江 西 理 工 大 学

本 科 毕 业 设 计（论文）

题　　目：面向大规模人脸识别模型的深度学习方法研究

专题题目：

学　　院(学部)：信息工程学院

专　　业：人工智能专业

班　　级：21 人智 2 班

学　　号：1520203586

学　　生：朱金康

指导教师：梁苗苗　　职称：副教授

时间：

摘　要

随着社会对安全问题的关注增加，迫切需要高效准确的人脸识别技术，因为对于精度要求的增大，现在越来越多规模越来越大的人脸识别数据集出现了。于是现存的方法在大规模人脸识别方向的应用变为了焦点，现存能提升精度的主要方法包含增加训练网络隐藏层的层数、增加训练用的数据集的大小和使用更加有效的损失函数等，前两种方法的局限性在于，因为模型和全连接层的参数需要存在显存里面，导致训练的过程对硬件性能的要求非常的高。

特别是全连接层，因为模型占用的显存的大小是固定的，而全连接层的大小会随着数据集内的人脸身份的增加而增加，所以对于较大的数据集，这点是不可接受的显存占用。本文采用动态的队列池来存储人脸特征向量，可以自行的选择动态队列池的大小，采用多种加载数据的方式来进行数据集的 Loader 构建，并且结合了现存的高效的角边距损失函数来确保精确度。

关键词：大规模数据集；动态队列池；角边距损失函数；节省硬件资源

Abstract

With the increasing concern of society about security issues, there is an urgent need for efficient and accurate face recognition technology, because of the increase in accuracy requirements, more and more large-scale face recognition datasets are now emerging. The main methods that can improve the accuracy include increasing the number of layers of the hidden layer of the training network, increasing the size of the dataset used for training, and using a more effective loss function.

This is especially true for the fully connected layer, which is not an acceptable memory footprint for larger datasets, because the size of the memory occupied by the model is fixed, and the size of the fully connected layer increases with the number of IDs in the dataset. In this paper, a dynamic queue pool is used to store face feature vectors, the size of the dynamic queue pool can be selected by yourself, a variety of ways to load data are used to construct the loader of the dataset, and the existing efficient corner margin loss function is combined to ensure accuracy.

Keywords: large-scale datasets; dynamic queue pools; corner margin loss functions; saving hardware resources

目 录

第一章 绪论

1.1 研究背景及其意义

智能手机和平板电脑等移动设备越来越依赖于生物特征识别技术（主要是人脸识别）作为便捷且安全的解锁方式，从而需要更高的精度来保证安全。社交媒体平台和在线服务也利用人脸识别技术为用户提供照片标记、好友推荐等功能，增强用户体验，为了节省资源的情况下也能得到较好的精度，本文的方法结合了角边距损失函数。

这段时间以来，深度卷积神经网络成为了研究人脸识别问题的一项被广泛研究的技术，因为其在人脸识别领域有较好的效果。人脸识别的通过大量的标注数据来进行训练的，训练过程中使用的网络能自动提取高维的人脸特征转化为概率，经过很长时间的发展，人脸识别已经达到了较高的精度。为了达到更高的精度，前人设计了很多专用于人脸识别的损失函数，如中心损失（Center Loss）[1]、三元组损失（Triplet Loss）[2]等，这些损失函数有助于模型学习到更加分离和紧凑的特征空间。除此之外，大规模人脸数据库（如 LFW、CASIA-WebFace、MS-Celeb-1M 等）的出现为人脸识别提供了丰富的训练资源，这也让人脸识别的精度进一步提升。

深度神经网络（DNN）可以对已标记的数据集进行训练，采用梯度下降的方法，让图片经过网络训练后得到的特征向量更加倾向于标签. 人脸识别可以看作是计算机视觉中最受欢迎的研究主题之一。人脸识别旨在学习与身份相关的嵌入空间，其中主要的原则就是减小类别距离，同时，增加类间的距离。以前的工作 ArcFace[3]、AMSoftmax[4]、CosFace[5]、L-Softamx[6]、SphereFace[7]已经证明，在大型数据集上进行训练可以获得比小型数据集更好的结果。为此，学术界和工业界收集了超大型扩展数据集。

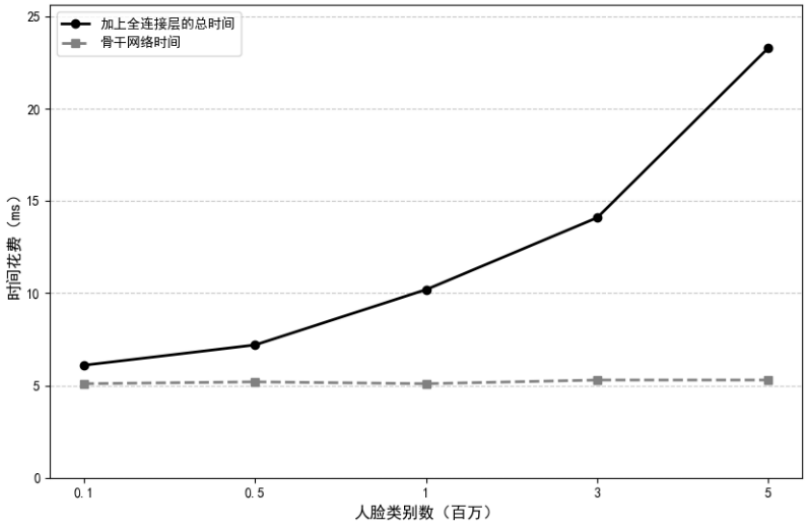

图 1.1 时间消耗对比图

一般来说随着数据集的增大，人脸识别的准确度也会随之增加，但是可以预见的是随着数据集的增大，识别过程对于硬件的消耗、显存的消耗、时间的占用是不可接受的。如图 1.1、图 1.2 所示，显示了人脸类别数的增长与时间消耗和显存占用的关系，可以发现即使是当前的主流顶级显卡面对超大类型数据集也没有优势。

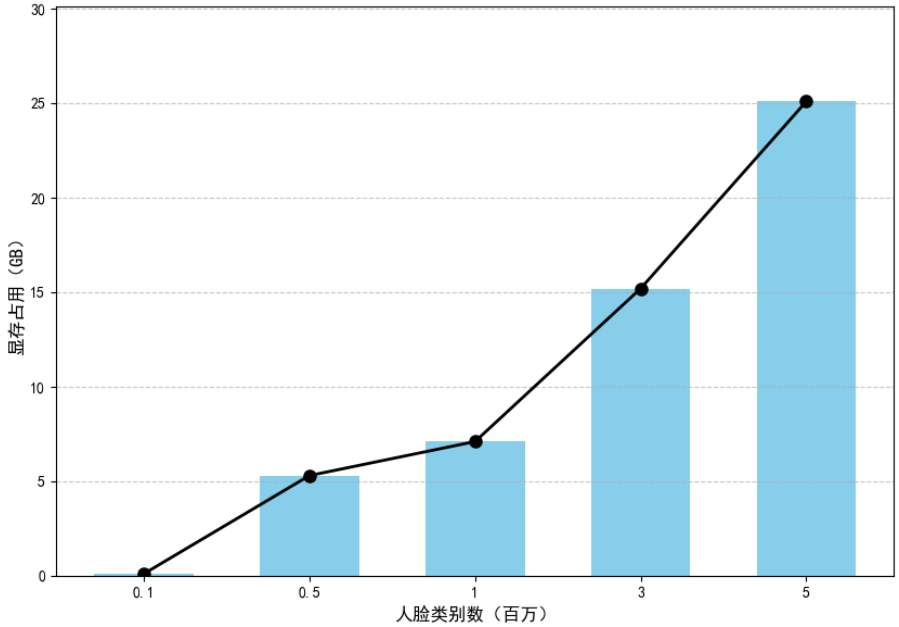

图 1.2 显存占用对比图

为了解决这些问题，之前的研究主要的思路是尝试将整个全连接层分配到不同的 GPU 上面，但是这样会带来沉重的通信成本。如何有效降低全连接层带来的计算和内存成本是本文主要的研究方向，直观的想法就是减少全连接层的大小，来解决降低训练时间和训练成本的问题。

1.2 研究现状

面部识别问题天然是一个分类问题，因此使用 Softmax 损失来训练是很正常。因为 Softmax 本身就是一个多分类器，但是原本的 Softmax 在分类方面存在诸多的问题，导致即使 Softmax 可以用于分类，但是分类的效果不佳，特别是对于人脸识别之中对于分类要求精度较高的分类任务，Softmax 作为分类器就有些不太合适了，因为 Softmax 只进行了像同类尽量在一起的任务，但是没有实现不同类之间尽量分开的任务，所以近年来的很多论文都在研究如何调整 Softmax 能使得 Softmax 在人脸分类领域有较好的结果。

人脸识别领域近年来取得了显著进展，其中基于角度间隔（angular margin）的损失函数设计成为提升特征判别性的核心方法，即改进版本的 Softmax。首先就是几何解释特别优秀并且分类效果也很好的 ArcFace 其核心思想在角度空间

引入加性间隔，实现更几何直观的优化。直接在角度上添加间隔，直接作用于角度空间，与测地距离（geodesic distance）关联更紧密。在统一超球面上实现类内聚合和类间分离，这点和人类的直观感受相同，具有良好的几何解释。另外分类优秀的就是 CosFace 其核心思想，提出加性余弦间隔（additive cosine margin），简化优化难度。直接在余弦空间中添加固定间隔，并且归一化了权重向量，消除了模长的影响，专注角度的优化，间隔 m 是在余弦空间里面添加的，线性可分性更强，训练更加的稳定，这两个是人脸分类任务里面最常见的任务。L-Softmax 首次在 Softmax 损失中引入角度间隔，通过调节决策边界附近的几何间隔增强类内紧致性。证明了角度间隔比欧氏间隔更适用于人脸特征学习。但是他的优化难度太大，，相较于前两种方法，L-Softmax 及其难以实现，甚至需要等效。以 L-Softmax 为基础，原作者提出了 SphereFace 将 L-Softmax 的间隔思想推向极致，提出乘性角度间隔（multiplicative angular margin），但是即使这样他的几何解释也还是没有上面的 ArcFace 和 CosFace 更加明显，优化难度也更大，所以本文的训练过程选用的度量函数就是 CosFace 与 ArcFace。Sphereface 提出了 A-Softmax，在计算损失前将每个类别的权重向量正则化。训练中实际上 Sphereface 是由 A-softmax 和 Softmax 组成的，并且占比是 A-Softmax 由小逐渐变大。这是因为直接使用 A-Softmax 会造成收敛困难。

除了上面介绍的人脸识别方向的进步之外，早期大规模人脸的训练方面，依赖参数服务器架构，使用多 GPU 通信[8]，通过中央节点同步梯度，但存在单点瓶颈，类似于计算机网络里面的星型结构。后续 NCCL 库成为主流，采用 Ring-ALLReduce[9] 优化了通信的效果，使得梯度同步的开销从 O(N) 变成了 O(1)。DCP（Dynamic Class Pooling）[10]由 F²C（CVPR 2022）提出，通过概率采样减少参与 Softmax 计算的类别数，结合记忆库管理类别中心，显著降低显存占用。DCQ（Dynamic Class Queue）[11]则采用滑动窗口队列动态维护活跃类别，优先计算队列内样本的对比损失，适用于流数据场景。两者的核心创新减少了训练过程的类别总数，显著降低了训练的时间和显存占用，并且可以与之前提到的 GPU 通信相结合，这样搭建的训练网络有较好的效果。

1.3 本章小结

这一个章节综合讲解了人脸识别领域的研究背景、意义以及当前的研究现状。随着智能设备和社交平台对人脸识别技术的广泛应用，这个奇数对效率和精度的要求日益增强。近年来深度卷积神经网络在人脸特征提取方面展现了很高的性能，加上大规模数据集的出现，以及各种效果较好的损失函数的提出，人脸识别的精度进一步的提升，但是人脸识别对硬件的消耗还是太大，对于较大的数据集可以说是不可接受的时间消耗。

常见的效果较好的损失函数主要包括以下几种，他们都是增加了特征的判别性，如 ArcFace、CosFace、SphereFace 等，尤其是 ArcFace 和 CosFace 在角度空间上引入间隔，提升了类内紧致性和类间分离度，取得了优越的性能表现。

随着人脸身份数量的快速的增长，前文已经证实，计算所需要的开销是不可接受的，硬件的限制成为了训练的瓶颈。为了解决这个问题，前人做了很多的尝试，比如多个 GPU 并行训练，优化 GPU 之间的通信有效的减少了训练的时长，或者使用 DCP、DCQ 策略进行类别选择和记忆机制有效的减少了资源的消耗。当前的人脸识别的研究朝着保证精度的前提下进一步提升训练效率和资源利用率的方向发展，为大规模人脸识别系统的构建提供了坚实的理论和技术基础。

第二章 相关工作

2.1 全连接层方法

全连接层（Fully Connected Layer，FC）是神经网络中的一个基本结构，每一个神经元和前一层的所有的神经元相连接，通过权重矩阵实现全局的连接，全连接层的参数量取决于输入和输出的维度，若输入有 n 个神经元，输出有 m 个神经元，则参数量为 nm + m，包含偏置，适合学习输入数据的高层抽象关系。其通过矩阵乘法将前一层的局部或全局特征组合成新的特征表示。

其经常用在分类任务中，最后一层的 FC 通常搭配 Softmax 输出类别的概率，起到了全局信息融和的作用，在人脸识别中，FC 层位于卷积层的后面，将卷积层提取的局部特征整合为全局特征，将空间信息压缩为类别概率。全连接层的优点就是拥有强大的表达能力，适合学习复杂的映射，实现较为简单，易于使用梯度下降进行优化。但是其参数量与数据量成正相关，对于较大的训练集，FC 层难以实现正常的训练过程。当身份类别数较大的时候 FC 层的权重矩阵特别的大，因为 FC 层的输出神经元的个数就是身份类别数。

2.2 相关损失函数

Softmax 中的线性变换矩阵的大小确实会随着身份数的增加而线性增长，这会导致模型的参数量显著增加，尤其是全连接层的参数，训练难度大，部署困难，可拓展性差。还存在的问题就是对闭集分类有效，但对开集问题判别能力不足。在闭集分类问题中，训练集和测试集的类别是相同的，Softmax 损失可以很好地学习到可分离的特征。但在开集人脸识别问题中，测试集可能包含训练集中未出现的身份，Softmax 损失学习到的特征判别能力不足，无法很好地泛化到未知类别。Softmax 损失函数主要关注分类正确性，而没有显式地优化类间间隔，这可能导致类间特征分布不够分散，类内特征分布不够紧凑，从而影响模型的判别能力。想象每个特征的表示为一个特征向量，对于人脸特征的几何解释就是传统的 Softmax 只做到了相同类别的向量之间的距离更近，没有做到不同向量之间的距离更远。

ArcFace：由于嵌入特征分布在超球上的每个特征中心周围，因此可以这样，在两者之间使用了一个加性角裕度惩罚 m，以同时增强类内紧凑性和类间差异。由于所提出的加性角边缘惩罚等于归一化超球中的测地线距离乘法（因为向量进行了归一化，测地线弧长在数值上完全等价于夹角），该方法被称之为 ArcFace，因为其相较于一般的损失函数的设计方法，有较强的集合可解释性质，比如新加的角度 margin，可以很好的解释为类别之间的夹角，因为归一化的缘故，这个夹角又可以解释为两个类中心向量之间的测地线的距离，所以本文选用 ArcFace

作为后接的损失函数来比较 FC 和 FFC 方法的一些评价指标的大小，公式 2.1，ArcFace 损失函数设计。

$$L = -log\frac{e^{scos(\theta_{y_i}+m)}}{e^{scos(\theta_{y_i}+m)} + \sum_{j=1,j\neq y_i}^{N} e^{scos(\theta_j)}} \tag{2.1}$$

L-Softmax：L-Softmax 损失有一个灵活的学习目标，可以通过调整参数 m 调节类间的角度约束，是一个难度可调节的学习任务，随着 m 的增大学习的难度逐渐的增加。它通过定义一个更困难的学习目标，在一定程度上避免了过度拟合。L-Softmax 不仅有利于分类问题，而且还有利于验证问题，在理想的情况下，所学习的特性的类间最小距离应该大于类内最大距离。在这种情况下，学习分离良好的功能可以显着提高性能。L-Softmax 损失不仅继承来自 Softmax 损失的全部优点，而且还学习了不同类别之间角度间隔的特征，该方法存在显著的缺点，首先 L-Softmax 因为提出的时间过早，其没有提出归一化类中心向量和特征向量的观点，这样导致他的几何解释不太简单，还有就是，因为他是乘性的角边距，很难直接应用到生产中，需要进行转化。相比 ArcFace，特就可以通过展开 cos 来寻求一个很简单的表示方法代替其内部增加角边距的形式，公式 2.2，L-Softmax 损失函数设计。

$$L_i = -log(\frac{e^{||W_{y_i}||\,||x_i||\psi(\theta_{y_i})}}{e^{||W_{y_i}||\,||x_i||\psi(\theta_{y_i})} + \sum_{j\neq y_i} e^{||W_j||\,||x_i||cos(\theta_j)}})$$
$$\psi(\theta) = \begin{cases} cos(m\theta), & 0 \leq \theta \leq \frac{\pi}{m} \\ D(\theta), & \frac{\pi}{m} < \theta \leq \pi \end{cases} \tag{2.2}$$

SphereFace：继承了 L-Softmax 的乘性裕度思想，但针对人脸识别任务进行了优化，对权重向量和特征向量进行 L2 归一化（即投影到超球面）。公式与 L-Softmax 几乎相同，实际实现更注重归一化和稳定性，在 L-Softmax 上面做了一定的改进，最大的一点就是前面提到了做了类中心向量和特征向量的归一化，更加适应人脸识别方面，但是相较于其他的方法，他的表示难度和几何可解释性还是太低。

CosFace：在 Softmax 的基础上，对目标类别的 logits 减去一个 margin，从而使模型在训练时需要更大的角度区分度才能正确分类。这样提升了人脸识别任务中模型的判别能力，这个损失函数的设计也能有较好的几何解释，而且，其实现非常的简单，所以经常应用在人脸识别领域，但是相较于 ArcFace 他的几何解释就没有那么的简单了，因为在类中心向量和特征向量余弦相似度之外剪掉一个数值，感觉没有很大的集合意义，但是这个减去一个 margin 的方法能实现

在两个向量之间间接加上 margin，比 L-Softamx 的实现简单很多，不失为一种简单有效的损失函数的设计，公式 2.3，CosFace 损失函数设计。

$$L=\frac{1}{N}\sum_{i}-log(\frac{e^{s(cos(\theta_{y_i},i)-m)}}{e^{s(cos(\theta_{y_i},i)-m)}+\sum_{j\neq y_i}e^{scos(\theta_j,i)}}) \tag{2.3}$$

ArcFace、CosFace 和 SphereFace 都属于基于 margin 的人脸识别损失函数，它们的目标是在嵌入空间中通过添加不同形式的 margin，拉近同类样本之间的距离，同时拉远不同类之间的距离，从而提升模型的判别能力。这些方法都是对传统 Softmax 损失的改进，它们通过在归一化特征与权重的角度或余弦相似度上引入 margin，从而使得模型的决策边界更具区分性。三者的主要区别在于 margin 的施加方式和作用空间。SphereFace（A-Softmax）最早提出角度 margin 的思想，它在角度空间将目标类别的角度乘以一个因子 m，相当于将类内的分布压缩，使得模型需要更小的角度才能正确分类，虽然判别力强，但优化难度较大，L-Softmax 论文中为了使用这个乘性角边距做了较难的等价。CosFace 提出直接在余弦相似度上减去一个 margin，绕过角度空间的复杂优化，在稳定性和训练效率上优于 SphereFace，结构简单，易于实现。ArcFace 则在角度空间中对目标类别加上一个固定的角度 margin，具备更清晰的几何解释，使得类内更紧凑、类间更分散。

2.3 本章小结

这一个章节介绍了相关的工作，主要包含，一般训练过程中的全连接层的作用和放置的位置以及相关的损失函数，FC 层可以理解为用于在训练过程中维护每个类别的类中心特征向量，将全局的特征转换为概率。为了解决大规模人脸识别中全连接层参数冗余与更新效率低下的问题，下文将介绍具体的操作方法实现大规模人脸识别的这个瓶颈。

此外，不管是本章节提到的 FC 还是下文的 FFC，都支持多种边界优化型损失函数（如 ArcFace、CosFace 等）。FC 层经常用在分类任务中，最后一层的 FC 通常搭配 Softmax 输出类别的概率，能起到了全局信息融和的作用，在人脸识别中，FC 层位于卷积层的后面，将卷积层提取的局部特征整合为全局特征，将空间信息压缩为类别概率。这样一来才能方便我们的梯度下降找到最好的参数，实现人脸特征的提取。

在损失函数方面，本文详细分析了多种基于角边距的人脸识别损失函数的原理与优势。传统的 Softmax 函数虽具有良好的分类能力，但其类间区分度不足，尤其在开集识别中表现有限。相比之下，ArcFace、CosFace 与 SphereFace 等方法通过引入 margin 操作显著提升了模型的判别能力。其中，ArcFace 在归一化

特征向量与权重基础上引入加性角度 margin，具有良好的几何解释性与稳定性；CosFace 以减性 margin 直接作用于余弦相似度空间，结构简洁，训练高效；而 SphereFace 则延续 L−Softmax 的乘性 margin 思想，并在归一化操作上作出优化，更加贴合人脸识别任务的需求。通过对这些损失函数的集成与比较，本文验证了 ArcFace 和 CosFace 在本框架下表现较优，成为后续评估中的首选。

总而言之，相关的工作为我们提供了很多的方法和手段实现后文中的 FFC，这里的 FC 层的缺点是后文 FFC 提出的关键，相关的损失函数，也可以很好的加在 FFC 的动态类别池后面，起到和 FC 层加上损失函数相似的效果，本章内容为后续章节的 FFC 层的引入、实验验证和性能评估奠定了坚实的理论基础。

第三章 FFC 方法

3.1 原理介绍

本方法的原理就是先构造一个类中心池，也就是前文提到的动态队列池（DCP Dynamic Class Pool），里面每个位置存储每一类的类中心向量，所以现在问题变成了在尽量不影响正确率的前提下，怎么样能让每个批次训练过程中内部的类别在动态队列池中尽量都存在，这样才能够将训练过程进行下去，实现的方式就是使用一个最近最少使用算法（LRU）来管理我们的动态队列池，该方法的原理图 1.3，当新的类别进来后 该类别放在队列的最前端，同时其他没被使用的类别自然向后移动，当超过队列的最大长度之后，尾部的类别被抛弃，当查询的类别不是新的类别，将该类别提到最前端即可。这样能够将经常出现的常见样本类别尽量维持在我们的队列里面，同时给了不常见类别出现在队列里面的机会。

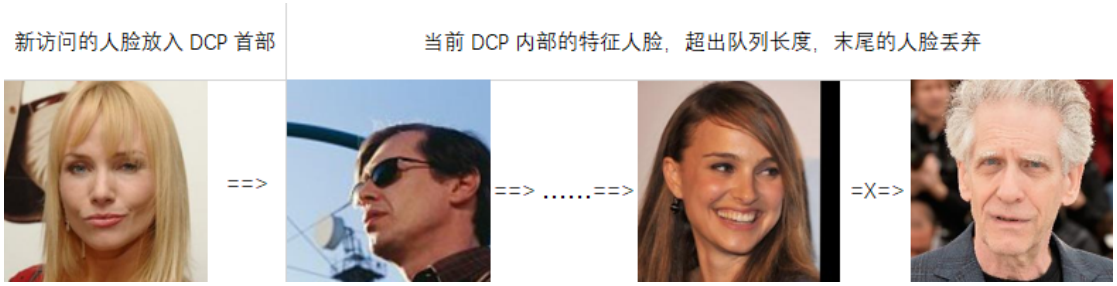

图 3.1 DCP 过程展示

训练过程选用的双向对比学习和动量机制，里面的核心计算损失的模块本文称之为核心模块为 FFC 类（Faster Face Classification），融合了特征更新、动态样本管理与多种边界损失函数，有效增强了模型在大规模人脸识别任务中的判别能力与训练稳定性。

该模块构建了两个特征提取网络，Probe Net（P 网络） 和 Gallery Net（G 网络）。两者结构相同，但 G 网络不参与梯度更新，而是通过动量更新策略进行权重同步。本文的 DCP 维护了一个双通道的人脸特征向量池，用于存储先前样本的特征。动态池以二维张量形式保存特征，使用 LRU（最近最少使用）机制记录身份及其对应的特征索引，通过轮换更新的策略更新当前动态池里面的特征向量，控制双通道更新策略。保留历史信息，又保证了人脸特征向量的多样性。

在每次前向传播中，模型从输入数据中提取特征，并基于标签更新队列。若身份首次出现，则从 LRU 中获取一个空位进行注册；若已存在，则更新对应位置的通道特征。

对于损失计算，损失函数的设计，是该方法的重点，模型支持三种常见的角边距损失函数 AMSoftmax、ArcFace、CosFace，均以余弦相似度为基础对正负样

本进行调节。其中正样本通过标签精确定位对应行的特征，施加 margin 操作，增加角边距；负样本则采用困难负样本挖掘策略，筛选相似度最高的若干个样本参与训练，提高模型鲁棒性。实现类内距离减小和类间距离增加的效果。

模型训练的过程中实现了两个函数，forward_impl 用于标准训练流程，forward_impl_rollback 则实现了回滚机制，在更新特征后可恢复原状态，确保训练过程中关键历史特征不被破坏。这样实现的原因是，每次传进去的两个批次分别作为 P 网络和 G 网络的输入，获得两个损失，增加模型的稳定性，因为需要两次损失的计算，所以第一次损失计算的时候需要回滚。最终的前向传播函数将两个路径的损失加权求和，实现了双向信息对比，提高训练的效果，方法原理图，如图 3.2。

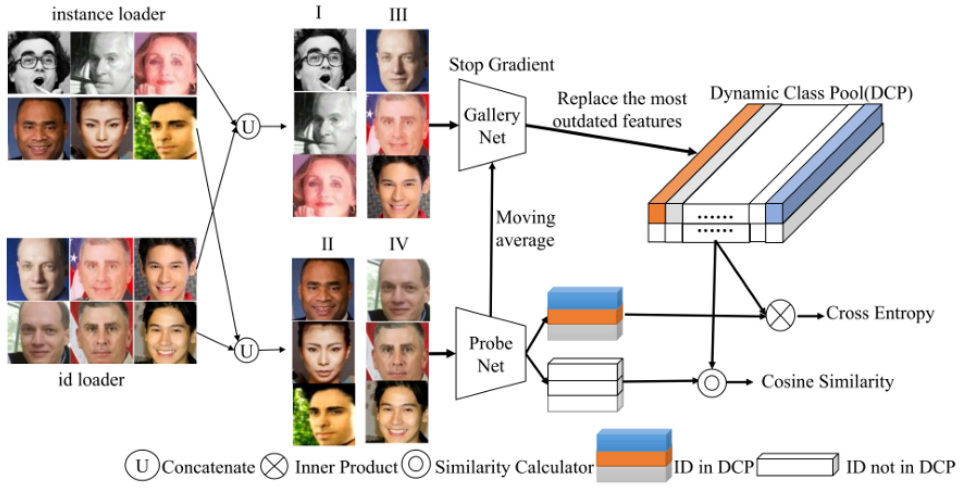

图 3.2 方法原理图

3.2 方法合理性

深入介绍 FFC 之前，下面重新介绍一下与常规 FC 层结合起作，用的损失函数，来引出 FFC 的可行性，经典的 Softmax 考虑如下，公式 3.1 为经典 Softmax 的定义，公式 3.2 为经典 Softmax 的梯度下降的方向，也就是优化方向：

$$\boldsymbol{z}=\begin{bmatrix}z_1\\z_2\\z_3\end{bmatrix}=\begin{bmatrix}w_{11}&w_{21}&\dots&w_{D1}\\w_{12}&w_{22}&\dots&w_{D2}\\w_{13}&w_{23}&\dots&w_{D3}\end{bmatrix}\begin{bmatrix}x_1\\x2\\\dots\\x_D\end{bmatrix}+\begin{bmatrix}b_1\\b_2\\b_3\end{bmatrix}=\boldsymbol{w}^T\boldsymbol{x}+\boldsymbol{b}$$

$$\hat{\boldsymbol{y}}=softmax(\boldsymbol{z})=\begin{bmatrix}softmax(z_1)\\softmax(z_2)\\softmax(z_3)\end{bmatrix}=\begin{bmatrix}e^{z_1}/\sum_{i=1}^{C}e^{z_i}\\e^{z_2}/\sum_{i=1}^{C}e^{z_i}\\e^{z_3}/\sum_{i=1}^{C}e^{z_i}\end{bmatrix}\tag{3.1}$$

$$J(\boldsymbol{w},\boldsymbol{b})=-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}y_j^{(i)}log\hat{y}_j^{(i)}=-\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{y}^{(i)})^T log\hat{\boldsymbol{y}}^{(i)}$$

$$\begin{gathered}\frac{\partial J}{\partial L^{(i)}}=\frac{1}{N}\quad\frac{\partial L}{\partial z_n}=\sum_{i=1}^{C}\frac{\partial L}{\partial \hat{y}_j}\frac{\hat{y}_j}{\partial z_n}=(\hat{y}_n-y_n)\quad\frac{\partial z_n}{\partial w_{nm}}=x_m\\ \frac{\partial J}{\partial w_{nm}}=\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_n^{(i)}-y_n^{(i)})x_m^{(i)}\end{gathered}\tag{3.2}$$

显然可以发现的是，参数的优化方向是根据训练过程自动进行反向传播的，意味着每个分类器的优化机会是相同的，相同身份的人脸被拉到一起，不同身份的人脸则推开，以求最小化损失，所以本文方法的合理性就在于可以将 FC 层本身看成多个分类器，对于小规模的数据集，完整的 FC 层存储了每个类别的分类器，在这个里面每次梯度下降的时候可以优化自己本类的分类器，但是面对大规模的数据集会导致全部的 FC 层存储到里面占据难以估量的硬件资源，于是想到每次更新该分类器的时候将该分类器添加到 FC 层，也就是我们的 DCP 里面，这样能保证更新的正确性，同时较合理的 DCP 的大小又能很好的节省硬件资源，那么全部分类器的更新方式就变为公式 3.3，DCP 内部更新分类器。

$$\begin{gathered}\frac{\partial J}{\partial L^{(i)}}=\frac{1}{N}\quad\frac{\partial L}{\partial z_n}=\sum_{i=1}^{C}\frac{\partial L}{\partial \hat{y}_j}\frac{\hat{y}_j}{\partial z_n}=(\hat{y}_n-y_n)\quad\frac{\partial z_n}{\partial w_{nm}}=x_m\\ \frac{\partial J}{\partial w_{nm}}=\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_n^{(i)}-st_n y_n^{(i)})x_m^{(i)}\end{gathered}\tag{3.3}$$

相较于原本的方法，该方法加入了一个标记数组当分类器存在于 DCP 里面的时候，该标记为 1，否则为 0，这样的设计能保证不在 DCP 内部的参数不更新，在 DCP 内部的参数正常的更新，因为我们的加载器的设计是先通过实例加载一半，再通过类别加载一半，其中一个批次用于更新 DCP，更新之后批次里面按照类别加载的一半将会出现在 DCP 里面，即保证正样本的存在，同时因为实例数据的一半是随机加载的，平衡了正负样本，因为这一半的批次大概率不会出现在队列里面，因为队列的大小相较于总的类别数小很多。

3.3 FC 和 FFC 方法比较

在深度学习领域，全连接层(Fully Connected Layer, FC)是神经网络架构中最基础也是最常用的组件之一。但是，随着模型规模的扩大和数据量的增长，传统是全连接层难以实现正常的训练过程，FFC 方法作为 FC 的一种优化变体，它通过引入动态类池管理机制来优化特征向量处理流程，在保持合理精度的前提

下显著提升训练效率。FFC 构建了一种动态向量池来动态管理其中的向量，通过 LRU 策略实现其中一直保持着相对合理的一些向量。具体一点的实现是，该方法设计了一个固定长度的队列，动态存储最近出现的样本特征，并在训练时与当前批次的特征进行相似性对比和梯度更新。当批次里面存在，动态向量池里面不存在的向量，我们称之为负样本，通过这个动态向量池 FFC 有效提升了负样本的丰富性，避免了 FC 层中因类别过多导致的稀疏更新问题，负样本的存在也拉开了类与类之间的距离。

FFC 方法在牺牲极小的准确率前提下，训练时间显著缩短。例如在百万 ID 规模的人脸识别数据集上，FFC 的单轮训练时间相比传统 FC 方法可减少 30%~50%，特别适合资源受限环境或需要快速迭代的应用场景。

FFC 保留一部分核心特征在特征池中。每次训练时将当前批次的特征送入池中替换最旧特征，结合硬负样本挖掘策略，仅针对最具挑战的负样本计算梯度。这种策略在保持训练稳定性的同时，大幅减少了模型在内存与计算上的开销。特别是在数据分布稀疏或类别极多的情况下，FFC 能显著加快训练速度，同时带来较强的泛化能力。

FC 以类别为中心进行优化，依赖于完整的类别权重矩阵更新；而 FFC 则以特征为中心，通过动态样本队列构造对比对，从而实现了对训练效率的显著优化。FFC 是一种以牺牲部分精度为代价，在训练速度和资源使用上实现权衡的有效策略，特别适用于大规模人脸识别系统的快速原型开发和迭代。

3.4 LRU 策略

这个策略是 DCP 能够实现的核心，该方法使用链表和集合进行实现，首先，LRU 里面存在一个集合，存储的是当前在 LRU 里面的所有的人脸和对应的链表中的位置，该集合的存在方便我们对动态池操作，当每次新增键值对的的时候，相应的该字典也会增加信息，删除键值对的时候，该字典也会删除，链表创建的是双向的链表，其主要的信息是，DCP 的大小，键值对字典，操作栈，主要的函数包含，get 函数，当发现当前访问的图片不在键值对字典里面的时候加入链表的头节点，将该信息写入键值对字典，当个访问的图片在 DCP 里面的时候将该图片放到链表的头节点， try_get 函数，当发现当前访问的图片不在键值对字典里面的时候加入链表的头节点，将该信息写入键值对字典，同时操作栈记录当前的操作，方便回滚的时候删除这个加入的节点，当个访问的图片在 DCP 里面的时候将该图片放到链表的头节点，操作栈记录操作，回滚的时候将这个节点放回一开始的位置，这是 LRU 操作的两个核心的函数，然后构建 LRU 的迭代器，从头节点开始一直往后读取直到尾节点，并进行输出，需要注意的是 get 插入 DCP 的时候需要判断这个插入的信息是不是会导致队列溢出，要是加入新的信

息导致队列溢出，需要删除当前队列的尾节点，保持队列大小始终小于等于默认设置的队列容量，LRU 类还设置了一些常用的类函数，包含清空 DCP，按照键查询对应的值。

最值得注意的一点就是，LRU 实现了回滚的操作，因为想要使用回滚操作的时候，需要使用 try_get 函数查询和添加新的类别到 DCP 里面，每次 try_get 操作都会更改对应的操作栈信息，每次回滚能得到上一步的 DCP 的状态，这样的话，进行对比训练损失的时候非常的方便，第一次计算损失使用 tr_get 函数更新 DCP，然后计算完损失之后直接进行回滚，回滚当前的 DCP 到历史状态，进行第二次损失的计算，对比损失的计算能提高模型的稳定性，第二次损失计算的时候直接使用 get 进行 DCP 更新即可，这次更新进去 DCP 里面的类别作为新更新的特征为下一个批次使用。

3.5 损失函数设计

在人脸识别任务中，损失函数的设计对于学习具有判别性的特征起着至关重要的作用。最基本的分类任务中常使用的损失函数是 Softmax 损失函数，它通过最小化预测分布与真实标签分布之间的交叉熵来优化模型。然而，传统的 Softmax 损失在特征分布的角度来看，其优化目标并未对类间间隔与类内紧致性施加明确约束，这对于人脸识别等需要高判别性特征的任务而言是远远不够的。因此，研究者提出了多种改进版本的 Softmax 损失函数，主要包括 SphereFace、CosFace 和 ArcFace，这些损失函数通过引入角度约束，使模型能够学习到具有更强判别性的特征。

SphereFace 是最早提出的角度 margin softmax 损失函数之一。该方法从几何角度出发，认为人脸特征应该在单位超球面上分布，因此在 Softmax 的基础上引入了角度 margin，将类别之间的区分从欧式距离空间转向角度空间。SphereFace 使用多倍角度函数来增加类间角度间隔，其优化目标是将同类样本的角度靠得更近，同时使不同类样本的角度尽可能远离。尽管 SphereFace 提出了新颖的角度 margin 思路，但由于多倍角度函数在优化过程中存在数值不稳定和收敛困难的问题，因此实际训练较为困难，对学习率和初始化敏感。

为了解决 SphereFace 的优化困难问题，CosFace 引入了更为简单且稳定的 margin 设计。CosFace 不再使用角度多倍函数，而是直接在 cosine 相似度上减去一个固定 margin，即 cosine(theta) – m。CosFace 的目标是使不同类之间的 cosine 相似度具有明确的 margin，从而提升判别能力。该方法在实际训练中比 SphereFace 更加稳定，并且可以更有效地促进类间分离和类内紧致性。CosFace 的提出为后续角度 margin 设计提供了更好的方向。

ArcFace 在 CosFace 的基础上进一步改进，通过引入角度空间中的加性

margin，实现更加直观和几何一致的优化目标。具体来说，ArcFace 先将特征和权重归一化，使得分类的依据完全转为角度的大小，然后在角度上加上一个固定的角度 margin m。由于这种设计直接作用于角度空间，使得每个类别之间的判别间隔更具有几何解释性和一致性。ArcFace 的优化目标可以被表述为最大化不同类别之间的角度差异，并收紧同一类别样本在角度上的聚合度，这种设计非常契合人脸识别的本质需求，即在角度空间学习具有强判别性的人脸表示。

ArcFace 损失函数的实现，对正样本使用正常的交叉熵损失，对于负样本，采用了困难样本挑选机制，这样的设计能够间距类内紧凑和类间分离。

对于传进来的正样本，先从 cos_theta 里面，将正样本对应的条目选出来，cos_theta 是模型输出的 logits，他的大小是（batch_size, queue_size）这正是 FFC 的优化点，即类别为 DCP 的大小。选出其对应的标签，因为 FFC 方法主要的思路就是对于正样本我们采用计算交叉熵的方法计算损失，所以最后对正样本使用了交叉熵损失函数，在这之前需要加上 margin m, 先通过张量索引选出每个样本对应的正确类别上的余弦值 gt，正确类别的 logits。使用角度转换，需要先计算出 sin(theta) 这样的话使用角度展开公式就能将 cos(theta) 转化为 cos(theta + m) 在内部加上一个角边距，这个之前讨论过，这样能够增大类别之间的距离，对于超球体，增大测地线的长度，是得符合类间更小，类内更大的要求。这一步是 ArcFace 的核心，这样的替换，使得模型在训练的过程中不得不拉大与其他类别的角度差距，增加判别能力。

将处理之后的 margin 填回对应位置，构成新的 logits，输入到交叉熵函数里面形成分类损失。

对于负样本，因为 FFC 的实现比较特殊，使得未出现在 DCP 里面的特征还是比较多的，要是放肆不管这些样本，会导致类间的判别能力非常的小，甚至训练结果出现阈值非常的大，大量的样本的余弦相似度都很大，难以区分是否属于同一张人脸，因为余弦相似度都挤在接近 1 的部分。

所以对付样本的处理再结合 ArcFace 才能的结合实现，才能让 FFC 有很好的效果，对于负样本，现根据他和当前样本的相似度进行排序，选出 top_K 个最大值作为困难负样本，提取这些困难负样本的 logits，最后对这些 logits 求均值，作为负样本损失的一部分。

这样设计的目标在于，对于负样本，无法使用明确的标签进行监督，但是仍可迫使其输出不要接近于任何的现有类别。抑制其对某个类别的倾向，防止干扰主干正样本的特征学习。

整体的损失设置为这两个损失的和，正样本使用 ArcFace 损失函数，负样本让其尽量不干扰正样本的特征学习，提升整体的鲁棒性。

3.6 网络搭建

MobileFaceNet 是一种专为人脸识别任务设计的轻量级深度神经网络，其核心目标是在保持较高识别精度的同时，显著降低模型参数量和计算开销，适用于移动设备或嵌入式系统等计算资源受限的环境。非常适合我们的 FFC 的初衷，因为本身提出 FFC 的原因就是想要加快训练的过程，假如使用了更加深层次网络结构，相比于该网络结构更加的浪费时间，MobileFaceNet 在模块的设计方面采用了大量的深度可分离卷积，在实现高效计算的同时保证了特征表达能力。

MobileFaceNet 的输入为 RGB 图像，大小通常为 112x112。网络最开始由一个标准卷积块和一个深度可分离卷积块构成，用于初步提取低级图像特征。其中第一个卷积块将通道数从 3 提升至 64，步长为 2，完成空间尺寸的下采样；随后使用一个深度可分离卷积保持通道数不变，同时进一步提炼特征。

标准卷积利用多通道卷积核对输入的多通道图像进行处理，输出的特征图既提取了通道特征又提取了空间特征，逐深度卷积与标准卷积的区别在于，逐深度卷积的卷积核为单通道模式，需要对输入的每一个通道进行卷积，这样就会得到和输入特征图通道数一致的输出特征图。即输入特征图通道数等于卷积核个数等于输出特征图个数。

逐点卷积，根据逐深度卷积可知，输入特征图通道数等于卷积核个数等于输出特征图个数，这样会导致输出的特征图个数过少（或者说输出特征图的通道数过少），从而可能影响信息的有效性。此时，就需要进行逐点卷积，实质上是用 1x1 的卷积核进行升维。1x1 的卷积核主要作用是对特征图进行升维和降维，深度可分离卷积是逐深度卷积与逐点卷积的结合，先对每个 channel 进行逐深度卷积，然后再通过逐点卷积合并所有 channels 为输出特征图，从而达到减小计算量、提升计算效率的目的。深度可分离卷积的参数量和计算量是逐深度卷积与逐点卷积之和，相较于一般卷积有较大的提升。

接下来的主干网络由多个 BottleNeck 模块组成，先使用 1x1 卷积进行通道扩张，再通过 3x3 深度可分离卷积[12]提取空间特征，最后用 1x1 卷积将通道压缩回目标输出维度。

每个模块后接 BatchNorm 和 PReLU 激活函数，提升了网络的非线性表达能力，加快了收敛的速度。在具体实现中，MobileFaceNet 定义了网络中每个阶段的结构参数，包括通道扩展倍数、输出通道数、重复次数和步长，方便网络模块的搭建，只需要更改模块里面的参数即可简单的修改整个网络的结构。整个瓶颈结构逐层构建，层数较深，有较强的判别能力，初始的参数如下图 3.3，简单修改初始的设置就能构建更深的网络，方便模型的搭建。

主干网络输出特征图后，MobileFaceNet 通过一个 1x1 卷积将通道数提升

至 512，之后衔接了一个 7x7 的深度可分离卷积压缩空间的信息。随后，再使用一个线性卷积将维度压缩至设定的特征维度，得到图像的特征向量的表示，作为最终输出的人脸特征嵌入。该嵌入特征经过归一化操作，得到单位范数的特征向量，便于后续计算余弦相似度或应用到度量学习任务中。

MobileFaceNet 在准确性和效率之间取得了良好平衡。设计了较为精巧的模块，能有效的提取图片的特征，除此之外，其网络的构架也没有太深，是一个轻量级人脸识别系统的经典架构，主要能应用在硬件资源受限的情况，在这样的情况下也能高效的得到鲁棒性较强的人脸识别模型。

主要使用该网络的原因是他与本文的 FFC 方法非常的符合，都是在尽量节省硬件资源的条件下得到较好的效果，于是本文的采用了该网络作为主要的训练网络，后面衔接了 FFC 里面的 DCP 作为动态队列池，对于网络生成的特征向量，先进性更新 DCP 然后进行损失的计算，整个过程朝向损失下降的方向优化，最终得到正确率还算不错的模型。

表 3-1 网络结构表

Type	Filters/Channels	Kernel/Stride	Output（H×W×C）
ConvBlock	3 → 64	3×3 / 2	56×56×64
DWConvBlock	64 → 64	3×3 / 1 (DW)	56×56×64
BottleNeck×5	64 → 64	3×3 / 2 (t=2)	28×28×64
BottleNeck×1	64 → 128	3×3 / 2 (t=4)	14×14×128
BottleNeck×6	128 → 128	3×3 / 1 (t=2)	14×14×128
BottleNeck×1	128 → 128	3×3 / 2 (t=4)	7×7×128
BottleNeck×2	128 → 128	3×3 / 1 (t=2)	7×7×128
ConvBlock	128 → 512	1×1 / 1	7×7×512
DWConvBlock	512 → 512	7×7 / 1 (DW)	1×1×512
ConvBlock	512 → 512	1×1 / 1	1×1×512

除了当前的训练网络，还设计了 resnet[13] 网络，也可作为训练网络。resnet

网络通过引入跳跃连接，有效缓解了深度神经网络中的梯度消失/爆炸问题，使得网络可以训练得更深（如超过 1000 层），同时保持较高的性能。传统深度网络随着层数增加，训练误差可能不降反升，出现退化问题。resnet 提出残差结构，让网络学习输入与输出之间的残差而非直接映射。输入 x 直接与残差 F(x) 相加，使得网络可以更容易学习恒等映射，避免深层网络的训练困难。

3.7 本章小结

本章节提出了该论文的核心，介绍了 FFC 方法的具体实现细节，主要包含 LRU（最近最少使用）策略实现动态缓存、损失函数设计与网络结构构建三个部分。LRU 策略致力于在人脸识别任务中构建高效的特征更新机制与优化策略，以提升模型在大规模身份分类任务中的表现，本文借助该策略算设计了类中心的动态管理策略。通过这种策略，模型能够在有限的硬件资源下动态管理类别特征，保持类中心的更新及时性与代表性，从而保障了训练过程中损失函数的有效计算。

LRU 策略是 FFC 的核心组成部分之一，模拟了有限容量动态队列池（DCP）的行为，通过双向链表和哈希集合共同构建。在每次特征访问时，若当前样本已存在于 DCP 中，则其位置被移动至头部；若不存在，则插入头部，并在队列满时自动移除尾部节点以维持容量。相比传统队列结构，该策略不仅支持按需访问，还提供了 try_get 与 get 两种关键接口，其中 try_get 支持操作栈回滚功能，使得训练过程中的两阶段对比损失计算更加灵活可控。通过这种机制，模型可以在第一次前向传播更新特征池后，在不影响后续训练状态的前提下计算另一种损失，有效提高模型的稳定性和泛化能力。

在损失函数设计上，本文引入了 ArcFace 和 CosFace 作为主损失函数，对正样本引入角度 margin，促使同类样本更加聚合、不同类样本更易区分。具体操作是对 logits 中正类对应的位置进行角度转换，将原始的 cos(theta) 替换为 cos(theta + m)，增强了类间区分性。而对于负样本，FFC 机制下存在大量未见类，若忽视这些信息将导致类别判别能力退化。因此，本文通过困难负样本挖掘策略，筛选出相似度最高的 top-K 非目标类样本，计算其平均损失并加权叠加在总损失中，从而提升负样本的约束能力。

本文采用 MobileFaceNet 作为主干网络。该网络以轻量级设计为核心，结合深度可分离卷积与 1×1 卷积实现通道扩展与压缩，既保持了特征表达能力，又显著降低了参数规模和计算开销。MobileFaceNet 能在计算资源受限的设备上高效运行，适用于本文追求训练加速与内存节省的目标。在主干网络提取出归一化的特征嵌入后，这些特征将用于更新 FFC 的 DCP，并参与损失计算与参数优化。此外，本文还支持使用更深的 ResNet 作为备选网络结构，以在资源允许的条件下进一步提升性能。

第四章 实验分析

4.1 数据集创建

因为本文使用的数据集比较的庞大，所以读取图片本身就是一个很大的时间消耗，于是本文采用 LMDB 数据库的形式存储图片，LMDB（Lightning Memory-Mapped Database）是一种高性能、低开销的键值存储数据库，特别适合机器学习的场景，通过直接的内存映射访问数据，避免了传统文件的 I/O 开销，直接读取在频繁读取大量的小文件的时候也就是机器学习的场景的时候性能较差，LMDB 有效了减少了时间的消耗，LMDB 读取数据的时候，数据按照需要加载进内存，不会一次性占用过多的 RAM，并且库函数 mxnet 支持直接的二进制文件转化为 LMDB 类型的文件，所以得到二进制的数据集之后，可以省去提取图片的步骤，直接转化，因为 resize 函数操作比较慢，所以想要发挥 LMDB 数据库的全部性能最好的方法就是先观察自己网络的图片大小的输入，然后转化 LMDB 数据库的时候，按照自己网络要求的输入来选择适合大小的图片存储进去 LMDB 里面，可以直接读出文件来使用，减少了很多操作的时间消耗。下面是直接加载图片和 LMDB 加载图片的时间消耗对比，可以发现两个方法的时间消耗都是随着照片的数量线性增加的，同时重要的是，LMDB 显著的减小的时间的消耗，也就是他花费的时间显著的随时间的增大而减小的更多。这就是为什么我们的训练过程采用 LMDB 的方式来存储数据。不仅 LMDB 是时间消耗比较少，而且 LMDB 方式读取照片非常的方便，正如前面所说的只要在创建 LMDB 数据库的时候存储指定大小的图片，这样读取后不需要对图片进行操作，LMDB 就能发挥他的全部性能。

LMDB 的核心原理为，LMDB 将数据库文件映射到进程的虚拟内存空间，操作系统负责按需加载数据页面，无需手动管理缓存，读取的时候直接访问地址，避免 read() 系统调用，速度接近于 RAM，数据以键值对形式存储，并通过 B+ 树索引，保证 O(logn) 的查询效率，在 LMDB 里面存储的键通常就是图片的编号，值就是图片的二进制数据，所以使用 LMDB 数据库的时候通常需要伴随着使用一个 KV 文件，该文件存储的键值对是编号和标签，每次需要读取图片数据的时候，先在 LMDB 数据库里面读取文件的二进制数据，转化为图片，然后通过相同或的编号从 KV 字典里面读取文件的标签，这样就能得到图片的所有数据了，大大降低了图片的加载时间。

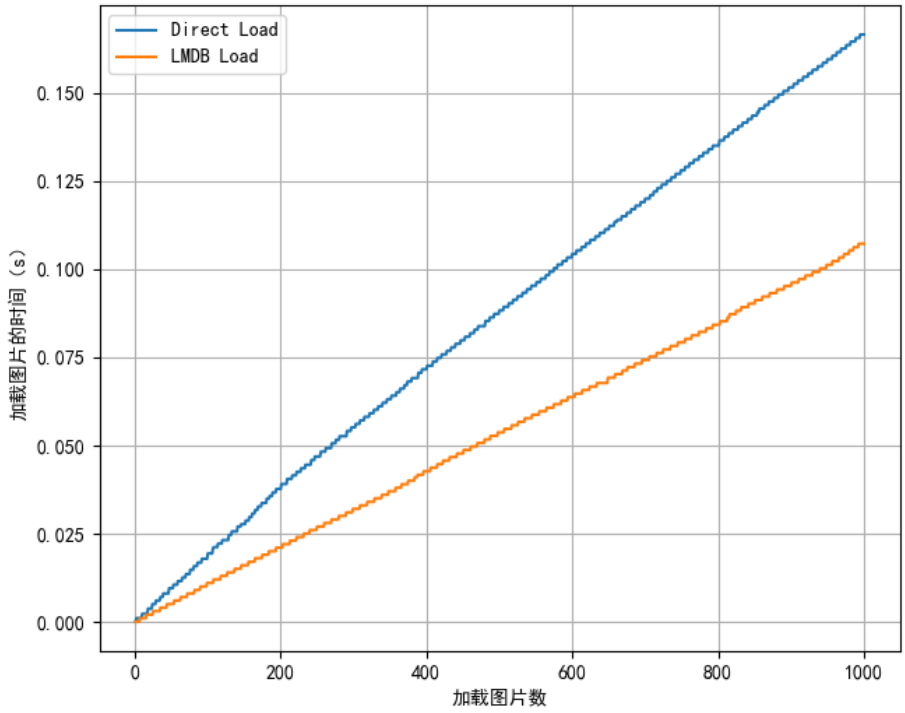

图 4.1 图像加载时间对比

创建过程中，因为 LMDB 自带的函数需要先指定想要创建的 LMDB 数据库的大小，直接指定大小的话，太大的话会占用本不该占用的空间，太小的话创建过程中会出现问题，于是创建 LMDB 数据库的过程中可以采用动态创建的方式。

每次当空间不够抛出异常的时候扩大 LMDB 的大小，这样的话能够构建出大小合适的数据库。

4.2 数据加载器

论文里面提到了两个网络，称为 P 网络和 G 网络，前者主要管理动态池，使用类别加载器来根据类别获取我们想要的人脸数据. 实例加载器获取的人脸数据还是随机获取的，这样将读取到的 batch_size 大小的人脸类别加载器和 实例加载器分别读取了一个 batch_size）拆分之后合并就能得到两个分别包含了一半 id_loadr 一半 instance_loader 的人脸数据，以实现正负样本均衡，保证每个 batch_size 都会有至少一半的正样本，因为 id_loader 分开了一半。每张图片读取之前会先进行随机剪切，然后进行归一化处理，将图片上面的像素值归一化到 -1 到 1 之间。

下面详细的介绍两个加载器，对于类别加载器，从 LMDB 数据库里面进行读取，根据 label 加载图片，实例加载器是随机加载的，所以构建类别加载器的时候需要构建一个 label 字典，在字典里面的就是我们已经读取到的 label，否则就是新的 label 要加到这个字典里面。每个类别的图片都放在他自己类别为标

号的列表里面，每次从每个类别列表里面随机加载两张图片，进行随即剪切和归一化即可。

上面的步骤实现之后就能实现我们的数据加载器的加载过程，然后分为两个批次进行更新 DCP 和优化网络参数，加载完数据之后执行训练的过程如下图 4.2。

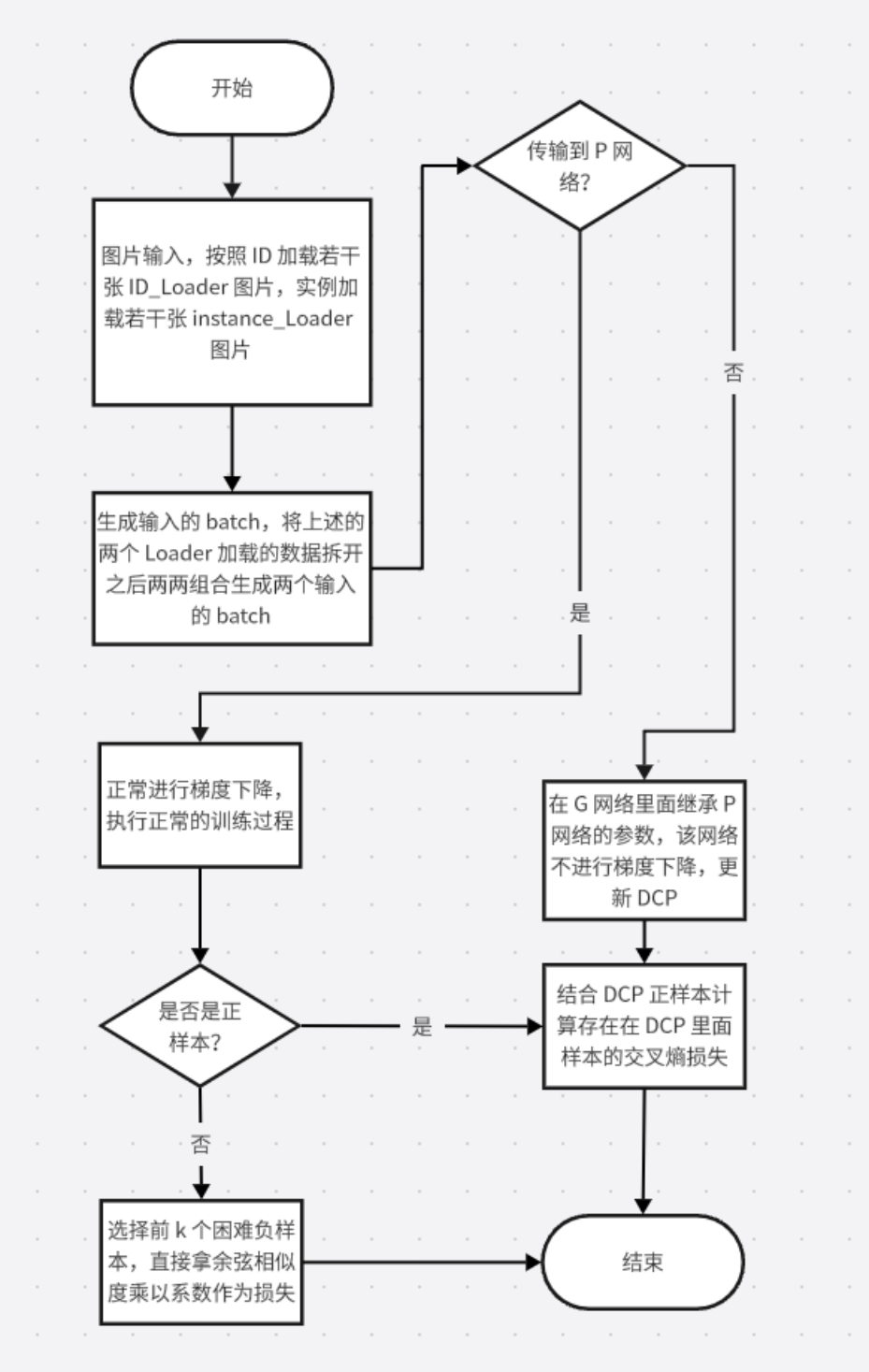

图 4.2 训练流程图

4.3 训练介绍

基于上面的架构搭建的网络，先使用 ArcFace 作为损失函数，训练得到的损失函数曲线如下，可以发现训练的过程中，FFC 的波动非常的大，因为每次迭代的过程当前的批次内存在的正样本的个数非常的不确定，即使保证了正样本的存在也会有很大的波动，其次就是 FFC 最终收敛时候的损失是 20 上下，相较于 FC 训练过程，收敛时候的损失大了一点，可见相同情况下的训练过程，FFC 表现的可能不如 FC，因为本身 FFC 设计的初衷就是空间换时间。相应的时间消耗的曲线可以参考图 1.1，下面将在训练得到的模型的基础上判断相应的较好的模型的正确率和阈值表现。

ArcFace 的方法上文简单的介绍过，其本质是在角度上面加上了一个 margin，因为我们先对所有的特征向量进行了归一化，所以这个添加的角度间隔就变为了两个特征在超球体上面的测地线距离，所以 ArcFace 有很好的几何解释，这也是为什么本文选用 ArcFace 作为损失函数的原因之一了。

文章中选用的度量函数都是有较好的几何解释并且效果比较好的损失函数，ArcFace 和 CosFace 相较于其他的损失函数具有方便代码的编写，方便调试，有较清楚的几何解释方便超参数的调整等优点，除此之外 ArcFace 和 CosFace 本身就有比较好的人脸识别分类效果。

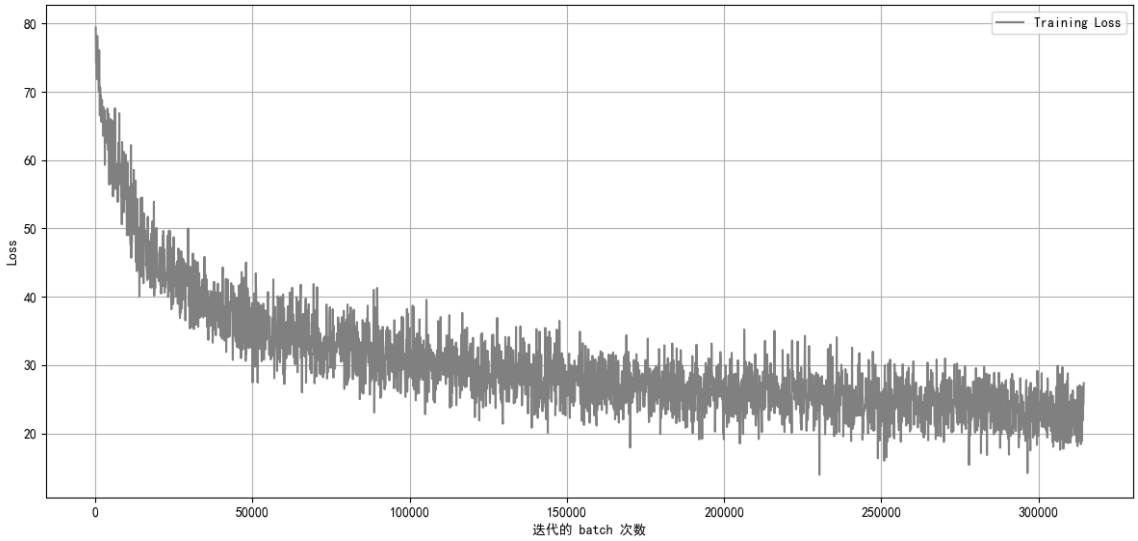

图 4.3 FFC 损失函数曲线

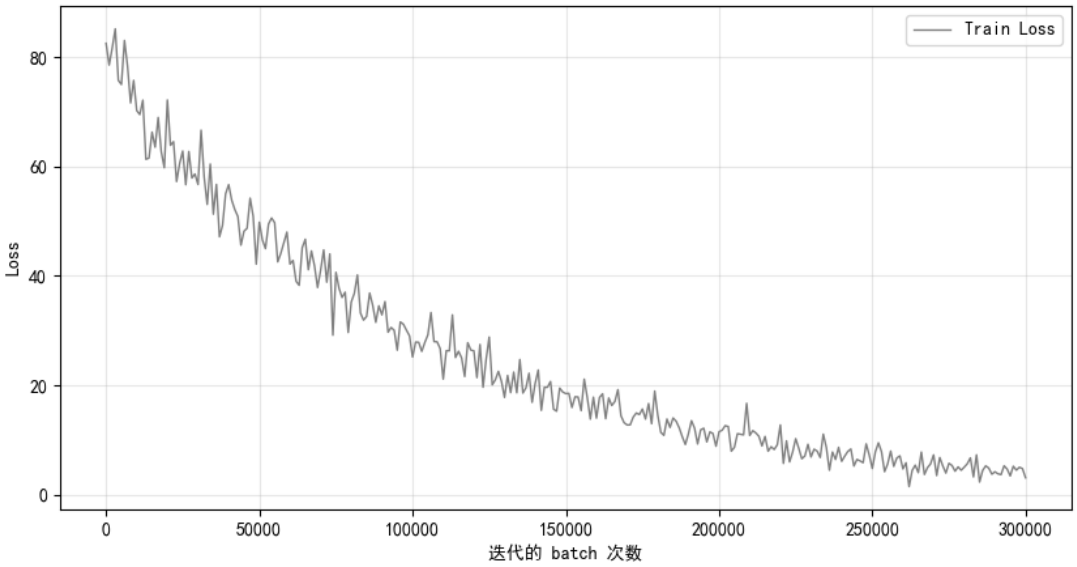

图 4.4 FC 损失函数曲线

FC 损失函数曲线，相较于上面的曲线变化程度要小很多，因为 FC 层保证了所有的类别都存在，所以损失函数相对就非常的平缓一点，因为 FFC 的设计可能导致正样本存在的数量不尽相同，存在的困难负样本也不尽相同，所以上面的曲线波动就非常的大，但是整体还是趋于收敛的。要注意 margin 的选取，不能太大和太小，除了这个参数 margin 之外 ArcFace 还有超参数 s，前者是为了增大不同类别之间的角间距，同时减小相同类别之间的角间距，后者是为了放大不同类别之间的不同性。因为对于超球体来说所有的余弦相似度都乘以一个超参数 s，会使得各种余弦相似度之间的差距被放大，这样的话也能更好的配合 margin 实现相同类别之间的余弦相似度更接近，不同类别之间的余弦相似度更加原理。因为 margin 和 s 的引入增加了模型的训练困难程度，所以模型会倾向于让样本靠近自己的类别来满足这个 margin 的限制，这就是 ArcFace 的原理，相较于一般的 Softmax，其能更好的提取人脸里面的特征。

4.4 测试集

本次实验采用的测试集为 lfw-algin-128，该数据集是基于国际通用人脸识别基准 LFW（Labeled Faces in the Wild）的优化版本。LFW 原始数据集包含 13,233 张非约束环境下采集的人脸图像，涵盖 5,749 个不同身份，图像具有显著的姿态、光照、遮挡和表情变化，能够有效验证模型在真实场景中的泛化能力。该数据集里面的数据经过 MTCNN 算法进行人脸检测和对齐，将关键点（双眼中心、鼻尖和嘴角）统一映射至标准位置，随后裁剪为 128 * 128 像素的归一化图像，并采用直方图均衡化消除光照差异。该测试集包含 6,000 对正负样本组合，其中正负样本比例均衡，每对图像均经过人工校验以确保标注准确性。评估协议

采用 10 折交叉验证，通过计算 6000 对图像的余弦相似度，以 ROC 曲线下面积（AUC）和平均准确率作为核心指标。lfw-algin-128 因为上面的特性成为了人脸识别领域重要的测试集，大量的人脸识别的论文都选用该测试集作为判断模型准确性的依据，本文也选用该数据集合作为模型准确率的依据。

4.5 最佳阈值选择

本次测试不仅要对比准确率，更重要的是要对比阈值，所以采用，绘制阈值图像的方式来评判模型的好坏，在此之前我们需要先计算出来模型的准确率和阈值，才好直观的得到阈值图像，准确率的计算包含下面的几个步骤，图像的加载和预处理，网络模型参数的加载，准确率的计算。准确率的计算过程需要按照如下的流程，对于所有的测试条目，先对比给出的两张图像的余弦相似度，然后根据标签（相同类别为 1，不同类别为 0），来动态进行阈值的选择，选择的依据就是那个阈值能对应更多的准确率就选用那个作为阈值，同时选择这个时候的准确率作为最终的准确率。

实现过程具体为，传进来预测值和真实值的列表，然后将阈值假定为每一个预测值，这样的话，如果发现两张图片的余弦相似度大于等于这个阈值的话，就认为他们两个是来自同一类的否则就认为他们两个不是来自同一类的，这样的话枚举完所有的预测值，就能找到一个最优秀的预测值作为最佳阈值，在这个阈值的条件下能够使的真实值和预测值的重叠最高。也就选择这个阈值作为当前模型的最佳阈值，算出这个阈值条件下的正确率作为这个模型的最佳正确率。

4.6 实验结果

下面两张图片就是按照这个方法进行绘制的相速度密度直方图和最佳的阈值，其中蓝色部分就是本来预测为相同，最终标签也是相同的，红色部分为本来预测为不同，最终标签也是不同的部分，深红色的就是预测错的部分，那么可以通过这个密度图对应的面积来计算正确率，在相同的条件下，FC 方法的正确率为 86.20%，对应的 FFC 方法得到的正确率为 85.53%。可见 FFC 在加快训练过程的前提下，损失了部分可以接受的精度，这也证明我们在时间和精度的取舍上面做了一定的平衡，和预期的实验结果非常的相近，FFC 牺牲了部分的精确度使得大大的减少了训练的时间和显存消耗。

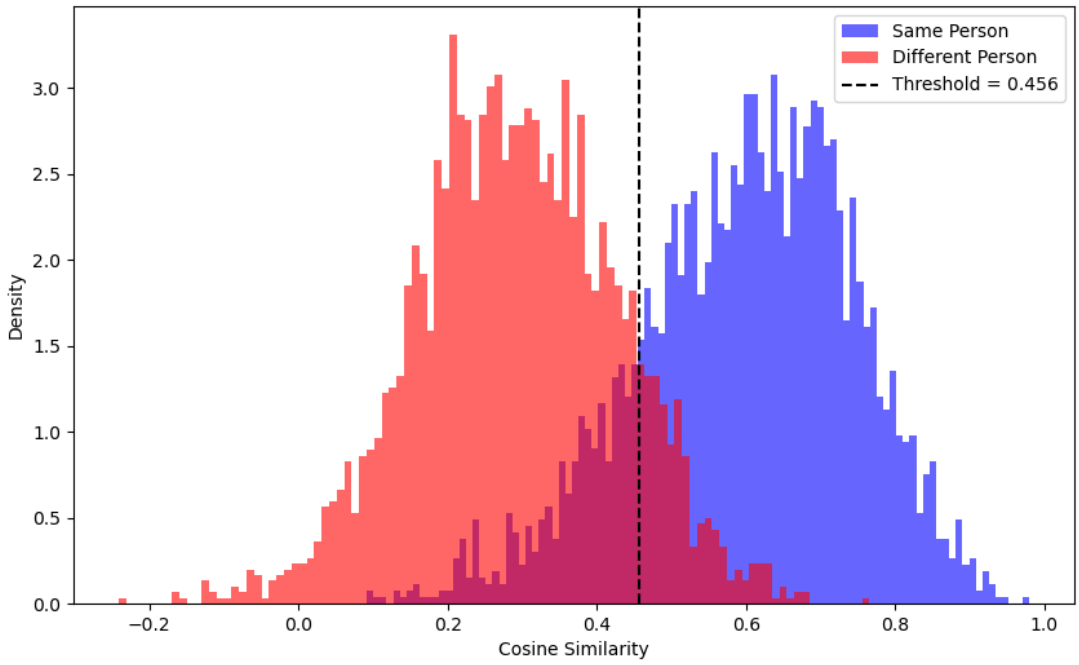

图 4.5 FC 训练得到模型的密度直方图

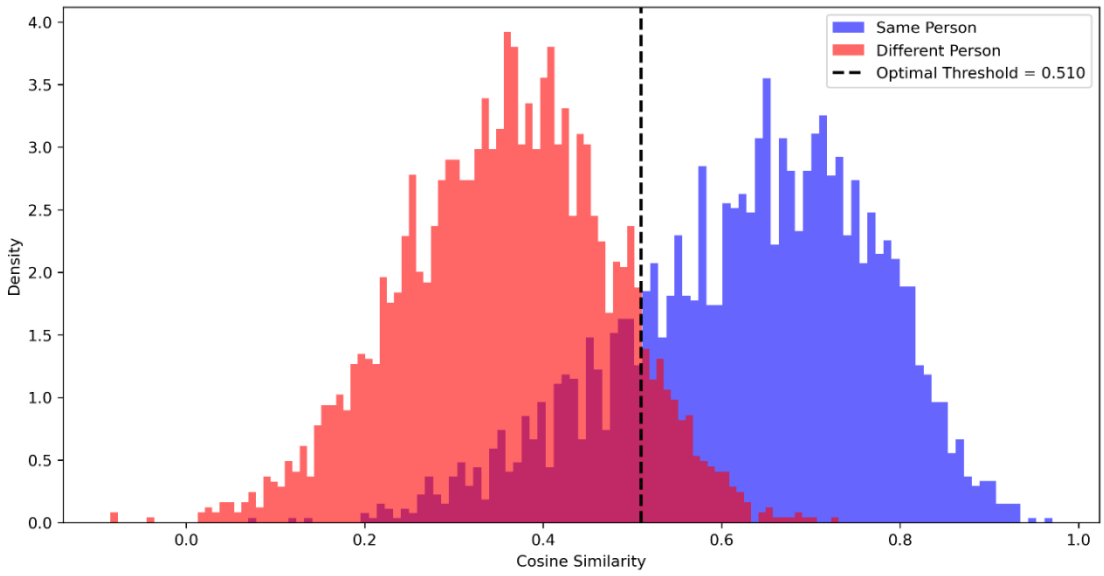

图 4.6 FFC 训练得到模型的密度直方图

可见两个训练方法的阈值选择也不相同，不同的训练方法会带来很多不同的方面，不同模型参数对应的密度分布图也有较大的差距。

在人脸识别系统的性能评估中，仅仅依赖“准确率”这一指标往往不足以全面衡量模型在真实环境下的泛化能力。特别是在开放集识别、弱监督或类间差异微小的识别场景中，模型可能出现偏向某一类的预测现象，导致虽然准确率看起来较高，但模型实际判断力有限。因此，我们引入四种经典的分类性能评价指标，分别是精确率（Precision）、召回率（Recall）、F1 分数（F1-score）以及 ROC 曲线下的面积（AUC），对人脸识别模型的性能进行多角度、系统性的分析与量化评估。

准确率的判断上面已经做了介绍了，并且给出了密度直方图来直观的判断模型的好坏。

下面介绍精确率，精确率是所有判定为同一个人的图像对中多少真正是属于同一个人的，精确率越高，表示模型的误报越少，也就是上面的密度直方图里面蓝色的面积与阈值线右边总面积的比例。阈值曲线的右侧都判定为一个人，红色部分为判断失误的部分，蓝色是判断正确的部分。

召回率，召回率的定义为，在所有实际为真的图像对中，模型成功识别出来多少，也就上蓝色部分与阈值曲线左边深红色部分和蓝色部分的和的比值，召回率越高说明模型漏报越少，当前的模型更擅长捕捉全部的正类。

F1 分数是精确率与召回率的调和平均值，是两者之间的折中指标，当一个系统在精确率和召回率之间权衡的时候，F1 是一个统一衡量综合性能的指标。F1 分数越高，表示模型在准确率和召回率之间取得了良好平衡。

ROC 曲线，是二分类任务中模型性能评估的重要曲线，综合真正率和假正率，展现出了真正率（召回率）和假正率之间的关系，是评价模型好坏的重要指标，现在我们的模型测试阶段可以简单的看成一个二分类的模型，通过计算模型的特征向量的相似度来判断人脸是否属于同一个人。真正率表示所有真实为正的样本中，模型正确识别出的比例。假正率表示所有真实为负的样本中，模型错误地判定为正的比例。

ROC 曲线的横轴为假正率，纵轴为真正率。通过不断改变模型的判别阈值，计算出一系列的（假正率，真正率）点并连接这些点，就得到了 ROC 曲线。一个模型应该是要求在真正率较高的情况下保证较低的假正率，于是模型的 ROC 曲线应该尽量趋向于左上角，AUC 是 ROC 曲线下的面积值，显而易见，他越接近于 1，效果越好，下面是两个方式训练的模型对应的 ROC 曲线和一些相关的指标对比

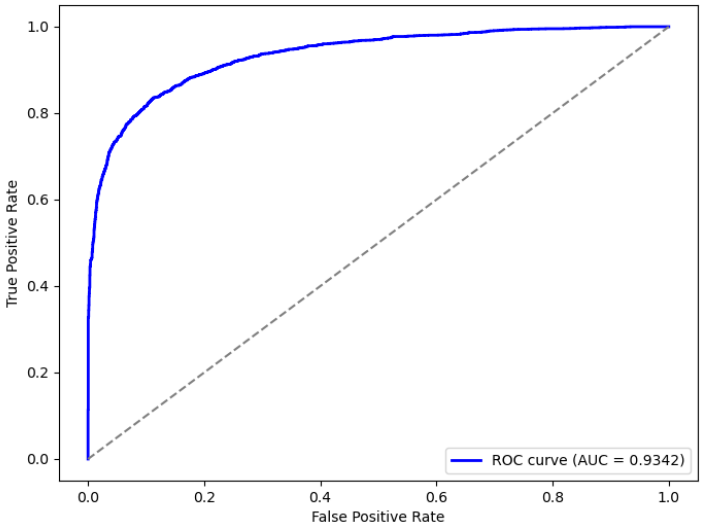

图 4.7 FC ROC 曲线图

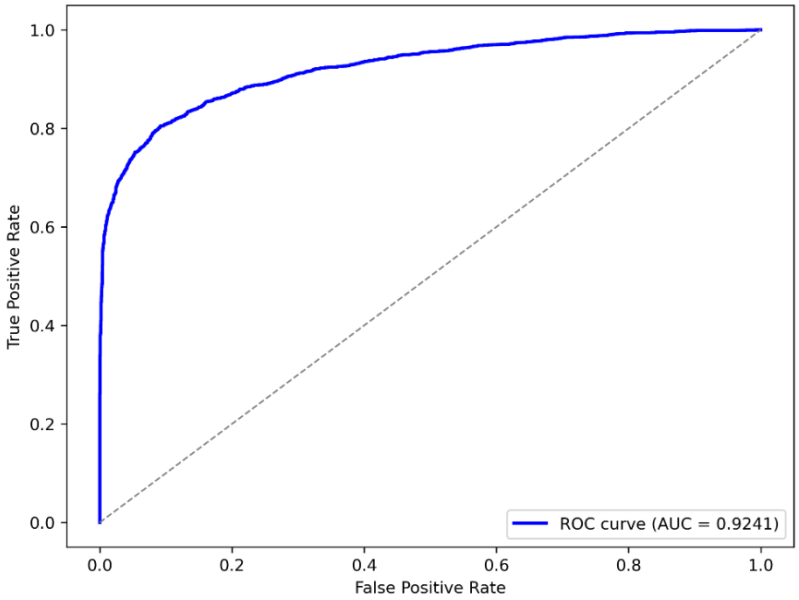

图 4.8 FFC ROC 曲线图

表 4-1 方法效果对比表

方法\指标	准确率	精确率	召回率	F1 分数	AUC
FC	86.20%	88.51%	83.57%	85.80%	93.42%
FFC	85.53%	89.80%	80.43%	84.86%	92.41%

分析上面的方法可以发现，FFC 在预测是否为同一个人上面更加的谨慎，这导致他的精确率很高，但是也正是因为这个谨慎，导致他漏掉了很多的本应该的同一个类别的样本，F1 分数作为精确率和召回率的权衡，发现两者的差距不大，但是 FFC 显著的减少了时间的消耗。AUC 曲线的差距也不大，FC 更可能在随机给定阈值的情况下分类出所有的样本。

4.7 本章小结

本章围绕人脸识别任务展开了一系列实验，旨在评估本文提出方法在实际应用中的表现。首先，为了高效处理大规模图像数据，本文采用了 LMDB 数据库作为数据存储形式。LMDB 通过内存映射机制显著减少了传统 I/O 操作的开销，尤其适用于频繁读取大量小文件的机器学习场景。实验证明，与传统图像读取方式相比，LMDB 在读取速度方面具有明显优势，尤其在图像数量逐渐增大时，其优势更加明显。此外，为进一步优化性能，本文在构建 LMDB 时采用了动态空间分配策略，避免因容量估算不准确导致的资源浪费或存储失败。

在数据加载方面，本文设计了两种数据加载器：类别加载器与实例加载器。类别加载器依据标签提取指定类别的图像，实例加载器则随机抽取图像样本，二者结合后可保证每个训练批次中包含充分的正负样本，有利于模型训练的稳定性。

数据在加载前还会进行随机剪裁与归一化处理，使模型更具鲁棒性。

测试部分选用的是广泛应用于人脸识别领域的 LFW-aligned-128 数据集。该数据集基于 MTCNN 对原始图像进行了关键点对齐与标准化处理，增强了数据质量，并提供了 6000 对正负样本，用于模型评估。测试流程包括图像预处理、特征提取、相似度计算、最优阈值选择与指标统计。通过计算图像对之间的余弦相似度，本文采用最佳阈值下的准确率作为主评估标准，并补充精确率、召回率、F1 分数和 ROC 曲线下面积（AUC）四个指标，从多个角度评估模型性能。

实验结果表明，传统 FC 方法在准确率上略高于本文提出的 FFC 方法（86.20% vs. 85.53%），但 FFC 显著减少了训练时间和内存占用，达到了效率与精度之间的良好权衡。从更细致的评估指标看，FFC 在精确率上表现更好，说明模型在判定为同一个人时更为谨慎，误报率更低。然而，由于其较为保守的判断策略，也导致召回率略低，即漏检率偏高。但从 F1 分数和 AUC 指标看，FFC 与 FC 差距不大，说明其整体识别能力仍具有较强的稳定性和实用价值。

综上所述，本文通过 LMDB 优化数据加载效率，结合双加载器提升样本质量，同时在评估阶段引入多指标分析，全面验证了 FFC 框架在保障识别准确性的同时大幅降低资源消耗的能力。这为面向大规模人脸识别任务的系统设计提供了有力支撑，也验证了本文方法的实用性和可行性。

第五章 总结与展望

5.1 本文主要工作

本文先了解到了人脸识别训练过程中出现的难题，即在面对数据越来越大的数据集的时候，人脸识别对于硬件的要求会越来越高，在这样的背景下，本文使用了一个新的方法使得大规模的人脸识别数据能减少对硬件的依赖，主要的方法，本文称之为 FFC，在这个方法下人脸识别的 FC 层的参数量不再和人脸的类别数成正比，而是引入了一个超参数，即动态队列池的大小，我们可以自己设置 DCP 里面存储多少的类别，而不是面对超级大的数据集根本无法训练。因为主干网络的参数量基本是固定的，所以 FC 的参数量能严重影响网络里面的总的参数，无论是从时间还是空间的角度考虑，面对较大的数据集，使用一个动态的类中心池都是较优的选择，不仅需要构建出来这个 DCP 池，更重要的是对这个 DCP 的管理要符合逻辑。

文章的中间部分介绍了管理 DCP 池的方法，即 LRU 策略，通过双向链表和一个字典，我们能够很简单的实现，对于经常出现的样本，让他较长时间的甚至一直出现在 DCP 里面，而对于出现的次数较少的样本，我们也给了他出现在 DCP 里面的机会，也就是优化他自己类的机会。具体来说，LRU 采用的就是这样一个策略，新访问的人脸特征不管有没有出现在 DCP 了都放到开头的位置，本来就在 DCP 里面的话就移动到开始的位置，否则的话就直接添加到开始的位置，这样当队列的长度超过他自己的上限的时候删掉这个队列的末尾，保证队列大小稳定的情况下，让经常出现的样本一直出现在 DCP 里面，不经常出现的样本随着其他样本的提前会慢慢的放到 DCP 的末尾，随着 DCP 容量的增大会被删除。

损失函数选用的是 ArcFace 用来计算交叉熵损失计算出现在 DCP 里面的特征的损失，采用 top-K 困难样本计算没有出现在 DCP 里面的特征的损失，结合这两个损失的计算来使得类间的间距尽量的大，类内的间距尽可能小。

文章的后面使用训练得到的模型计算了一些指标来判断 FC 方法和 FFC 方法的优劣，不考虑训练时间和硬件资源占用的情况下，FC 方法占据很小的优势，因为他考虑了全部的类别，不用麻烦的更新 DCP，在其他的指标上面比如精确率，召回率，F1 分数，AUC 方面，两者相差的并不大，而且可以发现 FFC 对于判断两个类别是否属于一个类别相当的谨慎，以至于错过了很多的本来是同一类的样本，看图 3.8 也能看出来一些。

经过这些实验，可以得出这样的结论，我们实现了一个减小时间消耗和硬件资源占用的新的训练方法，在这个方法下，训练出来的模型能得到较高的准确率、

精确率、召回率、F1 分数、AUC，相较于传统的训练方法，有着较大的提升。

5.2 遇到的困难

在本文实现 FFC（Feature-Centered Pooling）方法的过程中，实际工程实践中遇到了诸多挑战，主要集中在数据处理、训练稳定性以及样本分布等方面。在实际应用中，FFC 方法展现出了一定的优势，尤其是在增强特征的表达能力和提升类间判别性方面具有明显效果。通过引入动态维护的特征队列机制，FFC 能够在训练过程中持续更新全局语义信息，使模型具备对难分类样本的持续关注能力，增强了特征的区分度。此外，FFC 在处理大规模人脸识别任务中表现出良好的可扩展性，与深度神经网络结构结合紧密，能够有效兼容主流的训练范式与优化策略。

FFC 方法在提升特征判别力方面具有潜力，但其性能发挥依赖于精心设计的训练策略与合理的样本管理机制，仍有一定的改进空间。在工程实践中也暴露出若干不足。FFC 对特征队列的高度依赖使得其对样本分布的敏感性较高，若训练过程中正负样本比例失衡，可能会导致类间特征分布塌陷，影响模型泛化能力。其次，队列的维护策略（如 LRU 策略）虽有助于保留新鲜特征，但在实际使用中需精心设计更新频率和样本替换机制，否则容易引入噪声或失去多样性。此外，FFC 在训练初期对参数设置较为敏感，尤其是学习率和负样本挖掘策略需要精细调控，以保障训练稳定性和最终性能。未来工作中，可进一步优化负样本挖掘方式、提升特征队列的结构鲁棒性，进一步增强模型的稳定性与实用性。

5.3 未来发展展望

文章提出的 FFC 方法通过引入动态类中心池（DCP）以替代传统的人脸识别中全连接层（FC），有效缓解了在超大规模数据集下训练困难、硬件资源占用过高的问题。通过结合 LRU 管理机制、ArcFace 损失函数以及困难样本挖掘等策略，FFC 实现了模型训练效率与效果的双重平衡。尽管已经取得了显著成效，但仍有诸多值得深入研究与优化的方向。以下将从多个角度探讨未来 FFC 方法的演进路径与研究空间。

当前使用的 LRU 策略是一种经典但相对简单的缓存淘汰机制，通过时间戳或者访问频率来判断样本的重要性，但是在实际的训练过程中，样本出现的频率与该类别的重要性和特征判别能力总非成正比的，可以探索更加只能的 DCP 管理策略来更加智能的管理 DCP。以下是一些改进点的大致思路，可以设置加权访问机制，将访问次数，特征更新的次数，该类别间的间距等因素作为决策依据，设置评分机制，伴随着训练的进行动态的决定每个样本是保留还是替换。

当前工作聚焦于静态图像下的单模态人脸识别场景，而现实中人脸识别系统面临更复杂的应用环境，例如 RGB 与红外图像融合、视频帧序列识别等，这类

任务要求模型能同时处理不同来源的特征输入。例如照片与素描、不同光照或天气条件下的识别。当前的 FFC 架构假设训练与测试分布一致，即训练的集合和测试的集合都是单模态的人脸，而真实应用中，域间差异可能极大影响模型性能，需要探究 FFC 在多模态人脸上面的表现如何。

可以将 FFC 和目标检测的方法相结合，得到更快的更准确的响应速度，方便应用到实时监测场景下面，例如常见的 SSD[14] 方法和 Faster-RCNN[15]。

虽然 FFC 有效降低了 FC 层的参数量与计算负担，但在面对百万级或亿级人脸样本时，仍不可避免地带来一定内存压力，尤其是队列池的频繁更新可能造成显存碎片化。所以，可以研究新的更新策略，比如，减少不必要的 DCP 替换，选择更加适合 DCP 更新的更新策略等。

参考文献

[1] Wen Y, Zhang K, Li Z, et al. Discriminative Feature Learning with Center Loss for Face Recognition[J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(12): 2895-2907.

[2] Schroff F, Kalenichenko D, Philbin J. Deep Feature Learning with Triplet Loss for Face Verification[J].IEEE Transactions on Neural Networks and Learning Systems, 2019, 30(4): 1341-1352.

[3] Deng, J., Guo, J., Zhan, X., et al. ArcFace: Additive Angular Margin Loss for Deep Face Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021, 44(10): 5962-5979.

[4] Wang F., Liu W., Chawla N., et al. Additive Margin Softmax Loss with Adaptive Angular Margins for Face Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020, 42(12): 3021-3035.

[5] Wang H, Wang Y, Zhou Z, et al. CosFace: Large Margin Cosine Loss for Deep Face Recognition[J]. IEEE Transactions on Image Processing (TIP), 2020, 29: 6357-6367.

[6] Liu W, Wen Y, Yu Z, et al. Large-Margin Softmax Loss for Convolutional Neural Networks[J]. Journal of Machine Learning Research (JMLR), 2017, 18(1): 5079-5086.

[7] Liu W, Wen Y, Yu Z, et al. SphereFace: Deep Hypersphere Embedding for Face Recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2019, 41(1): 154-167.

[8] Li M, Andersen D G. Scaling Distributed Machine Learning with the Parameter Server[C]. Advances in Neural Information Processing Systems 27 (NIPS 2014). 2014: 583-591.

[9] Sergeev A, Balso M D. Horovod: Fast and Easy Distributed Deep Learning in TensorFlow[J]. arXiv preprint arXiv:1802.05799. 2018.

[10] Zhao Y, Zhao J, Li M. Fast Face Classification (F²C): An Efficient Training Approach for Very Large-Scale Face Recognition[C]. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022: 1-12.

[11] Wang X. Dynamic Class Queue for Large-Scale Face Recognition[J]. arXiv preprint arXiv:2105.11113. 2021.

[12] Howard A G, Zhu M, Chen B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications[J].arXiv preprint arXiv:1704.04861, 2017.

[13] He K, Zhang X, Ren S, et al. Identity Mappings in Deep Residual Networks[J].IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2016, 39(12): 2481-2495.

[14] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[C].Computer Vision – ECCV 2016. Cham: Springer, 2016: 21-37.

[15] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J].IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2017, 39(6): 1137-1149.

致　谢

首先感谢党和国家，其次感谢江西理工大学给我的教育机会。本科阶段我学到了很多的知识，除了人工智能专业本身要学习的知识点，我还学习了一些计算机方向的知识，江西理工大学是一个包容开放的学校，这段大学时光我生活的很惬意。

大学里面经历了太多的插曲，但是好在结果是美好了。回顾整个大学时光，我经历了留级，自我否定，考研，保研等一系列事情，总感觉不真实，但是确是实实在在的发生了。

大学阶段我遇到了很多的对我有很大帮助的老师和同学：

首先感谢的是我的论文的指导老师，梁苗苗老师和研究生导师龚雪沅老师，对于我论文里面遇到的问题，他们总是耐心的讲解，最终我成功完成了论文的编写。

其次感谢的是程序设计竞赛工作室的教练管希东老师，他在我的大学里面起到了很好的鼓励作用，之前的算法竞赛成绩不顺心的时候，他会鼓励我继续学习，不要放弃，虽然现在我的算法水平还不算太高，但是足够应付一些面试题目和一些算法普及的任务了，这条路让我找工作多了很多的选择。

最后感谢的是我在大学期间认识的好朋友们，主要都是算法竞赛工作室的朋友，他们给我了很大的帮助。

感谢班主任李建润老师和辅导员老师，他们会耐心处理我的问题，感谢电信 202 班、电信 201 班、人智 212 班，未来再见！