML (2)

**Free Codes**

- pudn
- freecode
- Peter's Functions
- CodeProject
- SourceCodeOnline
- Computer Vision Source Code
- Codesoso
- Digital Watermarking
- SourceForge
- HackChina
- oschina

[cpp]

```cpp
const int width = header.num_cols + 2 * x_padding;
const int height = header.num_rows + 2 * y_padding;

std::vector<uint8_t> image_vec(header.num_rows * header.num_cols);

ifs.read((char*) &image_vec[0], header.num_rows * header.num_cols);

dst.resize(width * height, scale_min);

for (size_t y = 0; y < header.num_rows; y++)
  for (size_t x = 0; x < header.num_cols; x++)
    dst[width * (y + y_padding) + x + x_padding]
    = (image_vec[y * header.num_cols + x] / 255.0) * (scale_max - scale_min) + scale_min;
```

C1层：卷积窗大小5*5，输出特征图数量6，卷积窗种类6，输出特征图大小28*28，可训练参数5*5*6+6=156，神经元数量28*28*6=4704；

S2层：卷积窗大小2*2，输出下采样图数量6，卷积窗种类6，输出下采样图大小14*14，可训练参数1*6+6=12，神经元数量14*14*6=1176；

C3层：卷积窗大小5*5，输出特征图数量16，卷积窗种类16，输出特征图大小10*10，可训练参数6*16*5*5+16=2416，神经元数量10*10*16=1600；

S4层：卷积窗大小2*2，输出下采样图数量16，卷积窗种类16，输出下采样图大小5*5，可训练参数1*16+16=32，神经元数量5*5*16=400；

C5层：卷积窗大小5*5，输出特征图数量120，卷积窗种类120，输出特征图大小1*1，可训练参数5*5*16*120+120=48120，神经元数量1*120=120；

输出层Output：输出特征图数量10，卷积窗种类10，输出特征图大小1*1，可训练参数120*10+10=1210，神经元数量1*10=10。

原有MNIST图像大小为28*28，此处为32*32，上下左右各填补2个像素，填补的像素取值为-1，其它像素取值范围为[-1,1]。

权值和阈值(偏置)初始化：权值采用均匀随机数产生，阈值均赋0。

C1层权值，初始化范围[sqrt(6.0/(25+150)), sqrt(6.0/(25+150))]；

S2层权值，初始化范围[sqrt(6.0/(4+1)), - sqrt(6.0/(4+1))]；

C3层权值，初始化范围[sqrt(6.0/(150+400)), - sqrt(6.0/(150+400))]；

S4层权值，初始化范围[sqrt(6.0/(4+1)), - sqrt(6.0/(4+1))]；

C5层权值，初始化范围[sqrt(6.0/(400+3000)), - sqrt(6.0/(400+3000))]；

输出层权值，初始化范围[sqrt(6.0/(120+10)),  -sqrt(6.0/(120+10))]。

前向传播：

C1层代码段：

[cpp]

```cpp
vec_t &a = a_[worker_index]; // w*x
vec_t &out = output_[worker_index]; // output
const vec_t &in = *(prev_out_padded_[worker_index]); // input

std::fill(a.begin(), a.end(), (float_t)0.0);

for_i(parallelize_, out_.depth_, [&](int o) {
    for (layer_size_t inc = 0; inc < in_.depth_; inc++) {
        if (!tbl_.is_connected(o, inc)) continue;

        const float_t *pw = &this->W_[weight_.get_index(0, 0, in_.depth_ * o + inc)];
        const float_t *pi = &in[in_padded_.get_index(0, 0, inc)];
        float_t *pa = &a[out_.get_index(0, 0, o)];

        for (layer_size_t y = 0; y < out_.height_; y++) {
```

关闭

libsvm
joys99
CodeForge
cvchina
tesseract-ocr
sift
TiRG
imgSeek
OpenSURF

**Friendly Link**

OpenCL
Python
poesia-filter
TortoiseSVN
imgSeek
Notepad
Beyond Compare
CMake
VIGRA
CodeGuru
vchome
aforgenet
CVLAB
Doxygen
Coursera
OpenMP

**Technical Forum**

Matlab China
OpenCV China
The CImg Library
Open Computer Vision Library
CxImage
ImageMagick
ImageMagick China
OpenCV_China
Subversion China

```cpp
            for (layer_size_t x = 0; x < out_.width_; x++) {
                const float_t * ppw = pw;
                const float_t * ppi = pi + (y * h_stride_) * in_padded_.width_ + x * w_st
                float_t sum = (float_t)0.0;

                // should be optimized for small kernel(3x3,5x5)
                for (layer_size_t wy = 0; wy < weight_.height_; wy++) {
                    for (layer_size_t wx = 0; wx < weight_.width_; wx++) {
                        sum += *ppw++ * ppi[wy * in_padded_.width_ + wx];
                    }
                }
                pa[y * out_.width_ + x] += sum;
            }
        }
    }

    if (!this->b_.empty()) {
        float_t *pa = &a[out_.get_index(0, 0, o)];
        float_t b = this->b_[o];
        std::for_each(pa, pa + out_.width_ * out_.height_, [&](float_t& f) { f += b; });
    }
});

for_i(parallelize_, out_size_, [&](int i) {
    out[i] = h_.f(a, i);
});
```

S2层代码段：

[cpp]

```cpp
vec_t& a = a_[index];

for_i(parallelize_, out_size_, [&](int i) {
    const wi_connections& connections = out2wi_[i];

    a[i] = 0.0;

    for (auto connection : connections)// 13.1%
        a[i] += W_[connection.first] * in[connection.second]; // 3.2%

    a[i] *= scale_factor_;
    a[i] += b_[out2bias_[i]];
});

for_i(parallelize_, out_size_, [&](int i) {
    output_[index][i] = h_.f(a, i);
});
```

C3层、C5层代码段与C1层相同。

S4层代码段与S2层相同。

输出层代码段：

[cpp]

```cpp
vec_t &a = a_[index];
vec_t &out = output_[index];

for_i(parallelize_, out_size_, [&](int i) {
    a[i] = 0.0;
    for (layer_size_t c = 0; c < in_size_; c++) {
        a[i] += W_[c*out_size_ + i] * in[c];
    }

    if (has_bias_)
        a[i] += b_[i];
});

for_i(parallelize_, out_size_, [&](int i) {
    out[i] = h_.f(a, i);
});
```

关闭

反向传播：

**Technical Blog**

邹宇华
深之JohnChen
HUNNISH
周伟明
superdont
carson2005
OpenHero
Netman(Linux)
wqvbjhc
yang_xian521
gnuhpc
gnuhpc
千里8848
CVART
tornadomeet
gotosuc
onezeros
hellogv
abcjennifer
crzy_sparrow

评论排行

| | |
|---|---|
| Windows7 32位机上，O | (120) |
| tiny-cnn开源库的使用(MI | (93) |
| Ubuntu 14.04 64位机上不 | (89) |
| tesseract-ocr3.02字符识 | (63) |
| Windows7上使用VS2013 | (47) |
| tesseract-ocr | (42) |
| 图像配准算法 | (41) |
| Windows 7 64位机上Ope | (36) |
| OpenCV中resize函数五利 | (34) |
| 小波矩特征提取matlab代 | (30) |

最新评论

Tesseract-OCR 3.04在Windows
fengbingchun: @ilikede:没有密码，那个commit只是提示是从哪个commit fork过来的，无需管那个

Tesseract-OCR 3.04在Windows
ilikede: 问一下，你第一句中的commit的那个密码，怎么用啊

卷积神经网络(CNN)的简单实现(
fengbingchun: @hugl950123:是需要opencv的支持，你在本地opencv的环境配好了吗，配好了就应该没...

卷积神经网络(CNN)的简单实现(
hugl950123: @fengbingchun:博主请问一下，test_CNN_predict()函数是不是需要open...

卷积神经网络(CNN)的简单实现(
hugl950123: @fengbingchun:博主请问一下，test_CNN_predict()函数是不是需要open...

卷积神经网络(CNN)的简单实现(
hugl950123: @fengbingchun:谢谢，能够成功运行了现在

卷积神经网络(CNN)的简单实现(
fengbingchun: @hugl950123:NN中一共有四个工程，它们之间没有任何关系，都是独立的，如果要运行这篇文章的...

输出层代码段：

[cpp]

```cpp
vec_t delta(out_dim());
const activation::function& h = layers_.tail()->activation_function();

if (is_canonical_link(h)) {
    for_i(out_dim(), [&](int i){ delta[i] = out[i] - t[i]; });
} else {
    vec_t dE_dy = gradient<E>(out, t);

    // delta = dE/da = (dE/dy) * (dy/da)
    for (size_t i = 0; i < out_dim(); i++) {
        vec_t dy_da = h.df(out, i);
        delta[i] = vectorize::dot(&dE_dy[0], &dy_da[0], out_dim());
    }
}
```

C5层代码段：

[cpp]

```cpp
const vec_t& prev_out = prev_->output(index);
const activation::function& prev_h = prev_->activation_function();
vec_t& prev_delta = prev_delta_[index];
vec_t& dW = dW_[index];
vec_t& db = db_[index];

for (layer_size_t c = 0; c < this->in_size_; c++) {
    // propagate delta to previous layer
    // prev_delta[c] += current_delta[r] * W_[c * out_size_ + r]
    prev_delta[c] = vectorize::dot(&curr_delta[0], &W_[c*out_size_], out_size_);
    prev_delta[c] *= prev_h.df(prev_out[c]);
}

for_(parallelize_, 0, (size_t)out_size_, [&](const blocked_range& r) {
    // accumulate weight-step using delta
    // dW[c * out_size + i] += current_delta[i] * prev_out[c]
    for (layer_size_t c = 0; c < in_size_; c++)
        vectorize::muladd(&curr_delta[r.begin()], prev_out[c], r.end() - r.begin(), &dW[c

    if (has_bias_) {
        for (int i = r.begin(); i < r.end(); i++)
            db[i] += curr_delta[i];
    }
});
```

S4层代码段：

[cpp]

```cpp
const vec_t& prev_out = *(prev_out_padded_[index]);
const activation::function& prev_h = prev_->activation_function();
vec_t* prev_delta = (pad_type_ == padding::same) ? &prev_delta_padded_[index] : &prev_del
vec_t& dW = dW_[index];
vec_t& db = db_[index];

std::fill(prev_delta->begin(), prev_delta->end(), (float_t)0.0);

// propagate delta to previous layer
for_i(in_.depth_, [&](int inc) {
    for (layer_size_t outc = 0; outc < out_.depth_; outc++) {
        if (!tbl_.is_connected(outc, inc)) continue;

        const float_t *pw = &this->W_[weight_.get_index(0, 0, in_.depth_ * outc + inc)];
        const float_t *pdelta_src = &curr_delta[out_.get_index(0, 0, outc)];
        float_t *pdelta_dst = &(*prev_delta)[in_padded_.get_index(0, 0, inc)];

        for (layer_size_t y = 0; y < out_.height_; y++) {
            for (layer_size_t x = 0; x < out_.width_; x++) {
                const float_t * ppw = pw;
                const float_t ppdelta_src = pdelta_src[y * out_.width_ + x];
                float_t * ppdelta_dst = pdelta_dst + y * h_stride_ * in_padded_.width_ +

                for (layer_size_t wy = 0; wy < weight_.height_; wy++) {
                    for (layer_size_t wx = 0; wx < weight_.width_; wx++) {
                        ppdelta_dst[wy * in_padded_.width_ + wx] += *ppw++ * ppdelta_src;
```

关闭

卷积神经网络(CNN)的简单实现(
hugl950123: @fengbingchun:下的是新的，我在CNN.cpp文件中每个函数都设置了断点，还是没有变化=...

卷积神经网络(CNN)的简单实现(
fengbingchun: @hugl950123:你用的是GitHub上最新的吗？既然能编译过，在Debug下设断点，应该很快...

卷积神经网络(CNN)的简单实现(
hugl950123: 博主，请问我按照您的代码成功编译后执行结果窗口一闪而过，并且里面什么内容也没有，应该如何解决，能不能...

阅读排行

| | |
|---|---|
| C#中OpenFileDialog的( | (47141) |
| tesseract-ocr3.02字符识 | (34575) |
| 举例说明使用MATLAB C | (25987) |
| OpenCV中resize函数五 | (24317) |
| 利用cvMinAreaRect2求 | (24277) |
| Windows 7 64位机上搭 | (22586) |
| opencv 检测直线、线段、 | (20776) |
| OpenCV运动检测跟踪(b | (20475) |
| 图像配准算法 | (19237) |
| 有效的rtsp流媒体测试地 | (19143) |

文章存档

| | |
|---|---|
| 2017年01月 | (18) |
| 2016年12月 | (11) |
| 2016年11月 | (8) |
| 2016年10月 | (7) |
| 2016年09月 | (16) |

展开

```cpp
                    }
                }
            }
        }
    }
});

for_i(parallelize_, in_padded_.size(), [&](int i) {
    (*prev_delta)[i] *= prev_h.df(prev_out[i]);
});

// accumulate dw
for_i(in_.depth_, [&](int inc) {
    for (layer_size_t outc = 0; outc < out_.depth_; outc++) {

        if (!tbl_.is_connected(outc, inc)) continue;

        for (layer_size_t wy = 0; wy < weight_.height_; wy++) {
            for (layer_size_t wx = 0; wx < weight_.width_; wx++) {
                float_t dst = 0.0;
                const float_t * prevo = &prev_out[in_padded_.get_index(wx, wy, inc)];
                const float_t * delta = &curr_delta[out_.get_index(0, 0, outc)];

                for (layer_size_t y = 0; y < out_.height_; y++) {
                    dst += vectorize::dot(prevo + y * in_padded_.width_, delta + y * out_
                }
                dW[weight_.get_index(wx, wy, in_.depth_ * outc + inc)] += dst;
            }
        }
    }
});

// accumulate db
if (!db.empty()) {
    for (layer_size_t outc = 0; outc < out_.depth_; outc++) {
        const float_t *delta = &curr_delta[out_.get_index(0, 0, outc)];
        db[outc] += std::accumulate(delta, delta + out_.width_ * out_.height_, (float_t)0
    }
}
```

C3层代码段：

[cpp]

```cpp
const vec_t& prev_out = prev_->output(index);
const activation::function& prev_h = prev_->activation_function();
vec_t& prev_delta = prev_delta_[index];

for_(parallelize_, 0, (size_t)in_size_, [&](const blocked_range& r) {
    for (int i = r.begin(); i != r.end(); i++) {
        const wo_connections& connections = in2wo_[i];
        float_t delta = 0.0;

        for (auto connection : connections)
            delta += W_[connection.first] * current_delta[connection.second]; // 40.6%

        prev_delta[i] = delta * scale_factor_ * prev_h.df(prev_out[i]); // 2.1%
    }
});

for_(parallelize_, 0, weight2io_.size(), [&](const blocked_range& r) {
    for (int i = r.begin(); i < r.end(); i++) {
        const io_connections& connections = weight2io_[i];
        float_t diff = 0.0;

        for (auto connection : connections) // 11.9%
            diff += prev_out[connection.first] * current_delta[connection.second];

        dW_[index][i] += diff * scale_factor_;
    }
});

for (size_t i = 0; i < bias2out_.size(); i++) {
    const std::vector<layer_size_t>& outs = bias2out_[i];
    float_t diff = 0.0;

    for (auto o : outs)
        diff += current_delta[o];

```

关闭

```cpp
    db_[index][i] += diff;
}
```

S2层、输入层代码段与S4层相同。

C1层代码段与C3层相同。

权值和偏置更新代码段：

[cpp]

```cpp
void update(const vec_t& dW, const vec_t& /*Hessian*/, vec_t &W) {
    vec_t& g = get<0>(W);

    for_i(W.size(), [&](int i) {
        g[i] += dW[i] * dW[i];
        W[i] -= alpha * dW[i] / (std::sqrt(g[i]) + eps);
    });
}
```

对MNIST中的60000个训练样本，依次执行上面的操作，并更新权值和偏置。

每此循环执行完60000个训练样本，会对10000个测试样本，进行测试，获得识别率。

共迭代30次，然后将最终的权值、偏置等相关参数保持到指定的文件中。

Android入门必备之C++快速入门　　MNIST手写数字的识别CNN篇

查看评论

1楼 张骞晖2 2016-05-02 20:58发表

迭代次数怎么改？

Re: fengbingchun 2016-05-03 08:29发表

回复u013139259：迭代次数是由上层控制的。可参考http://blog.csdn.net/fengbingchun/article/details/50573841 ，由int num_epochs = 30;指定迭代次数

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场