|  |  |
| --- | --- |
| Insight Face算法解构报告 | 文件类型：技术文档 |
| 文件编号： |
| 面向的部门： 研发中心 |
| 保密等级：高 |
| 作者： 刘凯 |
| 日期：2019-01-17 |
| 版本：1.0V |

Insight Face算法解构报告



目 录

[1 Resnet34网络结构 3](#_Toc534986553)

[1.1 PreProcess 3](#_Toc534986554)

[1.2 Stage1 4](#_Toc534986555)

[1.3 Stage2 5](#_Toc534986556)

[1.4 Stage3 6](#_Toc534986557)

[1.5 Stage4 6](#_Toc534986558)

[1.6 fc1 7](#_Toc534986559)

[参考文献 1](#_Toc534986560)

InsightFace的核心是最后的Loss函数，具体可参考之前的《深度人脸识别调研》一文的附录，但前向网络的复现主要跟网络结构有关，本文主要解析项目中使用的ResNet34网络的详细结构。

# Resnet34网络结构

ResNet34的完整网络结构如图1所示：

Data

PreProcess

Stage1

Stage2

Stage3

Stage4

fc1

图1：ResNet34网络结构

具体而言：

Data：输入图片，维度为(N, C, H, W)=(1, 3, 112, 112)，四个维度分别为输入图片数量、通道数、高度和宽度。

PreProcess：图片进入残差网络前的一些预处理，包括减均值、归一化、Convolution, Batch Norm, Leaky Relu几个模块。

Stage1-Stage4：ResNet的4个Stage。

fc1：经过一个全连接层输出512维的特征矢量。

下面对上述模块详细解析。

* 1. PreProcess

这里有5层：minusscalar，mulscalar，conv0，bn0和prelu0。

minusscalar层的输入是维度为(N, C, H, W)的Data，操作是将Data减去127.5，输出维度与Data相同。

mulscalar层的输入是minusscalar层的输出，操作是除以128，输出维度与输入相同。

conv0层的输入是mulscalar层的输出，进行卷积，参数维度和输出维度参见表1。

bn0层的输入是conv0层的输出，进行批量归一化(Batch Norm)，具体而言，输入信号，参数，辅助参数，其中分别表示moving\_mean和moving\_var。输出信号**y**和输入维度相同，且对于**y**的第*c*个通道，*c*=1,…,*C*有：



其中。

prelu0层的输入为bn0层的输出，进行Leaky ReLU的操作，具体而言，输入信号，参数，输出信号**y**和输入维度相同，且对于**y**的第*c*个通道里的每一个元素，*c*=1,…,*C*有：



* 1. Stage1

Stage1一共有3个残差结构的unit，如图2所示，为了使相加的维度一致，第一个unit要多接一个conv层和bn层。

bn0

stage1\_unit1\_sc

stage1\_unit2

stage1\_unit3

图2：Stage1的网络结构

每个unit的结构如图3所示。其中stage*i*\_unit*j*表示第i个stage第j个unit。右边虚线部分的conv1sc和bn1sc是只在每个stage的unit1才有，其他unit则是直接连接到plus层。各个conv层的参数维度见表1。

Input

stage*i*\_unit*j*\_bn1

stage*i*\_unit*j*\_conv1

stage*i*\_unit*j*\_prelu1

stage*i*\_unit*j*\_bn2

stage*i*\_unit*j*\_bn3

stage*i*\_unit*j*\_conv2

plus

stage*i*\_unit1\_conv1sc

stage*i*\_unit1\_bn1sc

图3：unit的网络结构

* 1. Stage2

Stage2的网络结构与Stage1类似，一共有4个unit。即stage2\_unit1\_sc, stage2\_unit2, stage2\_unit3, stage2\_unit4。如图4所示。

stage1\_output

stage2\_unit1\_sc

stage2\_unit2

stage2\_unit3

stage2\_unit4

图4：Stage2的网络结构

* 1. Stage3

Stage3的网络结构与Stage1类似，一共有6个unit。即stage3\_unit1\_sc, stage3\_unit2, stage3\_unit3, stage3\_unit4, stage3\_unit5, stage3\_unit6。如图5所示。

stage2\_output

stage3\_unit1\_sc

stage3\_unit2

stage3\_unit3

stage3\_unit4

stage3\_unit5

stage3\_unit6

图6：Stage3的网络结构

* 1. Stage4

Stage4的网络结构与Stage1类似，一共有3个unit。即stage4\_unit1\_sc, stage4\_unit2, stage4\_unit3。如图7所示。

stage3\_output

stage4\_unit1\_sc

stage4\_unit2

stage4\_unit3

图7：Stage4的网络结构

* 1. fc1

最后，从Stage4输出的数据经过bn1，fc1和bn\_fc1得到最后的输出，如图8所示。

其中fc1为全连接层，需要将输入拉成一个矢量，并与fc1的参数做矩阵乘法得到输出。需要注意的是，输入信号原本维度为(N,C,H,W)，拉成矢量前需要将维度变成(W,H,C,N)。

stage4\_output

bn1

fc1

bn\_fc1

output

图8：全连接层输出

表1：ResNet34中卷积层的维度

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Kernel | Padding | Stride | NumFilter | Output |
| Data | - | - | - | - | (1,3,112,112) |
| conv0 | 3 | 1 | 1 | 64 | (1,64,112,112) |
| st1\_u1\_conv1 | 3 | 1 | 1 | 64 | (1,64,112,112) |
| st1\_u1\_conv2 | 3 | 1 | 2 | 64 | (1,64,56,56) |
| st1\_u1\_conv1sc | 1 | 0 | 2 | 64 | (1,64,56,56) |
| st1\_u2\_conv1 | 3 | 1 | 1 | 64 | (1,64,56,56) |
| st1\_u2\_conv2 | 3 | 1 | 1 | 64 | (1,64,56,56) |
| st1\_u3\_conv1 | 3 | 1 | 1 | 64 | (1,64,56,56) |
| st1\_u3\_conv2 | 3 | 1 | 1 | 64 | (1,64,56,56) |
| st2\_u1\_conv1 | 3 | 1 | 1 | 128 | (1,128,56,56) |
| st2\_u1\_conv2 | 3 | 1 | 2 | 128 | (1,128,28,28) |
| st2\_u1\_conv1sc | 1 | 0 | 2 | 128 | (1,128,28,28) |
| st2\_u2\_conv1 | 3 | 1 | 1 | 128 | (1,128,28,28) |
| st2\_u2\_conv2 | 3 | 1 | 1 | 128 | (1,128,28,28) |
| st2\_u3\_conv1 | 3 | 1 | 1 | 128 | (1,128,28,28) |
| st2\_u3\_conv2 | 3 | 1 | 1 | 128 | (1,128,28,28) |
| st2\_u4\_conv1 | 3 | 1 | 1 | 128 | (1,128,28,28) |
| st2\_u4\_conv2 | 3 | 1 | 1 | 128 | (1,128,28,28) |
| st3\_u1\_conv1 | 3 | 1 | 1 | 256 | (1,256,28,28) |
| st3\_u1\_conv2 | 3 | 1 | 2 | 256 | (1,256,14,14) |
| st3\_u1\_conv1sc | 1 | 0 | 2 | 256 | (1,256,14,14) |
| st3\_u2\_conv1 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u2\_conv2 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u3\_conv1 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u3\_conv2 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u4\_conv1 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u4\_conv2 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u5\_conv1 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u5\_conv2 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u6\_conv1 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st3\_u6\_conv2 | 3 | 1 | 1 | 256 | (1,256,14,14) |
| st4\_u1\_conv1 | 3 | 1 | 1 | 512 | (1,512,14,14) |
| st4\_u1\_conv2 | 3 | 1 | 2 | 512 | (1,512,7,7) |
| st4\_u1\_conv1sc | 1 | 0 | 2 | 512 | (1,512,7,7) |
| st4\_u2\_conv1 | 3 | 1 | 1 | 512 | (1,512,7,7) |
| st4\_u2\_conv2 | 3 | 1 | 1 | 512 | (1,512,7,7) |
| st4\_u3\_conv1 | 3 | 1 | 1 | 512 | (1,512,7,7) |
| st4\_u3\_conv2 | 3 | 1 | 1 | 512 | (1,512,7,7) |
| fc1 | 25088 | - | - | 512 | (1,512) |

参考文献