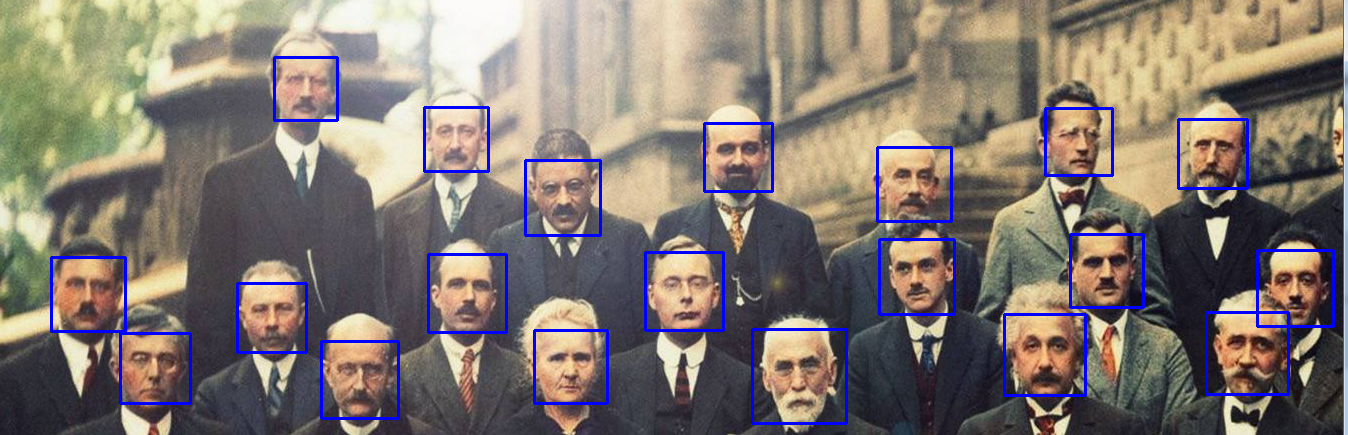
# 来源：[第三十七节、人脸检测MTCNN和人脸识别Facenet(附源码)](https://www.cnblogs.com/zyly/p/9703614.html)

**目录**

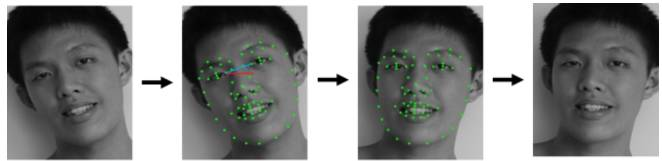
* [一 MTCNN算法结构](https://www.cnblogs.com/zyly/p/9703614.html#_label0)
  + [1、P-Net网络](https://www.cnblogs.com/zyly/p/9703614.html#_label0_0)
  + [2、R-Net](https://www.cnblogs.com/zyly/p/9703614.html#_label0_1)
  + [3、O-Net](https://www.cnblogs.com/zyly/p/9703614.html#_label0_2)
* [二 MTCNN损失函数](https://www.cnblogs.com/zyly/p/9703614.html#_label1)
  + [1、人脸识别损失函数](https://www.cnblogs.com/zyly/p/9703614.html#_label1_0)
  + [2、框回归](https://www.cnblogs.com/zyly/p/9703614.html#_label1_1)
  + [3、关键点损失函数](https://www.cnblogs.com/zyly/p/9703614.html#_label1_2)
  + [4、总损失](https://www.cnblogs.com/zyly/p/9703614.html#_label1_3)
  + [5、Online Hard sample mining](https://www.cnblogs.com/zyly/p/9703614.html#_label1_4)
  + [6、训练数据](https://www.cnblogs.com/zyly/p/9703614.html#_label1_5)
* [三 人脸识别](https://www.cnblogs.com/zyly/p/9703614.html#_label2)
  + [1、三元组损失](https://www.cnblogs.com/zyly/p/9703614.html#_label2_0)
  + [2、中心损失](https://www.cnblogs.com/zyly/p/9703614.html#_label2_1)
* [四 人脸识别的实现](https://www.cnblogs.com/zyly/p/9703614.html#_label3)
  + [1、配置Facenet环境](https://www.cnblogs.com/zyly/p/9703614.html#_label3_0)
  + [2、下载LFW数据集](https://www.cnblogs.com/zyly/p/9703614.html#_label3_1)
  + [3、LFW数据集预处理(LFW数据库上的人脸检测和对齐)](https://www.cnblogs.com/zyly/p/9703614.html#_label3_2)
  + [4、使用已有模型验证LFW数据集准确率](https://www.cnblogs.com/zyly/p/9703614.html#_label3_3)
  + [5、在LFW数据集上使用已有模型](https://www.cnblogs.com/zyly/p/9703614.html#_label3_4)
  + [6、重新训练新模型](https://www.cnblogs.com/zyly/p/9703614.html#_label3_5)
  + [7、三元组损失和中心损失的定义](https://www.cnblogs.com/zyly/p/9703614.html#_label3_6)

在说到人脸检测我们首先会想到利用Harr特征提取和Adaboost分类器进行人脸检测(有兴趣的可以去一看这篇博客[第九节、人脸检测之Haar分类器](https://www.cnblogs.com/zyly/p/9410563.html))，其检测效果也是不错的，但是目前人脸检测的应用场景逐渐从室内演变到室外，从单一限定场景发展到广场、车站、地铁口等场景，人脸检测面临的要求越来越高，比如：人脸尺度多变、数量冗大、姿势多样包括俯拍人脸、戴帽子口罩等的遮挡、表情夸张、化妆伪装、光照条件恶劣、分辨率低甚至连肉眼都较难区分等。在这样复杂的环境下基于Haar特征的人脸检测表现的不尽人意。随着深度学习的发展，基于深度学习的人脸检测技术取得了巨大的成功，在这一节我们将会介绍MTCNN算法，它是基于卷积神经网络的一种高精度的实时人脸检测和对齐技术。

搭建人脸识别系统的第一步就是人脸检测，也就是在图片中找到人脸的位置。在这个过程中输入的是一张含有人脸的图像，输出的是所有人脸的矩形框。一般来说，人脸检测应该能够检测出图像中的所有人脸，不能有漏检，更不能有错检。



获得人脸之后，第二步我们要做的工作就是人脸对齐，由于原始图像中的人脸可能存在姿态、位置上的差异，为了之后的统一处理，我们要把人脸“摆正”。为此，需要检测人脸中的关键点，比如眼睛的位置、鼻子的位置、嘴巴的位置、脸的轮廓点等。根据这些关键点可以使用仿射变换将人脸统一校准，以消除姿势不同带来的误差。



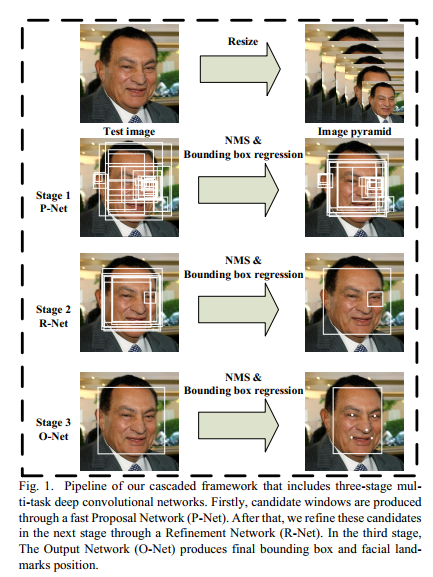
[回到顶部](https://www.cnblogs.com/zyly/p/9703614.html#_labelTop)

**一 MTCNN算法结构**

MTCNN算法是一种基于深度学习的人脸检测和人脸对齐方法，它可以同时完成人脸检测和人脸对齐的任务，相比于传统的算法，它的性能更好，检测速度更快。

MTCNN算法包含三个子网络：Proposal Network(P-Net)、Refine Network(R-Net)、Output Network(O-Net)，这三个网络对人脸的处理依次从粗到细。

在使用这三个子网络之前，需要使用图像金字塔将原始图像缩放到不同的尺度，然后将不同尺度的图像送入这三个子网络中进行训练，目的是为了可以检测到不同大小的人脸，从而实现多尺度目标检测。

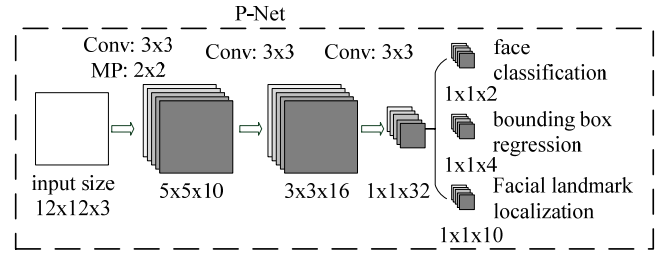


**1、P-Net网络**

P-Net的主要目的是为了生成一些候选框，我们通过使用P-Net网络，对图像金字塔图像上不同尺度下的图像的每一个12×1212×12区域都做一个人脸检测(实际上在使用卷积网络实现时，一般会把一张h×wh×w的图像送入P-Net中，最终得到的特征图每一点都对应着一个大小为12×1212×12的感受野，但是并没有遍历全一张图像每一个12×1212×12的图像)。

P-Net的输入是一个12×12×312×12×3的RGB图像，在训练的时候，该网络要判断这个12×1212×12的图像中是否存在人脸，并且给出人脸框的回归和人脸关键点定位；

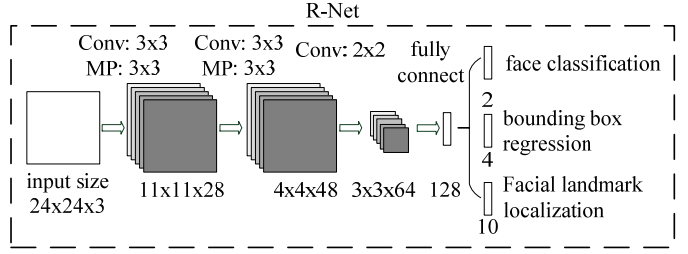
在测试的时候输出只有NN个边界框的4个坐标信息和score，当然这4个坐标信息已经使用网络的人脸框回归进行校正过了，score可以看做是分类的输出(即人脸的概率)：



* 网络的第一部分输出是用来判断该图像是否包含人脸，输出向量大小为1×1×21×1×2，也就是两个值，即图像是人脸的概率和图像不是人脸的概率。这两个值加起来严格等于1，之所以使用两个值来表示，是为了方便定义交叉熵损失函数。
* 网络的第二部分给出框的精确位置，一般称为框回归。P-Net输入的12×1212×12的图像块可能并不是完美的人脸框的位置，如有的时候人脸并不正好为方形，有可能12×1212×12的图像偏左或偏右，因此需要输出当前框位置相对完美的人脸框位置的偏移。这个偏移大小为1×1×41×1×4，即表示框左上角的横坐标的相对偏移，框左上角的纵坐标的相对偏移、框的宽度的误差、框的高度的误差。
* 网络的第三部分给出人脸的5个关键点的位置。5个关键点分别对应着左眼的位置、右眼的位置、鼻子的位置、左嘴巴的位置、右嘴巴的位置。每个关键点需要两维来表示，因此输出是向量大小为1×1×101×1×10。

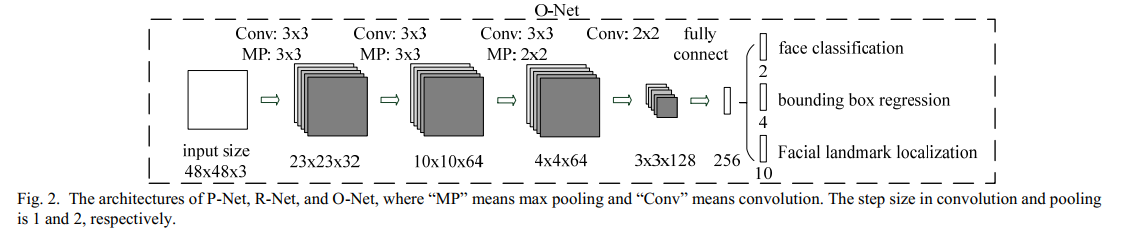
**2、R-Net**

由于P-Net的检测时比较粗略的，所以接下来使用R-Net进一步优化。R-Net和P-Net类似，不过这一步的输入是前面P-Net生成的边界框，不管实际边界框的大小，在输入R-Net之前，都需要缩放到24×24×324×24×3。网络的输出和P-Net是一样的。这一步的目的主要是为了去除大量的非人脸框。



**3、O-Net**

进一步将R-Net的所得到的区域缩放到48×48×348×48×3，输入到最后的O-Net，O-Net的结构与P-Net类似，只不过在测试输出的时候多了关键点位置的输出。输入大小为48×48×348×48×3的图像，输出包含PP个边界框的坐标信息，score以及关键点位置。



 从P-Net到R-Net，再到最后的O-Net，网络输入的图像越来越大，卷积层的通道数越来越多，网络的深度也越来越深，因此识别人脸的准确率应该也是越来越高的。同时P-Net网络的运行速度越快，R-Net次之、O-Net运行速度最慢。之所以使用三个网络，是因为一开始如果直接对图像使用O-Net网络，速度会非常慢。实际上P-Net先做了一层过滤，将过滤后的结果再交给R-Net进行过滤，最后将过滤后的结果交给效果最好但是速度最慢的O-Net进行识别。这样在每一步都提前减少了需要判别的数量，有效地降低了计算的时间。

[回到顶部](https://www.cnblogs.com/zyly/p/9703614.html#_labelTop)

**二 MTCNN损失函数**

 由于MTCNN包含三个子网络，因此其损失函数也由三部分组成。针对人脸识别问题，直接使用交叉熵代价函数，对于框回归和关键点定位，使用L2L2损失。最后把这三部分的损失各自乘以自身的权重累加起来，形成最后的总损失。在训练P-Net和R-Net的时候，我们主要关注目标框的准确度，而较少关注关键点判定的损失，因此关键点损失所占的权重较小。对于O-Net，比较关注的是关键点的位置，因此关键点损失所占的权重就会比较大。

**1、人脸识别损失函数**

在针对人脸识别的问题，对于输入样本xixi,我们使用交叉熵代价函数：

Ldeti=−(ydetilog(pi)+(1−ydeti)(1−log(pi)))Lidet=−(yidetlog(pi)+(1−yidet)(1−log(pi)))

其中ydetiyidet表示样本的真实标签，pipi表示网络输出为人脸的概率。

**2、框回归**

对于目标框的回归，我们采用的是欧氏距离：

Lboxi=∥y^boxi−yboxi∥Libox=‖y^ibox−yibox‖

其中y^boxiy^ibox表示网络输出之后校正得到的边界框的坐标，yboxiyibox是目标的真实边界框。

**3、关键点损失函数**

对于关键点，我们也采用的是欧氏距离：

Llandmarki=∥y^landmarki−ylandmarki∥Lilandmark=‖y^ilandmark−yilandmark‖

其中y^landmarkiy^ilandmark表示网络输出之后得到的关键点的坐标，ylandmarkiyilandmark是关键点的真实坐标。

**4、总损失**

 把上面三个损失函数按照不同的权重联合起来：

min∑i=1N∑j∈{det,box,landmark}αjβjiLjimin∑i=1N∑j∈{det,box,landmark}αjβijLij

其中NN是训练样本的总数，αjαj表示各个损失所占的权重，在P-Net和R-net中，设置αdet=1,αbox=0.5,αlandmark=0.5αdet=1,αbox=0.5,αlandmark=0.5，在O-Net中，设置αdet=1,αbox=0.5,αlandmark=1αdet=1,αbox=0.5,αlandmark=1,βji∈{0,1}βij∈{0,1}表示样本类型指示符。

**5、Online Hard sample mining**

**In particular, in each mini-batch, we sort the losses computed in the forward propagation from all samples and select the top 70% of them as hard samples. Then we only compute the gradients from these hard samples in the backward propagation.That means we ignore the easy samples that are less helpful to strengthen the detector during training. Experiments show that this strategy yields better performance without manual sampleselection.**

这段话也就是说，我们在训练的时候取前向传播损失值(从大到小)前70%的样本，来进行反向传播更新参数。

**6、训练数据**

该算法训练数据来源于wider和celeba两个公开的数据库，wider提供人脸检测数据，在大图上标注了人脸框groundtruth的坐标信息，celeba提供了5个landmark点的数据。根据参与任务的不同，将训练数据分为四类:

* 负样本：滑动窗口和Ground True的IOU小于0.3；
* 正样本：滑动窗口和Ground True的IOU大于0.65；
* 中间样本：滑动窗口和Ground True的IOU大于0.4小于0.65；
* 关键点：包含5个关键点做标的；

上面滑动窗口指的是：通过滑动窗口或者随机采样的方法获取尺寸为12×1212×12的框：

wider数据集，数据可以从<http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/>地址下载。该数据集有32,203张图片，共有93,703张脸被标记，如下图所示：



celeba人脸关键点检测的训练数据，数据可从<http://mmlab.ie.cuhk.edu.hk/archive/CNN_FacePoint.htm>地址下载。该数据集包含5,590张 LFW数据集的图片和7,876张从网站下载的图片。



[回到顶部](https://www.cnblogs.com/zyly/p/9703614.html#_labelTop)

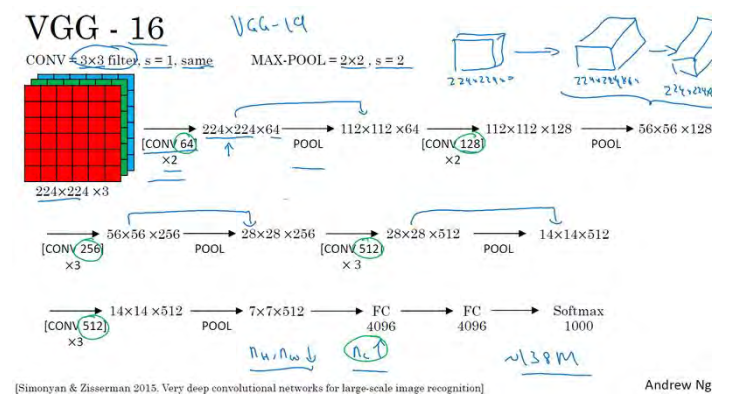
**三 人脸识别**

在上面我们已经介绍了人脸检测，人脸检测是人脸相关任务的前提，人脸相关的任务主要有以下几种：

* 人脸跟踪(视频中跟踪人脸位置变化)；
* 人脸验证(输入两张人脸，判断是否属于同一人)；
* 人脸识别(输入一张人脸，判断其属于人脸数据库记录中哪一个人)；
* 人脸聚类(输入一批人脸，将属于同一人的自动归为一类)；

下面我们来详细介绍人脸识别技术：当我们通过MTCNN网络检测到人脸区域图像时，我们使用深度卷积网络，将输入的人脸图像转换为一个向量的表示，也就是所谓的特征。

那我们如何对人脸提取特征？我们先来回忆一下VGG16网络，输入的是图像，经过一系列卷积计算、全连接网络之后，输出的是类别概率。



在通常的图像应用中，可以去掉全连接层，使用卷积层的最后一层当做图像的“特征”，如上图中的conv5\_3。但如果对人脸识别问题同样采用这样的方法，即，使用卷积层最后一层做为人脸的“向量表示”，效果其实是不好的。如何改进？我们之后再谈，这里先谈谈我们希望这种人脸的“向量表示”应该具有哪些性质。

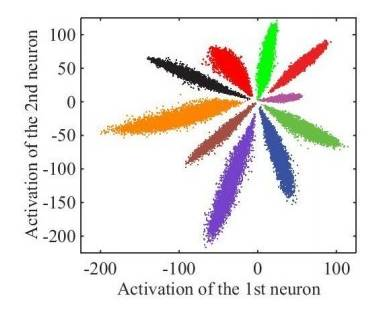
在理想的状况下，我们希望“向量表示”之间的距离就可以直接反映人脸的相似度：

* 对于同一个人的人脸图像，对应的向量的欧几里得距离应该比较小；
* 对于不同人的人脸图像，对应的向量之间的欧几里得距离应该比较大；

例如：设人脸图像为x1,x2x1,x2，对应的特征为f(x1),f(x2)f(x1),f(x2)，当x1,x2x1,x2对应是同一个人的人脸时，f(x1),f(x2)f(x1),f(x2)的距离∥f(x1)−f(x2)∥2‖f(x1)−f(x2)‖2应该很小，而当x1,x2x1,x2对应的不是同一个人的人脸时，f(x1),f(x2)f(x1),f(x2)的距离∥f(x1)−f(x2)∥2‖f(x1)−f(x2)‖2应该很大。

在原始的VGG16模型中，我们使用的是softmax损失，softmax是类别间的损失，对于人脸来说，每一类就是一个人。尽管使用softmax损失可以区别每个人，但其本质上没有对每一类的向量表示之间的距离做出要求。

举个例子，使用CNN对MNIST进行分类，我们设计一个特殊的卷积网络，让最后一层的向量变为2维，此时可以画出每一类对应的2维向量表示的图（图中一种颜色对应一种类别）：



上图是我们直接使用softmax训练得到的结果，它就不符合我们希望特征具有的特点：

* 我们希望同一类对应的向量表示尽可能接近。但这里同一类（如紫色），可能具有很大的类间距离；
* 我们希望不同类对应的向量应该尽可能远。但在图中靠中心的位置，各个类别的距离都很近；

对于人脸图像同样会出现类似的情况，对此，有很改进方法。这里介绍其中两种：三元组损失函数，中心损失函数。

**1、三元组损失**

三元组损失函数的原理：既然目标是特征之间的距离应该具备某些性质，那么我们就围绕这个距离来设计损失。具体的，我们每次都在训练数据中抽出三张人脸图像，第一张图像标记为xaixia，第二章图像标记为xpixip，第三张图像标记为xnixin。在这样一个"三元组"中，xaixia和xpixip对应的是同一个人的图像，而xnixin是另外一个人的人脸图像。因此距离∥f(xai)−f(xpi)∥2‖f(xia)−f(xip)‖2应该很小，而距离∥f(xai)−f(xni)∥2‖f(xia)−f(xin)‖2应该很大。严格来说，三元组损失要求满足以下不等式：

∥f(xai)−f(xpi)∥22+α<∥f(xai)−f(xni)∥22‖f(xia)−f(xip)‖22+α<‖f(xia)−f(xin)‖22

即相同人脸间的距离平方至少要比不同人脸间的距离平方小αα(取平方主要是为了方便求导)，据此，设计损失函数为：

Li=[∥f(xai)−f(xpi)∥22+α−∥f(xai)−f(xni)∥22]+Li=[‖f(xia)−f(xip)‖22+α−‖f(xia)−f(xin)‖22]+

这样的话，当三元组的距离满足∥f(xai)−f(xpi)∥22+α<∥f(xai)−f(xni)∥22‖f(xia)−f(xip)‖22+α<‖f(xia)−f(xin)‖22时，损失Li=0Li=0。当距离不满足上述不等式时，就会有值为∥f(xai)−f(xpi)∥22+α−∥f(xai)−f(xni)∥22‖f(xia)−f(xip)‖22+α−‖f(xia)−f(xin)‖22的损失，此外，在训练时会固定∥f(x)∥2=1‖f(x)‖2=1，以确保特征不会无限的"远离"。

三元组损失直接对距离进行优化，因此可以解决人脸的特征表示问题。但是在训练过程中，三元组的选择非常地有技巧性。如果每次都是随机选择三元组，虽然模型可以正确的收敛，但是并不能达到最好的性能。如果加入"难例挖掘"，即每次都选择最难分辨率的三元组进行训练，模型又往往不能正确的收敛。对此，又提出每次都选择那些"半难"的数据进行训练，让模型在可以收敛的同时也保持良好的性能。此外，使用三元组损失训练人脸模型通常还需要非常大的人脸数据集，才能取得较好的效果。

**2、中心损失**

与三元组损失不同，中心损失不直接对距离进行优化，它保留了原有的分类模型，但又为每个类(在人脸模型中，一个类就对应一个人)指定了一个类别中心。同一类的图像对应的特征都应该尽量靠近自己的类别中心，不同类的类别中心尽量远离。与三元组损失函数，使用中心损失训练人脸模型不需要使用特别的采样方法，而且利用较少的图像就可以达到与单元组损失相似的效果。下面我们一起来学习中心损失的定义：

设输入的人脸图像为xixi，该人脸对应的类别是yiyi，对每个类别都规定一个类别中心，记作cyicyi。希望每个人脸图像对应的特征f(xi)f(xi)都尽可能接近中心cyicyi。因此定义损失函数为：

Li=12∥f(xi)−cyi∥22Li=12‖f(xi)−cyi‖22

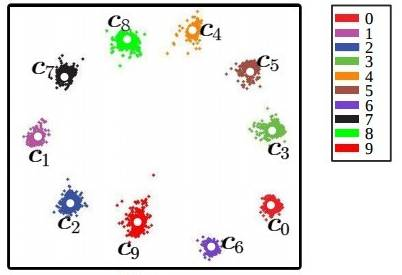
多张图像的中心损失就是将它们的值累加：

Lcenter=∑iLiLcenter=∑iLi

这是一个非常简单的定义。不过还有一个问题没有解决，那就是如何确定每个类别的中心cyicyi呢？从理论上来说，类别yiyi的最佳中心应该是它对应所有图片的特征的平均值。但如果采用这样的定义，那么在每一次梯度下降时，都要对所有图片计算一次cyicyi，计算复杂度太高了。针对这种情况，不妨近似处理下，在初始阶段，先随机确定cyicyi，接着在每个batch内，使用Li=∥f(xi)−cyi∥22Li=‖f(xi)−cyi‖22对当前batch内的cyicyi也计算梯度，并使得该梯度更新cyicyi，此外，不能只使用中心损失来训练分类模型，还需要加入softmax损失，也就是说，损失最后由两部分组成，即L=Lsoftmax+λLcenterL=Lsoftmax+λLcenter,其中λλ是一个超参数。

最后来总结使用中心损失来训练人脸模型的过程。首先随机初始化各个中心cyicyi，接着不断地取出batch进行训练，在每个batch中，使用总的损失LL，除了使用神经网络模型的参数对模型进行更新外，也对cyicyi进行计算梯度，并更新中心的位置。

中心损失可以让训练处的特征具有"内聚性"。还是以MNIST的例子来说，在未加入中心损失时，训练的结果不具有内聚性。在加入中心损失后，得到的特征如下：



当中心损失的权重λλ越大时，生成的特征就会具有越明显的"内聚性"。