分类号：_____ 单位代码：10300

密　级：_____ 学　　号：202212490737

南京信息工程大学

硕士专业学位论文

论文题目：轻量级人脸表情识别神经网络研究

申请人姓名：孙培源

指导教师：袁甲、孙玉宝

类别名称：电子信息

领域名称：计算机技术

培养学院：计算机学院、网络空间安全学院

提交时间：2025 年 6 月 2 日

二〇二五　年　六　月

目 录

摘　要

人脸表情识别作为情感智能计算的核心研究方向之一，在智能交互系统、驾驶员状态检测以及虚拟现实等领域具有重要的理论研究价值和工程应用意义。传统的表情识别方法依赖人工设计的特征提取算法，泛化能力有限，难以满足实际需求。深度学习技术虽然大幅提升了表情识别的准确率，但深层网络结构带来的高计算复杂度与存储需求，限制了其在移动端、边缘端等资源受限设备上的部署。因此，如何在保证识别精度的同时优化计算效率、降低存储需求，已成为表情识别领域亟待解决的关键问题，这对于推动相关技术的实际落地具有重要的研究意义。本文对轻量级表情识别方法进行了深入研究，并在此基础上开发了一款面部表情识别安卓应用。本文的主要工作包括以下几个方面：

⑴在已训练模型的基础上，本文提出了一种基于门控剪枝的全局滤波器修剪方法。该方法通过构建通道重要性评估指标，识别并剪除冗余的滤波器，从而有效地降低了模型的计算复杂度和内存占用。此外，为了减小模型剪枝带来的精度损失，本文还设计了“Punish-Reward-Judge”三阶段迭代剪枝框架，旨在降低模型复杂度的同时逐步恢复模型精度。实验结果表明，在模型有效压缩的基础上，该方法在 AffectNet 数据集上达到了 63.92%的分类精度。

⑵在模型剪枝的基础上，本文提出了一种基于 Hessian 矩阵感知的混合精度通道量化方法。该方法通过分析 Hessian 矩阵迹的分布，构建了通道敏感度评估机制，从而为不同通道的权重和激活值动态分配比特数。此外，为缓解量化过程中精度与效率的矛盾，本文还引入了基于强化学习的“演员-评论家”学习框架，以实现量化比特分配的自适应调整。最终，经过优化的模型在大幅度减小模型体量的同时，仍具有良好的性能表现。

⑶基于压缩后的模型，设计并开发了一款安卓端人脸表情识别应用。通过集成移动端 Pytorch Android Lite 框架，实现了完整的表情识别处理流程，包括图像预处理、人脸检测、区域裁剪以及表情识别推理。应用支持静态图片分析和摄像头信号实时处理两种模式，该应用所有开发工作均在 Android Studio 环境下完成。所使用的表情识别模型在移动端设备上展现出较好的识别准确率，验证了其在移动端部署的可行性。

关键词：人脸表情识别，模型轻量化，门控剪枝，混合精度量化，模型部署

Abstract

Facial expression recognition，one of the core research directions in affective computing，has become of great theoretical and practical significance in areas such as intelligent interactive systems, driver state detection, and virtual reality. Traditional expression recognition methods rely on manually designed feature extraction algorithms, which have limited generalization capabilities and often fail to meet practical needs. Although deep learning techniques have significantly improved recognition accuracy, the high computational complexity and storage demands of deep network structures restrict their deployment on resource-constrained devices such as those at the mobile or edge level. Therefore, how to optimize computational efficiency and reduce storage requirements while maintaining recognition accuracy has become a key challenge in the field of expression recognition, carrying important research significance for the practical implementation of related technologies. This paper conducts an in-depth study of lightweight facial expression recognition methods and, based on this, implements an Android application for facial expression recognition. The main contributions of this work include the following aspects:

(1) Building on a pre-trained model, we propose a global filter pruning method based on gated pruning. This method constructs a channel importance evaluation metric to identify and remove redundant filters, effectively reducing the model's computational complexity and memory usage. Additionally, to mitigate the accuracy loss induced by pruning, this paper developed an iterative three-stage pruning framework named Punish-Reward-Judge, aimed at gradually restoring model accuracy while reducing its complexity. Experimental results indicate that, with effective model compression, our method achieves a classification accuracy of 63.92% on the AffectNet dataset.

(2) Building upon the pruned model, we introduce a mixed-precision channel quantization method that leverages Hessian matrix awareness. By analyzing the distribution of the Hessian matrix trace, a channel sensitivity evaluation mechanism is constructed to dynamically allocate bits to the weights and activations of different channels. Furthermore, to address the trade-off between accuracy and efficiency during quantization, a reinforcement learning-based "actor-critic" framework is employed for adaptive adjustment of the

quantization bit allocation. The optimized model, therefore, not only significantly reduces in size but also maintains robust performance.

(3) Based on a compressed model, an Android face expression recognition application was designed and developed. By integrating the mobile PyTorch Android Lite framework, a complete expression recognition processing pipeline was implemented, including image preprocessing, face detection, region cropping, and expression recognition inference. The application supports both static image analysis and real-time processing of camera signals, and all development work was carried out in the Android Studio environment. The expression recognition model used has demonstrated relatively high recognition accuracy on mobile devices, validating the feasibility of deploying it on the mobile platform.

Keywords: Facial Expression Recognition, Model Lightweighting, Gated Pruning, Mixed Precision Quantization, Model Deployment

第一章 绪论

1.1 研究背景及意义

计算机视觉（Computer Vision, CV）作为人工智能领域的重要分支，旨在模拟生物视觉机制，结合传感器和算法，实现对目标的智能感知与分析。其在社会生产生活中的广泛应用深刻推动了社会智能化进程。人脸技术作为计算机视觉的核心研究方向之一，主要应用体现在身份识别和情感分析上。其中，人脸识别技术经过多年发展已取得突破性进展，大部分方法识别精度已超过 99%，甚至优于人类肉眼识别能力。作为人脸技术的另一重要分支，人脸表情分析的研究同样具有重要意义。情感交流是人类社会互动的基础，而面部表情作为情感表达的主要载体，在情绪传递中占据主导地位。研究表明，语言、声音和面部表情在情感交流中的权重分别为 7%、38%和 55%[1]，其中面部表情对情绪判断的影响最为显著。因此，通过分析面部表情，可以更准确地理解人类的情绪状态，从而为智能系统提供更自然、更人性化的交互能力。实际上，表情识别技术（Facial Expression Recognition, FER）已在心理健康监测[2]、人机交互优化[4]等领域展现出巨大的应用潜力，成为人工智能技术落地的重要研究方向。

在心理学研究中，Ekman[6]首次将人类表情定义为六种基本类型：高兴、悲伤、生气、惊讶、厌恶和害怕，后续又增加了中性表情，构成了人类表情的基础分类。这一分类体系为表情识别研究提供了明确的目标和方向。在此基础上，Ekman 与 Friesen 提出了面部运动编码系统（Facial Action Coding System, FACS）[7]，将面部表情分解为若干相互独立的运动单元（Action Unit, AU），并建立了表情与肌肉运动之间的映射关系，为表情分析提供了理论依据，在此之后的表情识别任务都是基于 6 种表情定义并在 FACS 上进行的[8]。然而，尽管 FACS 系统在记录面部表情运动方面具有高度精确性，但其在实际应用中仍面临诸多挑战。例如，FACS 系统无法应对复杂的应用场景，尤其是不同年龄段、不同种族人群的面部肌肉特征差异。年轻人面部肌肉饱满，动态表情丰富，而老年人面部肌肉僵硬、皮肤松弛，导致表情特征提取困难。此外，光照条件、遮挡物以及面部姿态的变化也会对表情识别的准确性造成显著影响[9]。这些局限性使得传统方法难以满足现代智能系统对面部表情识别技术的需求，亟需引入更先进的技术手段来解决这些问题。

随着深度学习技术的快速发展，基于卷积神经网络（Convolutional Neural Network，CNN）的表情识别方法因其在自动特征学习上的优势，在识别精度上取得了显著提升。深度学习模型通过端到端的特征学习，能够自动提取表情的高层次特征，避免了传统方法中手工设计特征的局限性。然而，主流深度学习模型通常依赖于复杂的网络结构和庞大的计算资源，这导致其在移动设备或边缘设备上部署时面临着高延时、高能耗和内存占用等问题。考虑到大多数移动设备和边缘设备的计算能力有限且存储空间较小，表情识别网络的资源占用成为实际部署的关键瓶颈[10]。尽管部分设备配备了高性能的 CPU 或 GPU，但表情识别通常只是作为人工智能系统的子模块运行，系统还需同时处理其他任务（如语音识别、环境感知等）。因此，降低表情识别网络的存储需求和计算消耗，对于提升整体系统的运行效率和实时性至关重要。在移动端和边缘端设备部署模型的场景中，资源受限的设备对模型的轻量化提出了更高的要求，存储资源越少、计算消耗越低的模型，越有利于被高效部署和广泛应用。近年来提出的轻量化网络模型（如 MobileNet[11]、EfficientNet[12]、ShuffleNet[13]等）通过深度可分离卷积、通道压缩等技术降低了模型复杂度，在保证一定精度的同时提高了模型的训练速度和部署效率。已有部分研究尝试将轻量化模型应用于表情识别任务，并取得了一定的进展。例如，文献[14]通过使用较少的卷积层构建一个参数较少的 FER 网络，整体参数量比传统神经网络小得多。文献[15]在 MobileNetV2 的基础上使用了多层轻量级卷积和特征融合的方法，提高了模型的运行效率和检测精度。这些方法都在 FER 领域取得了一定的进展，表明基于轻量化模型的研究可以在有效提取表情特征的基础上大幅降低模型的大小和计算需求。

表情识别技术的轻量化研究同样具有显著的社会价值。在智能交通领域，通过实时监测驾驶员的面部表情，可以识别其情绪状态（如悲伤、愤怒等），并及时发出预警[16]，从而降低交通事故的发生率。在教育场景中，通过定时监测，能够实时捕捉并解析学习者的面部表情特征，通过情感计算模型准确识别其认知负荷与情绪波动，为教师动态调整教学方案提供数据支撑[17]，进而实现精准化教学干预。在医疗健康领域，该技术可作为精神心理评估的重要辅助工具，通过量化分析患者的面部表情变化，为临床医生提供客观的情绪状态评估指标，特别是在抑郁症、焦虑症等心理障碍的早期筛查和疗效评估中发挥关键作用[18]。此外，表情识别技术还可应用于智能安防、电子商务、智能家居等多个领域，为社会生活的智能化发展提供支持。例如，在智能安防中，通过分析监控视频中的人脸表情，可以识别潜在的危险行为；在电子商务中，通过分析用户的表情反馈，

可以优化产品推荐和用户体验；在智能家居中，通过识别家庭成员的情绪状态，可以自动调节环境氛围，提升生活舒适度。这些应用场景充分体现了表情识别技术轻量化研究的重要性和广泛前景。

1.2 国内外研究现状

1.2.1 传统人脸表情识别方法

人脸表情识别技术体系可划分为传统机器学习和深度学习两大范式。在深度学习革命性突破之前，传统机器学习方法主导着该领域的研究方向，其技术框架遵循“图像预处理-特征提取-分类决策”的线性流程。其中，特征提取作为核心环节，主要采用人工设计特征与浅层学习相结合的策略，主要特征包括：基于 LBP（Local Binary Patterns）和 HOG（Histogram of Oriented Gradients）等算子的局部纹理特征、利用 ASM（Active Shape Model）模型获取的几何形变特征，以及遵循 FACS 标准的动作单元特征。代表性研究成果如 Li 团队开发的 HOG-BoW 混合模型[19]，该模型通过整合梯度方向直方图与视觉词袋方法，实现了多层次表情特征的融合表达。在分类器选择上，研究者广泛采用 K 近邻算法（K-Nearest Neighbors，KNN）、决策树[20]和支持向量机（Support Vector Machine，SVM）[21]等经典算法，这些方法在一些实验中取得了显著进展[22]。然而，传统方法面临两个根本性挑战：首先，特征工程严重依赖领域专业知识（如表情心理学理论和图像处理经验），导致系统开发效率低下且泛化能力受限；其次，手工特征在复杂现实场景（如光照变化、姿态多样性）下的鲁棒性不足，这从 CK+[25]等实验室数据集与 SFEW[26]等真实场景数据集的性能差异中可见一斑。上述方法在这些方面的局限性推动了表情识别研究向着数据驱动的深度学习方法转变。深度学习方法突破了传统方法的性能瓶颈，提供了新的解决方案。

1.2.2 基于深度学习的人脸表情识别方法

与传统算法依赖人工特征工程的显式建模不同，深度学习通过端到端的学习机制实现了特征表示与任务目标的协同优化，其核心优势在于建立了从数据驱动特征提取到自适应参数更新的闭环系统。这种基于反向传播的自动微分框架，使得模型能够通过多层次非线性变换来挖掘表情的潜在语义特征。对于面部表情的局部细微变化和全局结构特征，卷积神经网络可以通过局部感受野和权值共享机制进行有效捕获，避免了传统方法

中手工设计特征的局限性和主观性。深度学习领域方法在 FER2013[27]等复杂数据集上实现了超 85%的识别精度突破，较传统方法提升巨大。

随着卷积神经网络的快速发展，人脸表情的识别技术迎来了显著的进步。2019 年，Yong 等人[28]提出了一种针对面部遮挡问题的创新解决方案，通过动态权重分配机制，降低遮挡区域的影响，同时增强未遮挡区域的贡献，显著提升了识别精度。同年，Ronak Kosti 等人[29]将表情特征与外部环境信息结合，通过多模态融合技术进一步提高了识别性能。Siqueira 等人[30]创新性地将神经网络与集成学习相结合，设计了一种高效的特征融合框架，该框架在显著降低计算复杂度的同时，确保了模型的特征多样性和泛化性能。针对训练数据中的标注不确定性的问题，Jiahui She 等人[31]提出了基于分布学习的 DMUE 模型，通过潜在特征空间分析和不确定性量化机制，有效缓解了人工标注的主观偏差对模型性能的影响。2022 年，Wen 等人[32]提出的 DAN 网络通过多头注意力机制和不对称卷积，实现了对多个非重叠区域的并行关注，从而区分了细微的表情差异。2023 年，多项研究进一步推动了该领域的发展。Zhou 等人通过融合特征层和 3D 空间预处理技术[33]，增强了模型对遮挡问题的鲁棒性；Zhang[34]在 ResNeXt50 中引入多尺度特征融合层和 Soft Pooling 池化，提升了特征提取能力；Feng 的研究团队[35]设计了一种新型的 AA-Net 架构。该模型创新性地整合了抗混叠残差模块与注意力机制这两种方法，通过多层次特征增强策略和损失函数优化，显著提升了网络的表征学习的能力。2024 年，面部表情识别方面的研究继续取得突破。在第六届情感行为分析比赛（Affective Behavior Analysis in-the-wild ，ABAW）中，Yu 等人[36]通过结合半监督学习、去偏反馈学习和时序编码器，有效解决了数据不平衡和特征偏差问题。Tao 和 Duan[37]提出的层次化注意力网络和渐进特征融合方法，通过聚合多样化特征并增强面部关键区域的区分性特征，进一步提升了精度；Gong 等人[38]提出的增强时空学习网络（Enhanced Spatial-Temporal Learning Network，ESTLNet）通过空间融合学习和时间变换器增强模块，优化了空间和时间特征的表示能力，显著提高了动态面部表情识别的性能。

1.2.3 轻量级人脸表情识别方法

近年来，轻量级神经网络在人脸表情识别领域的研究进展呈现出两个明显的技术路径：其一是基于轻量级网络架构的创新性改进，其二是基于模型压缩的量化技术创新。这两个方向的协同发展有效平衡了模型性能与计算效率的矛盾，推动了表情识别技术在边缘端设备的应用进程。

在架构创新方面，研究者们通过改进现有轻量级网络或设计全新的小型网络，显著降低了模型的计算复杂度和参数量，使其更适合部署在资源受限的设备上。具体改进策略包括引入注意力机制、调整卷积层通道数量、结合标签学习等，旨在提升模型性能的同时减少资源消耗。研究[39]提出了一种基于 MobileNetV1 的改进方案，通过嵌入注意力机制来强化局部表情特征的表示能力。该方案采用双损失函数策略，将中心损失与 Softmax 损失相结合，有效提升了特征的类内紧凑性和类间区分度。实验结果表明，该方法在不增加模型参数的情况下实现了精度提升，但其分类性能仍存在优化潜力。另一项研究[40]则聚焦于 ShuffleNetV2 的结构优化，通过改进特征提取模块，在降低计算复杂度的同时维持了模型的识别性能，为移动端表情识别提供了可行的解决方案。类似地，文献[41]通过引入分组卷积和随机通道重排技术，优化了 ShuffleNetV2 的设计，使其在精度上表现更优。此外，文献[42]提出了一种基于多特征融合的轻量级卷积神经网络（Multi-Feature Fusion Convolutional Neural Network, MFF-CNN），该模型采用图像分支和局部块分支并行提取全局与局部特征，并通过特征选择机制筛选最具辨识性的局部特征，从而减少全连接层的计算负担。然而，由于 MFF-CNN 的卷积层数较少，其在处理复杂表情识别任务时性能有所下降。

在模型压缩层面，研究者们通过多种网络量化技术来减少表情识别模型的参数量、计算量和存储空间，同时尽可能降低精度损失。常见的量化方法包括低秩分解、模型剪枝、知识蒸馏和比特量化等。这些方法通常基于现有的成熟神经网络框架进行优化。本文的研究也属于这一领域，旨在通过量化技术降低模型的计算复杂度和参数量，同时保持其性能。例如，文献[43]提出了一种新的软标签生成方法和知识蒸馏策略，将标签置信度估计网络中的丰富知识迁移到表情分类网络中，不仅缩小了模型规模，还显著提升了其表征能力和泛化性能。类似地，文献[44]利用伪孪生网络进行知识蒸馏，进一步提高了表情识别模型的准确率。在模型剪枝方面，文献[45]通过对 GoogLeNet 进行剪枝和再训练，移除低权重连接，并引入全局最大池化层以保留目标位置信息，同时采用 Sigmoid 交叉熵作为训练目标，从而提取更全面的人脸表情特征。改进后的网络表现出更高的实用性。此外，文献[46]基于粗糙集和集成剪枝方法对人脸表情识别模型进行裁剪，通过剔除弱分类器或冗余分类器，筛选出最优分类器子集，并采用多数投票法进行集成，最终实现了较高的识别准确率。这些研究表明，网络量化技术在表情识别领域具有广泛的应用前景。

1.3 研究内容

本文围绕轻量级人脸表情识别神经网络研究展开，通过模型优化、量化压缩等维度的技术创新，有效地解决了网络模型性能和网络模型资源需求过高的矛盾。具体研究内容如下：

（1）基于门控剪枝的轻量级表情识别方法

首先，选取 ResNet[47]、MobileNet 和 EfficientNet 这些主流轻量级神经网络进行人脸识别预训练，通过性能评估筛选出最优基础模型，并将其迁移至表情识别任务，以此简化训练流程并提升模型初始精度。其次，设计了一种基于门控机制的全局滤波器剪枝算法，通过构建通道重要性评价指标，实现对冗余滤波器的识别与修剪，显著降低了模型的计算复杂度与内存占用。为进一步优化剪枝过程中的精度损失问题，创新性地提出了 Punish-Reward-Judge 三阶段迭代剪枝框架，用于在剪枝过程中逐步恢复模型精度。最后，需要对模型进行微调以进一步提高剪枝后模型的精度。

（2）基于混合精度量化的轻量级表情识别方法

在模型剪枝的基础上，进一步提出了一种基于 Hessian 矩阵感知的混合精度通道量化策略，以实现模型的深度压缩与加速。该方法通过分析 Hessian 矩阵的 Trace 分布，构建了通道敏感度评估机制，从而为不同通道的权重和激活值动态分配最优量化比特数。具体而言，对于量化敏感度较高的通道，采用较高精度进行量化；而对于低敏感度通道，则使用低精度进行量化，在保证模型精度的同时显著提升了存储效率和计算性能。为进一步解决量化过程中的精度-效率权衡问题，本文引入了基于强化学习的"演员-评论家"（Actor-Critic）框架，通过构建量化策略评估函数，实现了量化比特分配的自适应优化。

（3）人脸表情识别应用开发

设计并开发了一款基于深度学习模型的安卓端人脸表情识别应用。为确保模型在移动设备上的有效运行，采用了剪枝和量化技术优化过的模型，并将其转换为 PTL 格式，以实现轻量化部署。通过集成 Pytorch Android Lite 框架，实现了完整的人脸处理流程，包括图像预处理、人脸检测、区域裁剪以及表情识别推理。应用支持静态图片分析和摄像头信号实时处理两种模式，该应用所有开发工作均在 Android Studio 环境下完成。所使用的表情识别模型在移动端设备上展现出较好的识别准确率，验证了其在移动端部署的可行性。

1.4 论文组织结构

本文内容安排如下：

第一章，绪论。本章首先概述了轻量级表情识别的研究背景与重要性，探讨了该任务在实际应用中的广泛应用场景，阐述了本文工作的研究价值。接着，回顾了传统方法与深度学习方法在表情识别领域的国内外研究现状，对所使用的技术进行了对比，重点阐述了轻量级神经网络在该领域的最新研究动态与创新突破。最后，概述了本文的研究内容和组织结构。

第二章，相关理论基础。本章首先介绍了轻量级神经网络的基本概念，并重点分析了当前主流的轻量级神经网络模型架构及其特点。随后，章节深入探讨了实现模型轻量化的关键技术，包括结构剪枝、模型量化等方法，分析了每种方法的优势和局限性。最后，章节简要介绍了本文使用的数据集。

第三章，介绍了本文提出的基于门控剪枝的轻量级表情识别方法。首先，分析了计算资源与模型性能之间在表情识别任务上的矛盾，并基于此问题引出了本章所采用的模型结构化剪枝方法。接着，详细描述了表情识别模型构建的具体过程，涵盖了从模型设计到剪枝策略的应用。最后，通过实验验证了所提方法的有效性。

第四章，介绍了本文提出的基于混合精度量化的轻量级表情识别方法。首先，从硬件支持的角度分析了网络量化的必要性，并基于此问题引出了本章所采用的混合精度通道量化方法。接着，详细阐述了该方法的具体技术细节，涉及量化策略和优化步骤。最后，通过一系列实验与分析，验证了所提方法的有效性。

第五章，介绍了基于压缩后的模型进行安卓应用开发的过程。详细说明了使用 Pyto-rch Android Lite 框架将论文中提出的模型成功部署至安卓平台的步骤。实现了基于静态图片和摄像头信号的人脸表情识别功能，验证了所提方法在实际应用中的有效性。最后展示了应用程序的识别效果。

第六章，总结与展望。本章对全文进行了总结，回顾了主要研究内容，并对未来的研究提出了展望。

第二章 相关理论基础

本章首先对轻量级神经网络的基本概念和相关模型进行了系统梳理，重点介绍了当前主流的轻量级网络模型及其特点，包括这些模型的结构设计、优化策略以及它们在不同领域中的应用。通过对这些模型的全面分析，可以深入了解轻量级模型的理论基础、优势与局限性。接着，本章进一步探讨了与模型轻量化相关的关键技术，包括模型剪枝、量化等方法的基本原理，详细分析了这些技术在提升模型计算效率和降低内存占用的同时，如何最大限度地保持模型精度。最后，本章还将详细介绍实验中所采用的数据集，涵盖数据集的来源、数据集的结构及其代表性，以确保实验结果的科学性和可信度，同时为后续的实验分析提供充分的背景支持。

2.1 轻量级神经网络概述

轻量级神经网络是一类专注于在性能与效率之间实现最佳平衡的深度学习模型。通过对网络模型结构的精心设计、模型参数的高效压缩以及模型计算复杂度的有效降低，这类网络模型在保证较高模型性能的同时，大幅减少了对计算资源的依赖。这使得轻量级网络能够在计算能力有限的设备上（如智能手机、嵌入式系统、物联网设备等）稳定高效地运行，从而推动了人工智能技术在各个领域的普及与应用。与传统深度学习模型（如 VGGNet[48]、InceptionNet[49]等）相比，轻量级网络在保持较低计算资源和内存占用的同时，仍能实现高精度的分类任务，并且能够满足实时性和低功耗等实际应用需求。为了实现这一目标，轻量级神经网络的技术创新主要集中在网络架构的设计优化、模型压缩技术（如剪枝[50]、量化[53]、蒸馏[56]）等方面。在实际应用中，轻量级神经网络已经展现出其极高的实用价值，尤其是在移动端和边缘端场景下。以实时表情分析、物体识别和语音识别等分类任务为例，这些任务通常要求设备在短时间内对输入数据进行处理，并输出高精度的识别结果。然而，由于这些设备的计算资源和存储容量受限，传统深度学习模型往往难以部署。轻量级模型通过精细的网络结构设计和一系列模型压缩技术，有效降低了运算复杂度和内存占用，从而突破计算瓶颈，实现了快速且高效的数据处理。通过不断的技术创新和优化，轻量级神经网络已经成为推动人工智能技术落地的重要支撑。它们不仅提高了边缘设备的计算能力，显著缩短了从数据采集到结果输出的延迟，还为智能硬件的普及奠定了坚实的技术基础。

2.2 轻量级网络模型

2.2.1 MobileNet 模型

MobileNet 是由 Google 研究团队于 2017 年开发的一种高效卷积神经网络架构，专门面向移动终端及嵌入式设备等资源受限环境下的视觉计算任务。该网络创新性地采用了深度可分离卷积结构（Depthwise Separable Convolution），将标准卷积分解为逐通道卷积和逐点卷积两个独立操作，这一设计显著降低了模型的计算复杂度和存储需求，从而在保证精度的同时显著提升了效率。MobileNet 在图像分类、目标检测等任务中表现优异，尤其适用于在移动设备和嵌入式系统上部署。MobileNet 的核心思想是通过深度可分离卷积来优化卷积操作，从而减少模型的计算复杂度。这也是该网络模型的基本单元。与传统的卷积操作不同，深度可分离卷积将卷积过程分解成两个阶段：首先，进行深度卷积（Depthwise Convolution），即每个输入通道使用一个独立的卷积核进行卷积操作，生成与输入通道数相同的特征图；然后，使用逐点卷积（Pointwise Convolution）进行跨通道信息融合。下图 2-1 展示了深度可分离卷积的卷积过程。

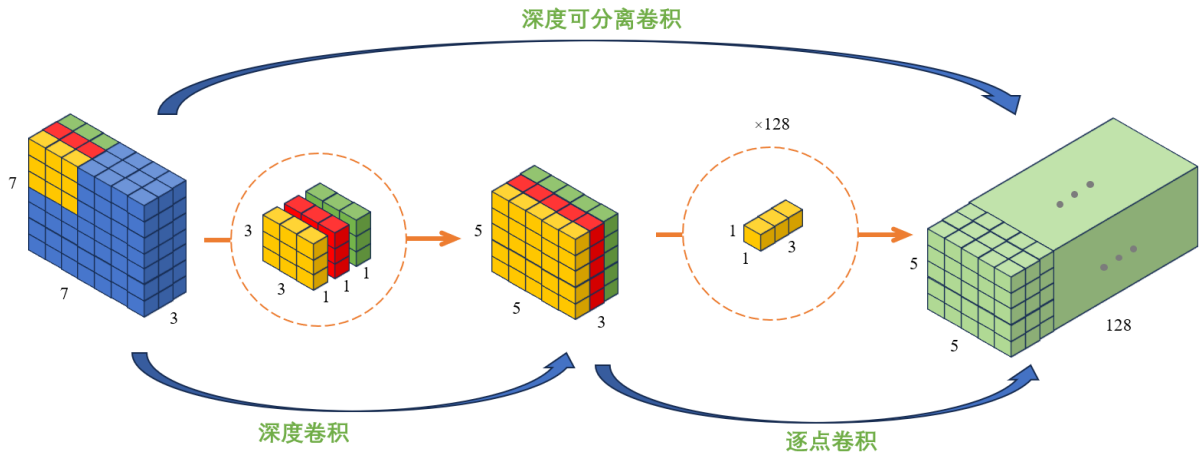


图 2-1 深度可分离卷积过程示意图

在标准卷积操作中，每个输出通道的卷积核需要与所有输入通道的对应区域进行卷积计算。假设输入特征图的尺寸为 $C_{in} \times H \times W$，输出特征图的尺寸为 $C_{out} \times H \times W$，卷积核尺寸为 $K \times K$。其中 C_{out} 是输出通道数，C_{in} 是输入通道数，H 和 W 分别表示特征图的高度和宽度，则标准卷积的计算量 F_{std} 和参数量 P_{std} 可以分别表示为：

$$F_{std} = C_{in} \times C_{out} \times H \times W \times K^2 \tag{2-1}$$

$$P_{std} = C_{in} \times C_{out} \times K^2 \tag{2-2}$$

标准卷积的计算复杂度与输入通道数、输出通道数和卷积核大小的乘积成正比，因此其计算资源消耗较大，尤其在深层网络中，这一问题尤为突出。而深度可分离卷积通过将标准网络卷积操作分解为逐通道的深度卷积和1×1的逐点卷积，从而显著减少了计算量和参数量，同时仍能实现相同的卷积效果。

深度卷积通过解耦特征提取过程中的通道相关性与空间相关性，有效降低了模型的计算负担。该方法为每个输入通道分配独立的卷积核，仅在单个通道的空间维度上进行滑动计算，从而实现了特征提取过程的轻量化。这种设计不仅减少了参数数量，还保持了特征的空间结构信息。此时深度卷积的计算量和参数量分别为 $C_{in}\times H\times W\times K^2$ 和 $C_{in}\times K^2$，输出特征图的通道数和输入特征图相同，但是无法实现跨通道信息的融合。为了解决深度卷积无法融合跨通道信息的问题，深度可分离卷积引入了逐点卷积。逐点卷积采用1×1卷积核，对深度卷积输出的每个像素点进行跨通道加权融合。其计算量和参数量可以表示为 $C_{in}\times C_{out}\times 1^2\times H\times W$ 和 $C_{in}\times C_{out}$。逐点卷积通过线性组合不同的通道特征，建立通道间的依赖关系，同时避免了传统卷积中大尺寸卷积核的额外计算开销。深度可分离卷积的总计算量和总参数量分别表示为：

$$F_{dp}=C_{in}\times H\times W\times K^2+C_{in}\times C_{out}\times 1^2\times H\times W \tag{2-3}$$

$$P_{dp}=C_{in}\times C_{out}+C_{in}\times K^2 \tag{2-4}$$

与标准卷积相比，深度可分离卷积的计算量和参数量减少比例均为：

$$\frac{1}{C_{out}}+\frac{1}{K^2} \tag{2-5}$$

作为 Google 提出的轻量级移动端分类网络，MobileNet 经过三个版本的迭代，逐步优化了模型结构，在精度与效率之间实现了更好的平衡。MobileNetV1 的核心创新在于引入了深度可分离卷积，其设计基于跨通道相关性与空间相关性可解耦的假设。该卷积将标准卷积分解为深度卷积和逐点卷积两个独立步骤，分别提取空间特征和融合跨通道信息。此外，MobileNetV1 提出了宽度因子和分辨率因子两个超参数，用于动态调整网络的通道数和输入分辨率，从而控制模型容量。MobileNetV2 在 V1 的基础上进一步优化，提出了线性瓶颈倒残差结构。这一模型的技术改进主要有两个方面，一方面是在线性瓶颈层使用线性激活函数，避免低维特征信息丢失。另一方面，MobileNetV2 模型采用倒残差结构，通过先扩展再压缩通道数的方式提升特征提取能力。在上述改进的基础上，MobileNetV2 还将跳跃连接应用于低维瓶颈层，进一步增强了梯度流动和特征复用能力。MobileNetV3 结合神经网络架构搜索和手动设计优化，进一步提升了模型的性能

和效率。其主要的创新点包括利用神经网络框架搜索技术（Neural Architecture Search，NAS）自动搜索最优网络结构，采用 h-swish 激活函数增强非线性表达能力，以及引入 SE 模块动态调整特征权重。

2.2.2 EfficientNet 模型

EfficientNet 同样是由 Google 提出的深度学习网络架构，旨在提高深度神经网络的计算效率，并在保证高精度的情况下显著减少计算开销。其核心创新在于引入了复合缩放策略（Compound Scaling Method），该方法通过协同优化网络的三个关键维度：深度（d）、宽度（w）以及输入图像的分辨率（r），以实现模型性能的显著提升。传统的网络扩展通常只对其中一个维度进行缩放，例如增加网络层数或通道数，但这种单一维度的缩放往往导致性能提升不均衡，计算和存储开销也会显著增加。EfficientNet 通过统一的缩放系数对深度、宽度和分辨率进行均衡扩展，生成了从 B1 到 B7 的一系列模型。复合缩放的比例关系可通过以下公式描述：

$$d' = \alpha^{\phi}, \quad w' = \beta^{\phi}, \quad r' = \gamma^{\phi} \tag{2-6}$$

其中，d'、w'、r' 分别为新网络的深度、宽度和输入分辨率，α、β、γ 为缩放因子，ϕ 为全局缩放系数。这种协调性的扩展策略使得 EfficientNet 在计算资源有限的情况下，能够显著提升模型的精度和效率。

复合缩放的核心思想是，深度、宽度和分辨率应按某种比例一起增加，而不是单独调整某一个维度。这种方法的优势在于，它能在增加计算资源时保持合理的性能提升，而不会造成过多的计算冗余。EfficientNet 通过联合缩放深度、宽度和分辨率，避免了传统单一维度扩展导致的无效计算增加。例如，仅增加网络深度会导致梯度消失问题，而仅增加宽度或分辨率则会显著增加计算量。复合缩放通过以下公式确保各维度的协调扩展：

$$F_{efficient} \propto d \times w^2 \times r^2 \tag{2-7}$$

其中 $F_{efficient}$ 表示模型的计算量，公式表明，深度、宽度和分辨率的增加都会导致计算量的增长，但宽度和分辨率的影响更为显著。EfficientNet 之所以能够实现轻量化，一方面是通过联合缩放深度、宽度和分辨率，避免了无效的计算增加，保证了计算资源的高效使用。另一方面，该设计通过减少非线性映射降低了计算量，同时保留了特征表达能力。

2.3 模型轻量化技术

2.3.1 模型剪枝

对于一个深度神经网络 N 和验证数据集 D，剪枝问题可以被转化为一个优化问题，目的是根据以下公式进行优化：

$$\max \quad Acc(N_{pruned}, D) \tag{2-8}$$

其中满足以下约束条件：

$$s.t. \quad N_{pruned} \subseteq N, \mathrm{Cost}(N_{pruned}) \le C \tag{2-9}$$

剪枝任务的主要目标是找到一个剪枝后的网络结构 N_{pruned}，使其在验证集 D 上获得尽可能高的精度。剪枝后的网络是原始网络 N 的一个子集，是原始网络的精简版本。在剪枝算法的实施过程中，需要严格控制其计算复杂度，使其始终保持在预定的约束范围内。这一约束条件的具体定义可根据实际应用场景的需求灵活调整，既可以是传统的计算量（FLOPs）或参数量（Parameters），也可以进一步扩展为硬件相关的性能指标，例如能耗或推理时延等。通过这种方式，剪枝技术能够在保证模型性能的同时，显著降低资源消耗，从而满足实际应用中的高效部署需求。

基于剪枝操作在网络结构中的实施粒度差异，剪枝技术可系统性地划分为四种主要类型：层间剪枝、特征图剪枝、卷积核剪枝以及卷积核内剪枝。

层间剪枝作为最粗粒度的剪枝方式，其主要通过直接移除网络中的完整卷积层来实现模型压缩，如图 2-2(a)所示，其中阴影部分表示被裁剪的网络层。特征图剪枝则针对卷积层中的三维张量进行优化操作，如图 2-2(b)所示，其中阴影区域表示被裁剪的特征图。值得注意的是，由于特征图与卷积核在计算过程中存在紧密的相互依赖关系，被裁剪的特征图会同步影响对应卷积核的维度，而裁剪卷积核则会进一步导致下一层输出特征图数量的减少。第三类剪枝方式为卷积核剪枝，其核心在于对滤波器中的二维卷积核进行精细化修剪，如图 2-2(c)所示，该操作本质上是对三维滤波器的二维通道实施裁剪。最后，卷积核内剪枝作为最细粒度的剪枝方法，其主要通过对卷积核内的单个权重进行稀疏化处理，将部分冗余权重置为零，如图 2-2(d)中阴影部分所示。这种多层次、多粒度的剪枝策略不仅为神经网络压缩提供了灵活的技术路径，同时也体现出对不同实际应用的适应性。

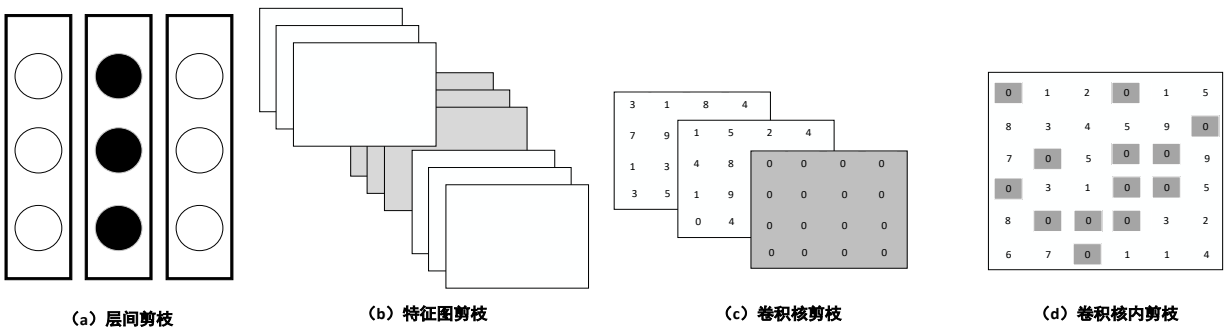


图 2-2 不同粒度剪枝示意图

根据剪枝策略的实施方式不同，剪枝技术主要可分为非结构化剪枝与结构化剪枝两大类。非结构化剪枝的核心在于通过权重稀疏化实现模型压缩，其具体方法是将网络中冗余或不重要的连接权重置为零。这类技术的剪枝对象为神经元级别的权重参数，尽管能够实现较高的稀疏化比例，但由于其生成的稀疏模式具有非规则性，在实际应用中需要依赖特定的稀疏计算框架或定制化的硬件加速器才能充分发挥性能优势，这在一定程度上制约了其工程实用性。

与之相对，结构化剪枝以更大尺度的结构单元作为剪枝目标，例如移除完整的卷积核或特征图。这种剪枝策略不仅显著降低了网络的计算复杂度，而且能够保持剪枝后模型结构的规整性，从而无需额外的软硬件适配即可在通用计算平台上高效执行。具体来说，结构化剪枝包括了层间剪枝、特征图剪枝和卷积核剪枝，这些方法通过删除完整的网络组件来减少计算开销，从而简化了硬件部署的复杂性。而卷积核内剪枝属于非结构化剪枝范畴，其通过删减单个卷积核中的权重来实现压缩。由于结构化剪枝在硬件部署方面具有更好的适配性，其应用场景和推广范围远远超出非结构化剪枝。

剪枝技术的核心目标是识别并移除神经网络中冗余或贡献较小的连接权重，实现模型结构的优化与压缩。这种优化不仅能够显著降低模型的参数量，还能有效减少计算复杂度，从而提升模型的运行效率。如图 2-3 所示，典型的剪枝流程通常包含以下三个步骤：

图 2-3 剪枝流程示意图

（1）模型剪枝初始阶段的目的是构建一个高性能的基准模型，即通过充分的预训练来获得在目标任务上满足精度要求的初始网络。预训练模型为后续剪枝提供了可靠起点和性能保障，是剪枝任务的前提。

（2）在获得预训练模型后，依据预先设定的剪枝规则，对网络结构进行优化。通过移除冗余的层、通道或权重参数，生成一个更加紧凑的网络架构。这一步骤的核心目标是在降低模型复杂度的同时，尽可能保留其表征能力，从而实现模型效率与性能的平衡。

（3）剪枝操作完成后，由于模型结构或参数发生了变化，通常会导致性能的下降。因此，需要在目标数据集上对剪枝后的模型进行微调，通过多轮训练来恢复模型的精度，从而弥补剪枝带来的性能损失。

模型的剪枝过程主要分为迭代式剪枝（Iterative Pruning）和一次性剪枝（One-shot Pruning）两种模式。迭代式剪枝采用剪枝与训练交替进行的策略，每次剪枝后通过短期训练来补偿精度损失。尽管这种方法需要更多的计算时间，但由于其渐进式的特性，能够在保证模型性能的同时实现更高的压缩率。因此，迭代式剪枝已成为工业界广泛采用的主流方法。

一次性剪枝则是按照预设的剪枝比例一次性对整个网络进行剪枝。虽然这种方法操作简便，但早期的研究发现，过度剪枝之后，即便通过微调训练，网络的精度也难以恢复。这种方法的缺点是，一旦剪枝比例过大，模型的精度会显著降低且难以恢复。然而，一次性剪枝能够显著提高剪枝效率，因此在某些场景下具有优势。近年来，学术界对一次性剪枝方法进行了深入研究，也取得了一些新的成果。

2.3.2 模型量化

神经网络模型量化是一种通过将模型中的浮点参数转换为定点参数从而实现减少模型大小、加快运算速度的压缩方法。这一过程同样可以看作一个优化问题，其核心目标是最大程度压缩模型的同时，确保量化后的模型性能不受显著影响。

$$W_q^* = \arg \ \min_{W_q} L_{quant}(W, W_q) + \lambda \bullet L_{model}(f_{W_q}(x), y) \tag{2-10}$$

其中，W_q^* 是最优量化权重参数，是通过优化得到的最优量化值。W_q 是量化后的权重参数，$L_{quant}(W, W_q)$ 是模型的量化损失，度量了原始权重 W 和量化后的权重 W_q 之间的差异，$L_{model}(f_{W_q}(x), y)$ 表示模型损失，度量了量化后的权重对模型性能的影响，确保量化不会过度降低模型的预测精度。 $f_{W_q}(x)$ 是量化后权重下的模型预测。正则化参数 λ 用来平衡这两部分的权重，对量化精度和模型性能进行折中。

量化技术通过启用硬件中较低精度的计算单元，有效缓解了内存带宽的限制。这一技术通过将模型中的浮点权重替换为定点数值，不仅保持了网络架构的完整性，还实现

了模型体积的缩减。在权重参数和激活值均被量化为低比特表示的情况下，通过采用精简的算术运算单元以及高度优化的计算流程，能够显著提升权重与激活值之间矩阵乘法运算的效率。这种基于低精度量化的计算策略不仅能够大幅降低计算资源的占用率，还可以有效减少推理阶段的时间延迟，从而为资源受限的边缘或者移动平台提供高效的模型部署解决方案。量化程度直接影响着执行速度和能耗，而这一切都是在确保模型精度尽量不受影响的前提下进行的。

不同的精度的计算操作所需要的计算能量和硬件面积成本都不相同，在实际芯片处理中，8 比特定点加法所需要的能量仅是 32 比特浮点加法所需能量的三十分之一[59]，计算面积成本也会大大减少。在低比特位宽量化领域，传统方法会通常采用统一比特宽度策略，即整个网络的每一层均使用相同的比特宽度进行量化。这样的量化方法最为“粗暴”，精度损失也比较大。随着研究的深入，混合精度量化方法逐渐受到关注。该方法允许网络中的各个层级依据其自身特性，采用不同的比特宽度进行量化。通过分析网络中各层对量化误差的敏感度差异，能够为每一层分配合适的量化精度，从而实现更为精细的量化策略。

混合精度量化方法根据神经网络中各层对量化位宽的敏感度差异，动态地为不同层分配不同的比特数，以此优化性能。与传统的低比特位宽量化方法相比，混合精度量化在保持相同压缩比的情况下，能够显著提高模型的推理性能。该方法通过灵活调整各层的量化精度，减少了量化带来的精度损失，并且能够在不显著降低模型精度的前提下，实现较高的压缩率，从而提升推理效率。下图 2-4 表示混合精度量化的操作过程。

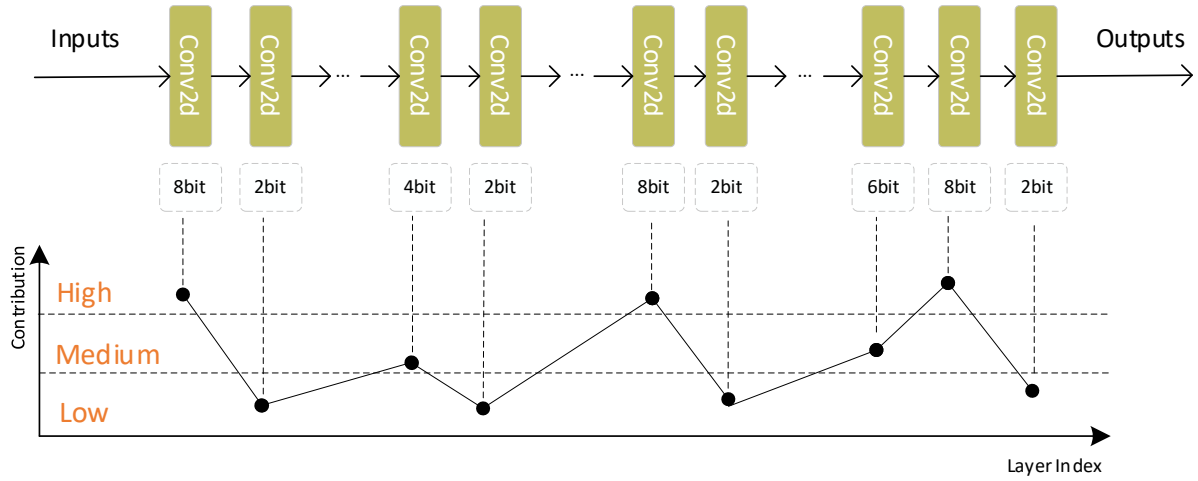


图 2-4 混合精度量化操作示意图

神经网络量化通过将权重和激活值的比特宽度降低从而优化内存利用率并加速计算过程。在实现这一目标时，混合精度量化面临着两个主要挑战：（1）如何在较低精度

的条件下量化网络的权重和激活值，同时尽量减少由此引起的精度损失；（2）如何根据网络各部分的特点，为其权重和激活值分配适当的比特宽度，以达到最佳的性能表现。解决这两个问题的关键在于设计一种合理的混合精度量化策略，能够为每一层选择最优的比特宽度，从而提升整体量化网络的性能。如图 2-5 所示，混合精度量化流程的第一步是从预训练的全精度模型中确定每一层的最优比特宽度，接下来依据这些确定的宽度进行混合精度量化，最后通过微调优化模型以得到最终的量化神经网络。该方法在确保较高压缩比的同时，显著降低了模型的精度损失，从而在资源受限的设备上实现更为高效的推理性能。

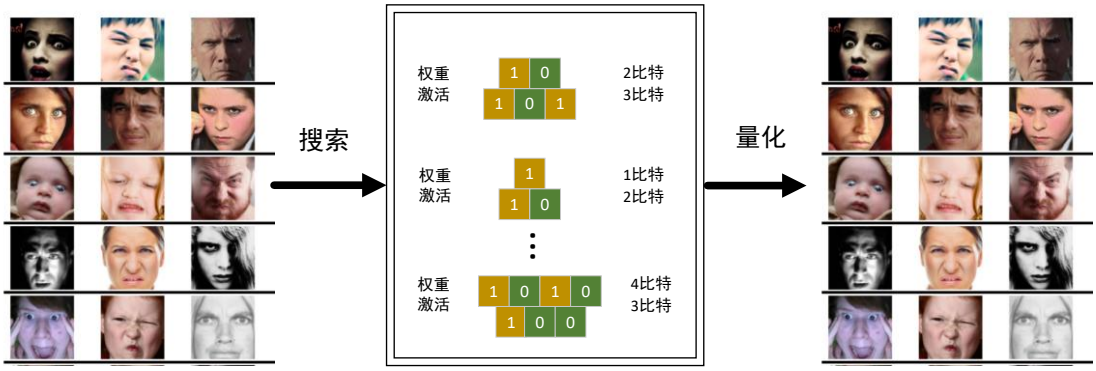


图 2-5 混合精度量化流程示意图

2.4 相关数据集

为了验证所提方法的有效性，本研究使用了三个数据集进行实验评估。表情识别部分采用了来自互联网的大规模 AffectNet 数据集以及实验室的小规模 RAF-DB 数据集。此外，为进一步验证方法的通用性，还额外使用了 ImageNet 这个经典的大规模视觉识别数据集进行实验。

AffectNet 数据集[60]：该数据集包含了超过 100 万张从互联网上收集的面部图像，提供了离散的情感分类注释和连续的情感维度测试。与实验室数据集相比，虽然 AffectNet 数据集提供了更丰富的表情样本，但是在研究应用中面临着更大的挑战，如标注一致性、模型训练时间管理以及各类别样本数量不均衡等问题。如表 2-1 所示，“开心”类别的样本数量最多，这符合预期，而“中性”类别的样本数排名第二。相对而言，“轻蔑”和“厌恶”类别的数据量在数据集中是最少的，不同类别的样本分布差异显著。为此，本文从 AffectNet 数据集中选取了 8 个表情类别（中性、快乐、悲伤、惊讶、恐惧、厌恶、愤怒和轻蔑）进行训练和验证，并在去除“轻蔑”类别后，基于剩余的七个表情类别再次进行训练和验证。

表 2-1 AffectNet 数据集各分类样本数量

表情	数量/张
愤怒	25382
轻蔑	4250
厌恶	4303
恐惧	6878
快乐	134915
中性	75374
悲伤	25959
惊讶	14590

RAF-DB 数据集[61]：该数据集包含了 29672 张来自真实世界的面部图像，这些图像由 315 名注释员根据不同的情感分类进行标记。每张图像都标注了 40 个独立的注释。RAF-DB 数据集被分为两个子集：单标签子集和多标签子集。在本研究中，所有实验均使用单标签子集。

ImageNet 数据集[62]：ImageNet 是一个大规模视觉数据库，广泛用于图像分类和物体识别任务。该数据集包含了超过 1400 万张图像，覆盖了 20000 多个类别，其中每个类别代表了一种具体的物体或概念。ImageNet 的数据被标注为多种级别，包括图像的标签和物体的位置等信息。作为计算机视觉领域的重要基准数据集，ImageNet 广泛应用于深度学习模型的训练和评估，直到目前，该数据集仍然是深度学习领域中图像分类、检测、定位的最常用数据集之一。

2.5 本章小结

本章首先梳理了轻量级神经网络的基本概念及相关模型，重点介绍了当前主流的轻量级模型及其特点，为读者全面呈现了轻量级网络的理论基础及其广泛的应用领域。随后，深入探讨了与模型轻量化相关的关键技术，包括模型剪枝和量化等方法的原理及其在模型压缩中的作用，分析了模型压缩对计算效率、内存占用和模型精度的影响。最后，介绍了本课题实验所采用的数据集，以确保实验结果的科学性和可信度。通过对轻量级网络模型、轻量化技术以及数据集的全面介绍，本章为后续的实验研究奠定了坚实的理论基础。

第三章 基于门控剪枝的轻量级表情识别方法

3.1 引言

尽管许多表情识别神经网络已经取得了显著成果，但是在实际应用中，包括当前一些轻量化神经网络在内的许多模型仍然面临计算量过大和存储需求过高的问题，一些资源受限的环境（如边缘设备和嵌入式设备）极大地限制了它们的应用。尤其在高效部署的需求下，如何在不牺牲性能的前提下减少模型复杂度并提高计算效率，成为了亟待解决的核心挑战。此外，随着数据集规模的不断扩展，如何确保模型在不同应用场景中的稳定性和高效性，依旧是研究的关键课题。

本章提出了一种基于门控剪枝的轻量级模型构建方法，旨在通过迁移学习和模型剪枝压缩技术优化模型的训练流程和性能表现。首先，该方法通过在轻量化模型上进行迁移学习，简化了表情识别任务的训练过程。随后，对迁移学习后的模型进行压缩，采用细粒度的通道剪枝技术进行小幅度迭代压缩。基于此，本章设计了一种惩罚-奖励-决断（Punish-Reward-Judge）三阶段的迭代门控剪枝算法，用于进一步压缩迁移学习后的基础模型。在 Punish 阶段，卷积层权重被冻结，并在子数据集上进行更新训练，目的是加速修剪过程，并通过门控剪枝算法计算滤波器的重要性得分以进行裁剪。在 Reward 阶段，卷积层权重被解冻，并在全数据集上进行更新训练，以减少因滤波器删除引起的误差累积，同时增强对缩放因子的稀疏性约束。在 Judge 阶段，对分类任务的精度进行检测，当精度损失达到预设阈值时，迭代剪枝过程将终止。最后，在剪枝的基础上进行模型的恢复训练，以提升最终模型的分类精度。

3.2 模型构建方法

本章提出的模型构建方法包含三个主要的训练过程。首先，在人脸识别任务上使用现有流行的轻量级神经网络模型进行预训练，并通过迁移学习将其调整为表情识别任务。接着，在表情识别数据集上对该模型进行再训练，获得未压缩但性能较好的基础模型。最后，使用本章提出的 Punish-Reward-Judge 三阶段迭代剪枝框架对模型进行剪枝训练。该框架使用门控剪枝算法作为裁剪依据，旨在实现模型的轻量化，从而有效减少计算复杂度和内存占用。图 3-1 展示了整个工作流程。

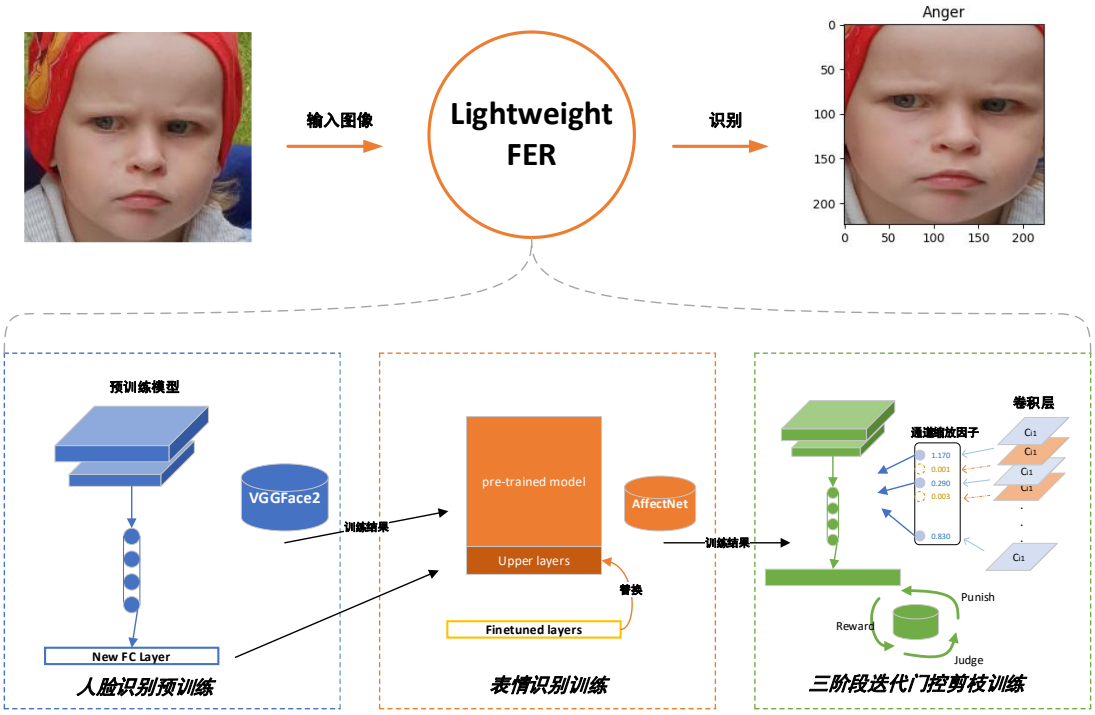


图 3-1 模型整体构建流程

3.2.1 基础模型训练

基础模型的训练包含两个阶段，其主要目标是获得一个精度较高且未经剪枝压缩的模型架构，这也是进行门控剪枝操作的前提。

（1）本研究首先基于 VGGFace2 数据集开展人脸识别任务的预训练工作。实验中所采用的模型均已在 ImageNet 数据集上完成预训练，这一策略简化了后续训练流程，同时显著提升了模型的收敛效率。

在预训练阶段，为每个网络架构的顶层引入了一个新的全连接层，该层包含 9131 个输出单元，并采用 Softmax 激活函数进行类别概率预测。为确保模型稳定性，在此阶段需要冻结基础网络的所有权重参数，仅对新增的全连接层进行训练。训练过程采用锐度感知最小化（Sharpness-Aware Minimization, SAM）与自适应矩阵估计（Adaptive Moment Estimation, Adam）优化器联合优化分类交叉熵损失函数，初始学习率设置为 0.001，训练时长为 1 个 epoch。随后，为优化模型性能，对整个网络进行 10 个 epoch 的微调训练，并将学习率调整为 0.0001。在模型选择方面，对比评估了 EfficientNet、ReXNet[63]和 MobileNetV2 等轻量级神经网络在 VGGFace2 数据集上的分类性能。实验结果表明，MobileNetV2 取得了最优的分类准确率（95.64%），被选定为后续表情识别任务的基础预训练模型。

（2）在完成人脸识别预训练之后，需要在此基础上进行表情识别的迁移学习。首先，在预训练模型的基础上引入了一个独立的表情识别头部层，并对其权重参数进行优化学习。模型更新训练分别在 AffectNet 和 RAF-DB 两个数据集上进行。考虑到 RAF-DB 数据集仅包含 7 个表情分类，为了确保后续实验的对比性以及丰富 RAF-DB 数据集的表情多样性，本章从 AffectNet 数据集中精细挑选了 600 张较为明显且具有代表性的“轻蔑”分类的图片，加入到 RAF-DB 数据集中进行训练和验证。在数据预处理阶段，采用基于眼睛位置的图像旋转对齐方法进行人脸标准化处理。针对 AffectNet 数据集存在的类别不平衡问题，本研究采用加权分类交叉熵损失函数进行优化，其权重系数根据各类别样本数量的逆比例进行调整，具体计算公式如(3-1)所示：

$$W_{sort} = \max_{c\in\{1,2,\cdots,C_e\}} N_c / N_y \tag{3-1}$$

其中，C_e 是不同表情分类类别数量；c 是训练图像的表情分类标签；N_y 表示第 y 类的训练样本总数。

在以 AffectNet 作为数据集的迁移学习阶段，模型在添加了具有 C_e 个输出单元的新头部层的同时冻结其他所有的网络层的参数，采用 SAM 优化器，对新添加的头部层进行了为期 3 个 epoch 的训练。类似地，在之后的 10 个 epoch 内解冻整个神经网络参数并对整个模型进行训练。最终结果表明，8 分类的表情任务识别精度达到了 60.53%，而 7 类主要表情任务识别精度为 64.51%。这一优化后的模型将作为后续网络剪枝流程的基础模型。

为优化模型的计算效率并降低其存储需求，本研究提出了一种硬件友好的结构化剪枝方法。具体地，将经过迁移学习得到的基础模型及其权重参数作为输入，引入门控迭代剪枝框架进行模型压缩。该方法通过系统性地移除冗余的网络结构，在保证模型性能的同时显著降低了计算复杂度和存储开销。

3.2.2 门控剪枝算法

门控剪枝算法作为对基础模型压缩的关键技术，在整体架构优化过程中扮演着重要角色。该算法通过动态评估滤波器的重要性权重，并基于评估结果对滤波器进行筛选，有效降低了基础模型的计算复杂度和内存占用。在 Punish-Reward-Judge 三阶段迭代剪枝框架中，门控剪枝算法主要应用于 Punish 阶段，其核心目标是通过识别并移除冗余滤波器来实现模型的初步压缩。门控剪枝算法的执行流程包含以下几个关键步骤。

在表情识别模型的剪枝训练中，假设损失函数为$L(X,Y;\theta)$，其中X表示输入的表情识别图像，Y为对应的标签，包含八个情感类别，θ表示基础模型的参数。对于基础模型中的滤波器集合K，剪枝的本质就是从K中选择一个子集$k \in K$，并移除对模型表达能力贡献较小的滤波器集合θ_k^r，同时保留对模型表达能力更为关键的滤波器集合θ_k^l。为了最小化剪枝后模型的损失，定义k^*为剪枝导致的损失的最小值，设剪枝后的损失增量为ΔL，则ΔL和k^*可通过以下(3-2)公式表示：

$$\Delta L = | L(X,Y;\theta) - L(X,Y;\theta_k^l) | \tag{3-2}$$

$$k^* = \arg\min_k \Delta L \quad s.t. \| k \|_0 > 0$$

其中，$\| k \|_0$表示子集k中的滤波器的数量，且至少保留一个滤波器。由于网络模型通常包含数百万个滤波器，穷举所有可能的子集k在计算上是不可行的。为此，采用泰勒展开的方法对剪枝后的损失增量ΔL进行近似地估计，从而显著降低模型计算复杂度并加速剪枝过程。

假设神经网络中特征图m是滤波器k的输出，由于批量归一化（Batch Normalization, BN）层紧接在卷积层之后，可以利用BN层中原有的缩放因子γ作为滤波器重要性的评估指标。将$\phi = m\gamma$作为判断滤波器重要性的门控因子，用于指导剪枝过程。当ϕ为0时，相当于对相应的滤波器进行了软剪枝（soft pruning），此时损失函数的增量ΔL可以表示为：

$$\Delta L_\Omega(\phi) = | L_\Omega(\phi) - L_\Omega(0) | \tag{3-3}$$

泰勒展开$L_\Omega(0)$：

$$\begin{aligned} & L_\Omega(0) \\ & = \sum_{p=0}^{p} \frac{L_\Omega^{(p)}(\phi)}{p!} (0-\phi)^p + R_p(\phi) \\ & = L_\Omega(\phi) - \phi \nabla_\phi L_\Omega + R_1(\phi) \end{aligned} \tag{3-4}$$

联合上述公式得到：

$$\begin{aligned} & \Delta L_\Omega(\phi) = | \phi \nabla_\phi L_\Omega - R_1(\phi) | \\ & \approx | \phi \nabla_\phi L_\Omega | = | \frac{\delta L}{\delta \phi} \phi | \end{aligned} \tag{3-5}$$

通过上述公式，在模型训练的反向传播过程中，可以有效地计算出损失函数的梯度，从而实现对模型参数的优化更新。

因此，对于每个滤波器 $k_i \in k$，都可以根据公式(3-5)推导出的公式计算来其重要性得分以量化滤波器的重要性，具体计算公式如下，其中 D 表示训练数据集：

$$\Phi(\phi_i) = \sum_{(X,Y)\in D} | \frac{\delta L(X,Y;\theta)}{\delta \phi_i} \phi_i | \tag{3-6}$$

在三阶段迭代剪枝框架的 Punish 阶段，基础模型通过上述门控剪枝算法对网络中的滤波器进行裁剪。具体操作流程为：首先，表情图像经过滤波器卷积操作处理后，BN 层会计算每个滤波器的统计量 ϕ，并根据这些统计量评估滤波器的重要性 Φ。利用等式(3-6)，可以在后续的反向传播过程中计算出这个结果。训练完成后，所有滤波器根据重要性得分 Φ 进行排序，并设定裁剪的阈值。接下来，筛选出得分低于阈值的滤波器，并生成相应的滤波器掩码。最后，依据该掩码，删除重要性得分较低的滤波器，完成裁剪过程。裁剪的具体步骤如图 3-2 所示。

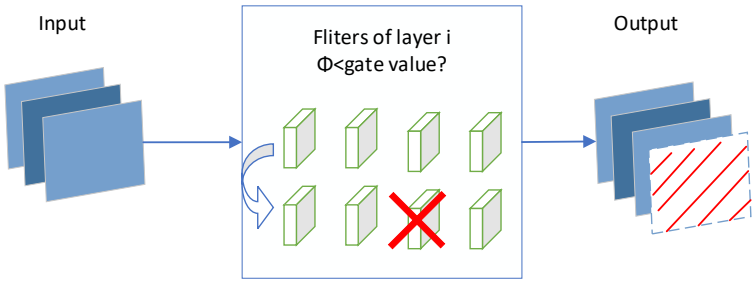


图 3-2 门控剪枝裁剪过程

3.2.3 三阶段迭代剪枝框架

在采用上述门控剪枝算法对模型进行裁剪的实验中，随着剪枝比例的增加，基础模型的精度呈现急剧下降的趋势。这种精度损失的加速增长使基础模型进行裁剪时，难以在保持较高精度的同时显著减少模型的复杂度。因此，简单粗暴地进行门控剪枝并不能取得理想的效果。为了解决这一问题，本文提出了一种创新的 Punish-Reward-Judge 三阶段迭代剪枝框架，旨在基于门控剪枝算法进一步提升剪枝的精度和有效性。该框架的核心思想是在三个阶段中分别应用不同的训练和裁决策略，逐步缓解由于剪枝所带来的精度损失。

（1）在 Punish 阶段，模型训练会冻结卷积层的权重，并仅在子数据集上进行训练。在此阶段，根据滤波器的重要性排名逐步去除那些低重要性的滤波器。此阶段的主要目标是：1）加速修剪过程，同时避免模型过拟合；2）利用泰勒展开的近似方法对滤波器进行重要性评估，并据此进行裁剪。

（2）接下来的 Reward 阶段，模型将在全数据集上进行更新训练，旨在减少滤波器裁剪所带来的误差积累。在这一阶段，还引入了 L1 正则化，即在损失函数中加入 L1 范数，促使模型参数更加稀疏，从而有效地去除不重要的通道。为了实现这一目标，修改后的损失函数如公式(3-7)所示：

$$L(X,Y;\theta) = -\log softmax(z_Y)\cdot \max_{c\in\{1,\ldots,C_e\}} N_c / N_y + \lambda \sum_{\gamma\in\Gamma} g(\gamma) \tag{3-7}$$

公式的前半部分代表了 CNN 训练中的基本损失，而 $g(\gamma)$ 作为惩罚函数的部分则作用于缩放因子，用于调整模型的复杂度。此部分引入了一个平衡因子 λ，用于控制前后两项之间的关系。通过改变 λ 的大小，可以在训练过程中调节 L1 正则化的影响。当 λ 较大时，网络模型会更倾向于学习较为稀疏的权重，即通过将一些权重值推向零来实现特征选择，从而达到稀疏化的效果。增加正则化约束，特别是对稀疏性的加强，可以更快速更准确地识别并去除不重要的滤波器。

（3）由于不同的分类任务和数据集具有不同的特点，因此很难为所有情况设定一个统一的裁剪比例。为了确保模型在剪枝过程中的精度不受到过度影响，本章提出在迭代剪枝框架的最后加入 Judge 阶段。Judge 阶段的主要作用是监控模型在剪枝过程中精度的变化，并通过设置精度损失阈值来决定是否继续进行剪枝。当剪枝后模型的精度损失超过预设的阈值，Judge 阶段会终止当前的剪枝过程，从而为剪枝过程的结束提供合理时机，防止模型因过度压缩而导致性能显著下降。迭代剪枝过程如下图 3-3 所示：

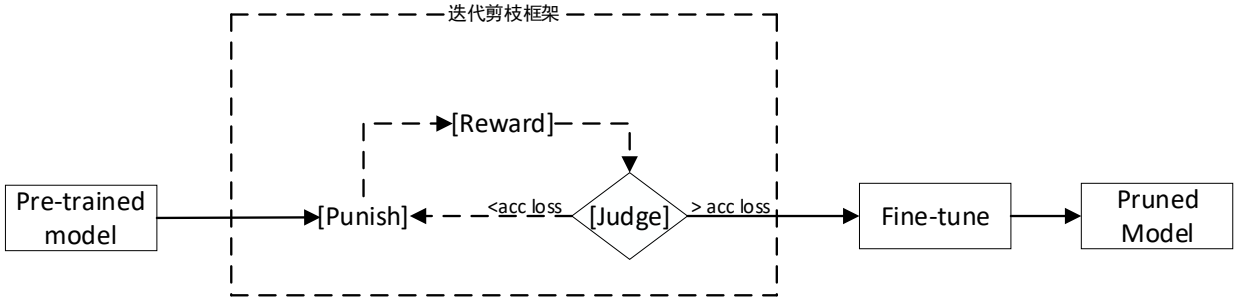


图 3-3 Punish-Reward-Judge 三阶段迭代剪枝框架

3.3 实验及结果分析

3.3.1 实验设置

本文首先比较了基础模型与最终剪枝基线模型在 AffectNet 数据集和 RAF-DB 数据

集这两个数据集上的性能差异，接着，围绕三阶段迭代剪枝框架，进行了不同剪枝策略的对比分析。主要的对比指标包括分类精度、模型参数量以及模型计算量。

基线模型训练：在基础模型的基础上进行 Punish-Reward-Judge 三阶段的门控迭代剪枝。在 RAF-DB 数据集上，批量大小设置为 128，初始学习率为 0.1。图像仅进行了中心对齐与脸部区域裁剪处理，没有进行其他额外操作。针对 AffectNet 数据集，由于数据量较大，批量大小调整为 256，初始化学习率同样设置为 0.1。在训练 AffectNet 数据集时，为了缓解数据集中的类别不均衡问题，采用了加权分类交叉熵损失函数。所有图像最终裁剪的尺寸统一为224×224。

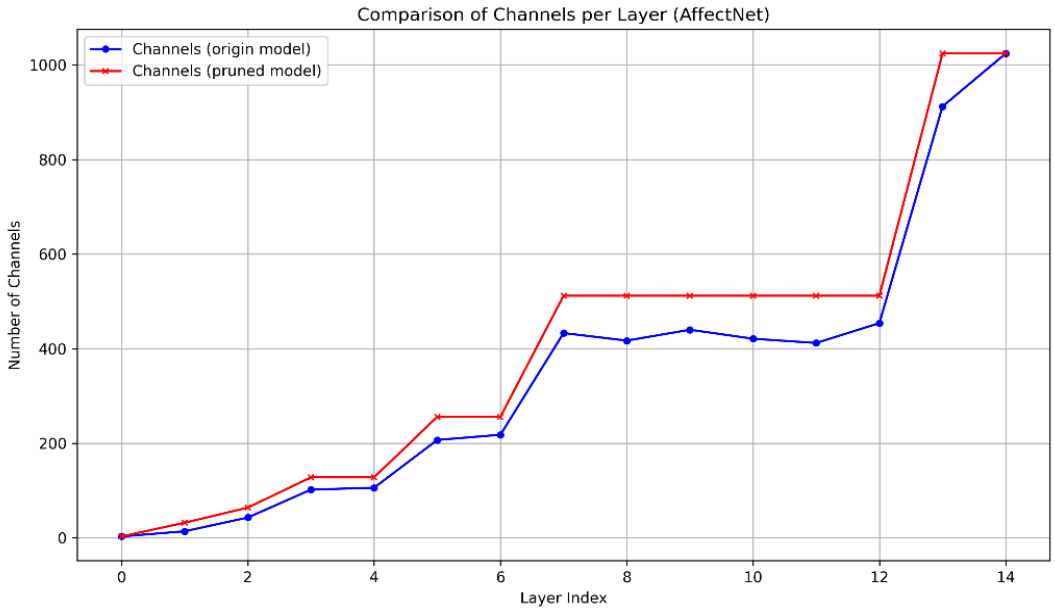
Punish-Reward-Judge 设置：在 Punish 阶段，仅对 MobileNetV2 网络中 0.2%的滤波器进行裁剪，训练过程中使用数据集的子集。对于 RAF-DB 数据集，由于数据量较小，使用了数据集中的全部数据；而对于 AffectNet 数据集，则从每个类别中随机抽取 100 张图像组成子集。在所有实验中，Punish 阶段的 epoch 设置为 10 次，即每进行 10 次 Punish 后进入 Reward 阶段。由于两个子数据集相对均衡，未使用加权分类交叉熵损失函数进行更新训练。在 Reward 阶段，使用全数据集进行训练。对于 RAF-DB 数据集，epoch 设置为 5，而对于 AffectNet 数据集，由于数据量较大，训练周期更长，epoch 设置为 2。在 Judge 阶段，进行测试集上的验证，以评估分类精度。

剪枝方法对比设置：最终以224×224大小的图像进行训练的模型作为基线模型。在对比实验中，选择了两种常见的模型剪枝方法与基线模型进行对比分析。为了确保实验的公平性，在实验设置上，三种剪枝策略都基于相同的基础模型，仅在剪枝方法上有所不同。由于实验涵盖了小规模和大规模数据集的剪枝任务，且模型训练周期较长，因此仅选择了网络瘦身剪枝（Network Slimming，NS）[64]和渐进正则化剪枝（Growing Regularization，GREG）[65]这两种常见的细粒度通道剪枝方法进行比较分析。这两种方法在小模型剪枝和大规模数据集训练中均已进行过研究并得到验证，适合在本实验中进行进一步的对比。

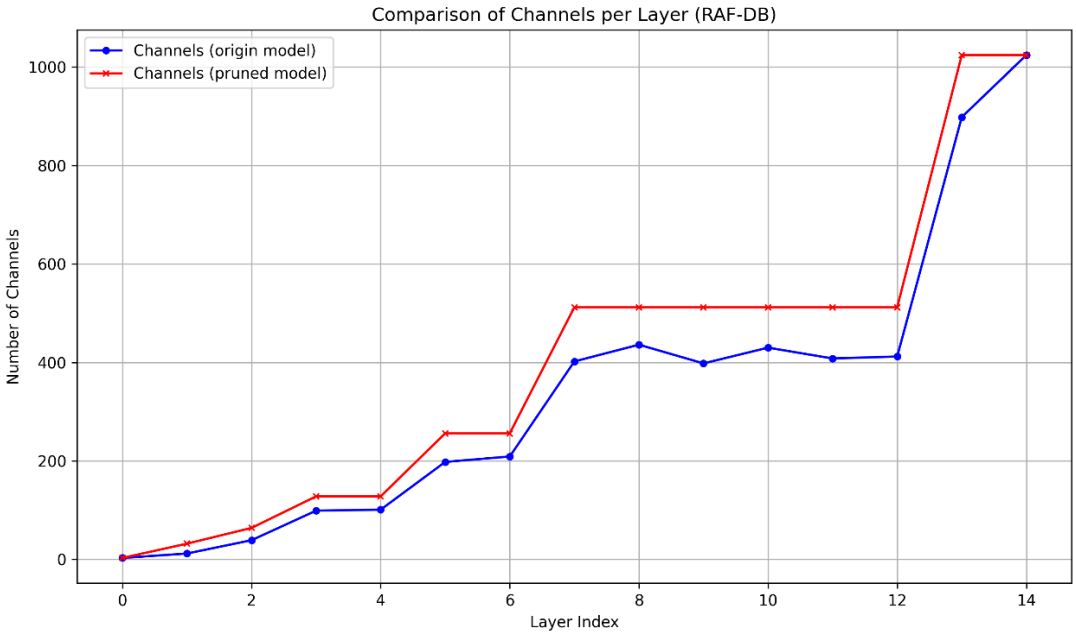
3.3.2 结果分析

表 2 和表 3 展示了在 AffectNet 数据集和 RAF-DB 数据集上，初始模型和最终经过剪枝的基线模型在性能方面的对比。性能差异主要体现在表情不同分类的识别精度、模型计算量、模型参数量和不同表情分类的识别精度四个方面。实验结果表明，在接受轻微精度损失的前提下，改进后的模型在参数量和计算量上均显著减少。在 AffectNet 数

据集这一大规模数据集实验中，实际内存占用减少了 23%。通过计算，最终模型在 AffectNet 数据集上的整体参数量为 2.81million，计算量为 762million，在 RAF-DB 数据集上的参数量为 2.45million，计算量为 694million。裁剪后的通道数量变化如下图 3-4 所示：



（a）剪枝前后各层通道数变化（AffectNet）



（b）剪枝前后各层通道数变化（RAF-DB）

图 3-4 剪枝前后模型各层通道数变化

如图 3-5 所示，经过权重加权处理后，模型显著提升了不同分类类别之间由于数据量差异而导致的精度差异。较少的计算量和参数量增强了模型对硬件的适应性，确保了

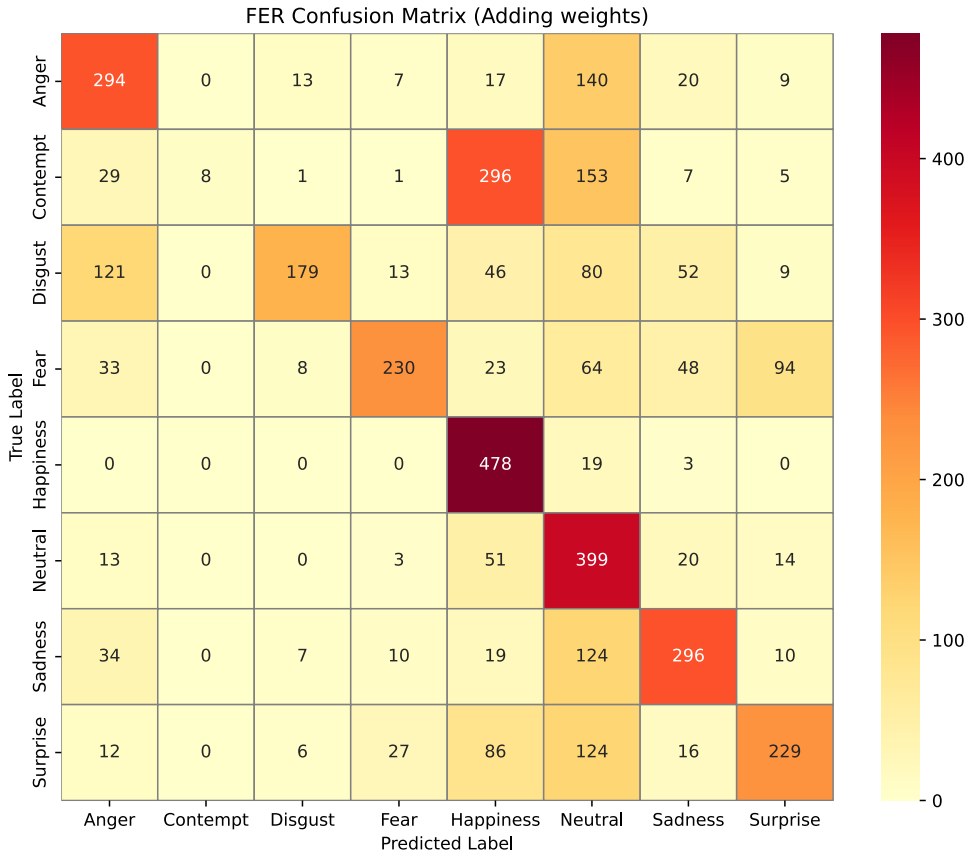
较低的部署成本，并扩展了模型的应用范围。同时，适度的精度损失也确保了模型在应用中的有效性。

表 3-1 基线模型在 AffectNet 上的对比

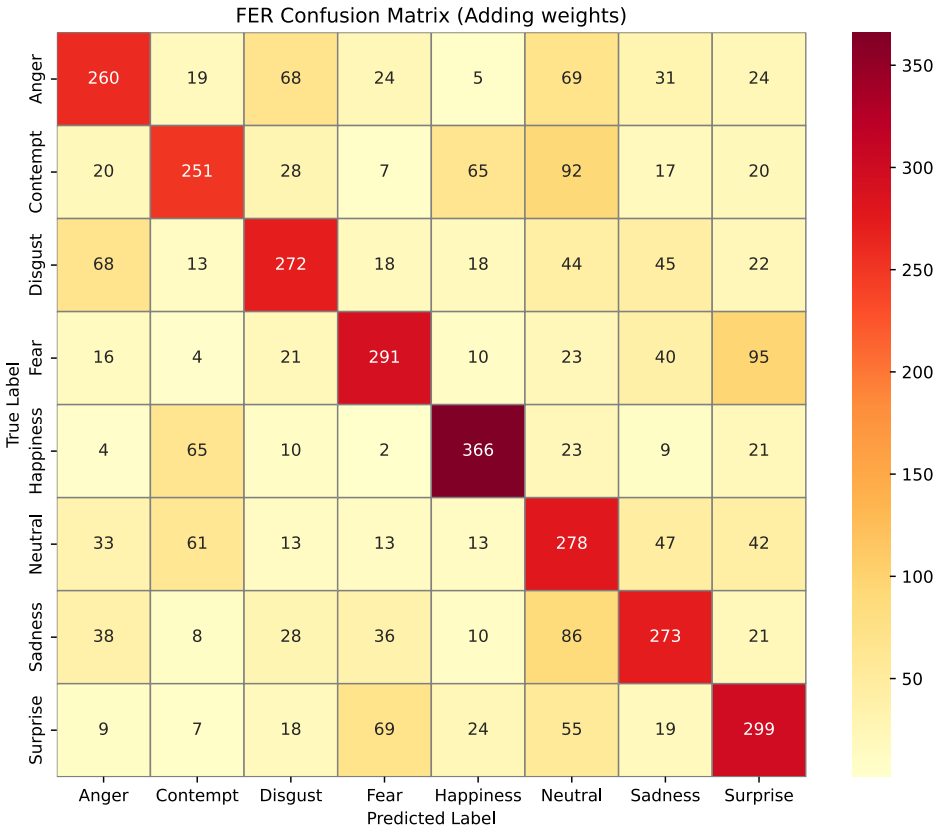
模型	8 分类精度/%	7 分类精度/%	计算量/million	参数量/million
初始模型	52.83	60.14	1150	3.49
权重损失优化后模型	60.52	65.51	1150	3.49
基线模型	59.71	63.92	762	2.81

表 3-2 基线模型在 RAF-DB 上的对比

模型	8 分类精度/%	7 分类精度/%	计算量/million	参数量/million
初始模型	85.76	87.1	1150	3.49
基线模型	83.91	86.53	694	2.45



（a）未添加权重损失模型的混淆矩阵



（b）添加权重损失模型的混淆矩阵

图 3-5 添加分类权重损失前后混淆矩阵对比

表 3-4 和表 3-5 展示了本章提出的剪枝方法与其他剪枝方法在 AffectNet 和 RAF-DB 数据集上的实验结果对比。为了确保公平性，所有实验均使用相同的初始模型，并在保持类似参数量的基础上，仅通过不同的剪枝方法对模型进行压缩。本次对比的剪枝方法包括网络瘦身剪枝（NS）和渐进正则化剪枝（GREG）。实验结果表明，在小规模数据集上，三种方法的性能差异较小。然而，在对 AffectNet 这一大规模数据集进行压缩时，本文提出的剪枝方法在相似压缩率的条件下，能够实现更小的精度损失，相较于 NS 和 GREG 具有更优的性能表现。

表 3-3 不同方法在 AffectNet 上的对比

模型	8 分类精度/%	7 分类精度/%	计算量/million	参数量/million
基线模型	59.71	63.92	762	2.81
NS	57.25	61.20	781	2.83
GREG	60.03	63.50	732	2.78

表 3-4 不同方法在 RAF-DB 上的对比

模型	8 分类精度/%	7 分类精度/%	计算量/million	参数量/million
基线模型	83.91	86.53	694	2.45
NS	85.10	86.09	688	2.44
GREG	85.33	87.12	703	2.51

3.4 本章小结

本章探讨了在表情识别任务中，通过门控迭代剪枝技术实现对轻量级表情识别模型有效压缩的方法。通过结合迁移学习和细粒度门控迭代剪枝，在模型性能略有下降的基础上，成功地实现了模型的显著压缩。这不仅大幅度降低了模型的存储和计算资源需求，也提升了其在资源受限环境中的部署能力。实验中，比较了不同剪枝策略和训练方法对模型性能的影响，并从中选择出最佳的方案。实验结果表明，使用迭代剪枝策略后，模型的性能仅微小下降，但模型参数量和模型计算量得到了显著减少。这表明该方法在移动设备和边缘计算场景中具有良好的应用价值。本章为表情识别模型的轻量化提供了一种有效的解决途径，并深入探讨了模型量化和压缩技术在实际应用中的潜在价值。

第四章 基于混合精度量化的轻量级表情识别方法

4.1 引言

在第三章中，已经通过门控剪枝对整个表情识别网络模型进行了压缩优化，显著减少了模型的参数量和计算复杂度。然而，在实际部署和应用时，模型的计算资源和存储资源在硬件上会受到严格的约束。为了进一步提升模型的部署能力，本章从硬件支持的角度出发，引入模型量化技术。量化技术相较于其他压缩方法（如剪枝、蒸馏等），在设备部署中能够展现出更大的优势，尤其是在硬件支持方面。现代硬件（如移动处理器、嵌入式设备和边缘设备）普遍支持低精度计算，而量化技术能够充分利用这一硬件特性。通过将高精度的权重参数和激活值量化为低精度参与计算，使得量化技术不仅能够显著减少模型的存储空间，还能提升模型的计算效率。量化后的模型在支持低精度计算的硬件上运行时，能够更快地完成推理任务，同时降低功耗，提升设备的能效。这种硬件友好的特性使得量化技术在资源受限的边缘设备上具有显著优势。

然而，传统的量化方法通常采用统一的量化比特数（Quantization Bit Number，QBN）来处理整个模型[66]，这种“一刀切”的量化策略往往忽视了不同层和通道之间存在的敏感性和信息重要性方面的显著差异。网络中的某些层或通道可能对量化操作异常敏感，而另一些层或通道则相对不敏感。在统一量化策略下，敏感层和通道可能会遭受较大的精度损失，从而导致模型性能的显著下降。随着量化比特位宽的降低，模型的量化误差会显著增长，尤其是在深层网络中，这种误差可能会在前向传播过程中逐渐累积并被放大，最终导致模型的准确性大幅下降。这一现象在实际应用中尤为突出。

上述问题可以通过混合精度量化技术解决。混合精度量化通过对模型的不同网络层分配不同量化比特数来更好地平衡模型从而提高计算效率和性能。更高的精度用于网络中更敏感的层，而对于低敏感网络层，则使用低精度进行优化。这种灵活的量化策略不仅能够提升模型的效率，还能在一定程度上缓解传统量化方法带来的精度损失。在本章的研究中，不单局限于网络层的量化，还提出了一种基于通道敏感性的混合精度量化（Hessian-Aware Weight Quantization with Channel Sensitivity，HAWQC）方法，进一步提升轻量级人脸表情识别模型的计算效率和存储效率。实验表明，本章提出的量化方法能在高压缩率下保持较好的性能。

4.2 混合精度量化策略

4.2.1 量化敏感度分析

在深度学习领域，神经网络架构通常采用模块化设计方法，即将网络结构划分为多个功能模块。这些模块分别表示为$B_1, B_2, \ldots, B_n$，其中每个模块B_i都包含一组可学习的参数θ_i。这种模块化的架构设计不仅提高了神经网络的灵活性和扩展性，也为复杂任务的建模提供了更强的表达能力，是当前深度学习领域广泛采用的主流架构范式。在监督学习的框架下，神经网络模型的损失函数通常被定义为训练样本损失函数的平均值。损失函数$L(\theta)$通常可以用以下公式表示：

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(x_i, y_i, \theta) \tag{4-1}$$

其中，θ表示所有模块参数的组合，即$\theta = \{w_1, w_2, \ldots, w_n\}$，$l(x_i, y_i, \theta)$是单个样本$(x_i, y_i)$的损失函数，$X$是输入集合，$Y$是对应的标签集合，$N$是训练集大小。

在实际计算中，神经网络模型的训练通常使用单精度浮点数（32 位精度）进行上述公式的计算，参与计算的内容包括网络权重和中间激活值。这种计算精度配置已成为深度学习模型训练和评估的标准规范，能够在保证模型计算效率的同时维持必要的数值稳定性。在训练完成后，神经网络模型会形成特定精度的参数分布和激活值分布。这些分布不仅涵盖了所有模块参数θ的统计特征，还包括输入数据和中间层的激活值特征。参数分布的特性对于模型的性能评估和后续优化具有重要意义，同时也是进行模型量化的基础。

量化是一种通过将神经网络中的数值（如权重或激活值）映射到预定义的离散集合，以减少数据表示范围的技术。这种方法可以通过以下公式表示：

$$Q(e) = q_j, \quad \text{for } e \in (\mathrm{t}_j, t_{j+1}] \tag{4-2}$$

其中，e表示具体的输入值，如果e在$(\mathrm{t}_j, t_{j+1}]$内，那么它会被映射为q_j（量化后的值）。其中，$j = 0, \ldots, 2^k - 1$，k是量化位数，表示具体的量化精度。

在极端情况下，当$k = 1$时，量化操作会退化为二进制量化，通常通过符号函数进行表示。此时，输入的数值被映射到两个离散的数值上，例如正值和负值。这种二进制量化方法在某些特定场景下具有其独特的优势，但在实际应用中，非二进制量化更为常见。对于非二进制量化，即当$k > 1$时，选择量化区间的划分方法就变得尤为重要。最常见的

处理方法是均匀量化，即将整个数值范围划分为等宽的区间，每个区间对应一个量化值。均匀量化函数在数值范围上的变化如下图 4-1 所示，表现为均匀的阶梯函数，其特点是简单直观，易于实现。然而，均匀量化的局限性在于它假设了神经网络中所有层和通道的浮点值分布是均匀的，这与实际情况不符。实际上，不同层和不同通道对量化误差的敏感度存在显著差异。某些层和通道对量化误差极为敏感，即使是微小的量化误差也可能导致模型性能显著下降；而另一些层或通道则对误差具有较高的容忍度，能够在较大范围的量化误差下仍然保持较好的性能。这种差异性表明，均匀量化虽然简单，但在实际应用中可能并非最优选择，需要结合具体的网络结构和任务需求，采用更为灵活和适应性的量化策略。

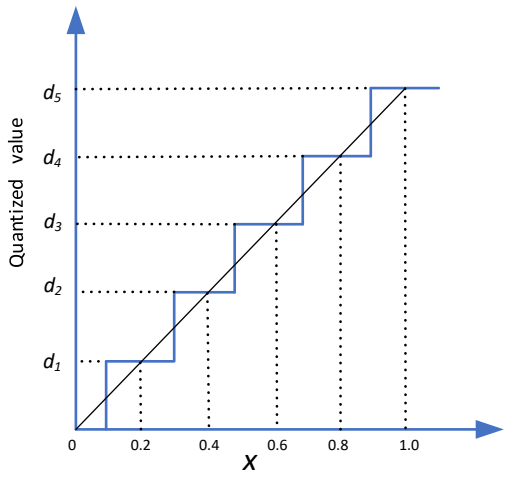


图 4-1 均匀量化函数

评估量化敏感性的一种常用方法是利用梯度向量，即一阶信息。然而，梯度并非总能准确反映敏感性，这一点可以通过一个简单的一维抛物线函数 $y = ax^2$ 在原点（即 $x = 0$ ）的例子来佐证。在原点处，无论参数 a 取值如何，梯度均为零。这一现象表明，仅使用梯度作为敏感性度量的方法存在局限性，因为梯度在此处无法反映函数对参数扰动的真实敏感性。通过引入二阶信息，即 Hessian 矩阵，可以更精确地度量敏感性。Hessian 矩阵能够捕捉函数的曲率，从而更可靠地反映出敏感性的变化。在此情况下，较大的 a 值会使得函数对数值扰动的敏感性增强。在高维空间中，Hessian 矩阵所包含的二阶信息能够更准确地量化不同层之间的敏感性差异。

二阶信息在评估各层对量化敏感性的作用的有效性已被证明。在 HAWQ-V2[69]方法中，作者通过计算 Hessian 矩阵的平均 Trace 来确定不同层对量化的相对敏感性，并根据这一敏感性排序后，采用差异化的量化精度分配策略对不同层进行量化，从而实现了

对模型量化的优化，取得了较好的实验结果。由于 Hessian 矩阵的直接计算和存储存在计算可行性问题，特别是在高维参数空间情况下，部分 Trace 很难得到精确的求解。因此，可以采用 Hutchinson 算法[70]来估算神经网络权重层的 Hessian 矩阵 Trace。计算过程如下：

Hutchinson 算法是通过高斯分布中随机采样的随机向量 z 来估算 Hessian 矩阵的 Trace。这种方法简化了计算流程，公式(4-3)中展示了这一点：

$$Tr(H) = Tr(HI) = Tr(HE[zz^T]) = E[Tr(Hzz^T)] = E[z^T Hz] \tag{4-3}$$

该公式展示了如何通过随机向量来估算 Hessian 矩阵的 Trace，其中，H 表示目标矩阵，即要估算 Trace 的 Hessian 矩阵。z 表示一个随机向量，其维度为 $d(z \in R^d)$。$E[zz^T]$ 表示随机向量 z 外积的期望。Tr 代表着 Trace 运算符号。该公式通过随机向量来估算 Hessian 矩阵的 Trace，公式(4-4)进一步给出了计算 Trace 的估算方法，即通过多次随机采样来平均估算。

$$Tr(H_w) \approx \frac{1}{m}\sum_{i=1}^{m} z_i^T H_w z_i = Tr_{Est}(H_w) \tag{4-4}$$

其中，m 表示用于估算 Hessian 矩阵 Trace 的随机样本的总数，$z_i^T H_w z_i$ 表示第 i 次采样时的矩阵 H 和随机向量 z_i 的二次型。

在深度学习的量化过程中，激活值的计算和量化同样扮演着至关重要的角色。激活值量化是指将神经网络中每层的激活值（即经过激活函数变换后的中间输出）压缩为较低精度表示，以减少存储空间占用和计算开销。与权重量化不同，激活值量化不仅需要考虑到模型权重的精度变化，还需要动态调整各层激活值的精度表示。尤其是在对象检测等复杂任务中，输入数据的尺寸和分布往往是动态变化的，这使得激活值的计算和量化过程更加复杂。因此，如何高效地计算和量化激活值，成为了在这些高复杂度任务中提升模型性能和推理效率的关键技术挑战。

在激活值量化过程中，激活值的大小不仅与网络权重密切相关，还与输入数据的分布直接关联。由于每个训练样本的输入数据 x_i 可能具有不同的尺寸或特征，这使得每层的激活值在不同输入下的表现各异。在这种情况下，激活值的量化与优化不仅要考虑精度损失，还需要确保计算效率。以下公式(4-5)是通过无矩阵 Hutchinson 算法计算激活值 Hessian 矩阵 Trace 的公式。

$$z^T H_{a_j} z = z^T \left(\nabla^2_{a_j} \frac{1}{N} \sum_{1}^{N} f(x_i, y_i, \theta) \right) z \tag{4-5}$$

其中，a_j 是第 j 层的激活值（通过非线性激活函数处理后的输出），是网络中某一层的中间表示，a_j 是输入 x_i 对应的激活值，激活值会受到输入数据 x_i 和网络参数 θ 的影响。$f(x_i, y_i, \theta)$ 表示对输入 x_i 和真实标签 y_i 的损失函数，$\nabla^2_{a_j}$ 是第 j 层激活值 a_j 对应的二阶导数。二阶导数描述了损失函数的曲率，是 Hessian 矩阵的一部分用于衡量某一参数点的二阶信息。

考虑到 H_{a_j} 是一个块对角矩阵，其结构特性使其由多个更小的子矩阵块组成，这些块按照对角线排列，每个块对应于 $H_{a_j}(x_i)$，表示为第 j 层激活值对应输入 x_i 的二阶导数。由于输入样本之间具有相互独立性，因此可以对每个输入样本单独计算相应的 Hessian 矩阵块，并通过对这些块的计算结果取均值来估计整个 Hessian 矩阵的 Trace。这种方法相较于公式(4-5)进一步简化了计算流程，使得在实际应用中更为高效和可行。具体计算方法如下：

$$z^T H_{a_j} z = \frac{1}{N} \sum_{i=1}^{N} z_i^T H_{a_j(x_i)} z_i \tag{4-6}$$

其中，z_i 表示与第 i 个输入 x_i 对应的随机向量 z 的分量。由于 Hessian 矩阵是块对角的，因此，可以只对每个输入的 Hessian 矩阵进行计算，并对结果进行平均，从而大大提高了计算效率。

4.2.2 基于 Hessian 矩阵感知的通道量化

在对 Hessian 矩阵的计算中，上述方法仅对神经网络的不同层进行敏感度估算。然而，不同通道对量化的敏感度也存在显著差异。本章通过对 HAWQ-V2 算法的改进，进一步细化到对每个权重通道的敏感度进行估算。为此，本章提出了一种基于每个权重通道来估算 Hessian 矩阵 Trace 的方法。

在之前的敏感度分析中，是通过引入随机向量 z_i 来近似估算 Hessian 矩阵 Trace。为了将这一方法扩展到不同的权重通道，本章提出使用掩码筛选通道的方式，单独对每个通道的 Hessian 矩阵的 Trace 进行计算。掩码 M^j 是针对 j 通道的二进制向量，其中第 j 通道的元素值为 1，其余通道的元素值为 0。通过将此掩码 M^j 与随机向量 z_i 相乘。得到掩码后的随机向量 z_i^j，即：

$$z_i^j = M^j * z_i \quad j \in [0, output, channels] \tag{4-7}$$

接着，Hessian 矩阵 Trace 的计算公式从对不同层的求和转变为对不同权重通道的 Trace 的求和。具体地，基于 Hessian 矩阵感知的通道量化计算公式如下：

$$Tr(H_w^j) \approx \frac{1}{m}\sum_{i=1}^{m} z_i^{jT} H_w z_i^j = Tr_{Est}(H_w^j) \tag{4-8}$$

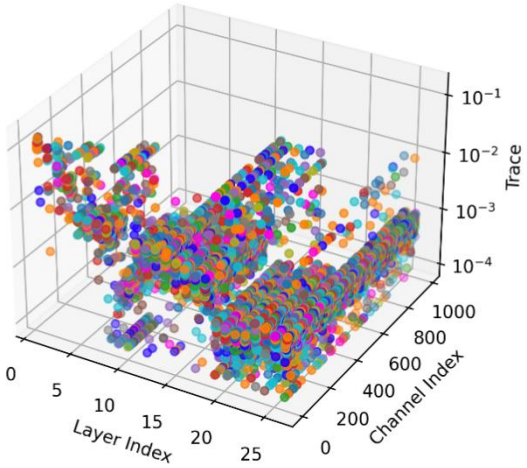
其中，m 是迭代次数，用以获取对 Hessian 矩阵 Trace 的良好估计，利用上述的随机向量和通道掩码的乘积来逐步逼近最终的估计值。通过上述方式，计算不仅可以在通道级别上进行细分，而且也没有带来额外过多的计算量。

同样地，为了进一步扩展到每个激活通道的计算，对于每个激活通道，本章采用类似的掩码操作，可以从激活层的 Hessian 矩阵的 Trace 计算中提取每个通道的独立贡献。最终得到的计算公式如下：

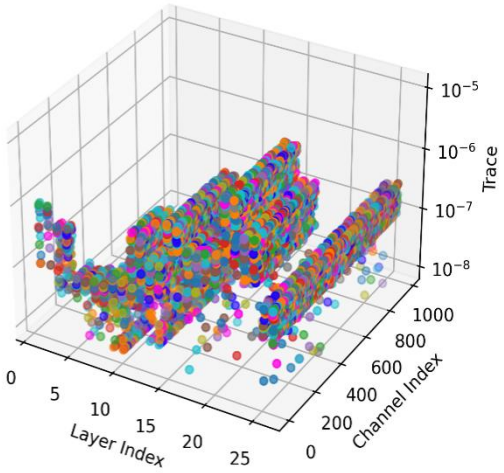
$$Tr(H_a^j) \approx \frac{1}{N}\sum_{i=1}^{N} z_i^{jT} H_a(x_i) z_i^j \tag{4-9}$$

这种方法的优势在于能够针对每个激活值通道单独进行 Hessian 矩阵 Trace 的计算，从而能够更有效地提升效率。针对通道地量化方法能够更精细地调整每个通道的参数，避免了对整个层的粗略估算。

采用上述提出的方法，计算了剪枝后的 MobileNetV2 不同通道的 Hessian 矩阵的 Trace。如图 4-2 所示，在 MobileNetV2 模型中，不同层的 Hessian 矩阵平均 Trace 差异显著，同一层内不同通道也存在较大差异。该结论适用于权重和激活值。



（a）不同权重通道的平均 Trace



（b）不同激活通道的平均 Trace

图 4-2 剪枝后模型不同通道的平均 Trace

本节提出的方法通过引入掩码的操作，将 Hessian 矩阵 Trace 的计算扩展到每个权重通道和激活通道，增强了量化过程中的进度。通过这种方式，能够更加精确地评估每个通道对于量化程度的敏感度，为混合量化方案提供了更精细的解决方案。

4.2.3 量化精度分配策略

在本章前文中，已通过估算不同通道的 Hessian 矩阵信息来评估通道对量化的敏感程度。然而，Hessian 矩阵的 Trace 无法直接提供具体的比特精度设置。因此，虽然可以根据 Trace 判断哪些通道需要更多的比特精度以提高准确性，但仍缺乏精确的比特精度分配方法。

为了克服这一不足，本章提出了一种基于强化学习的精度分配方法。在该方法中，首先通过 Hessian 矩阵的 Trace 对不同通道进行排序，从而确定每个通道的相对重要性。根据这一排序，模型可以明确哪些通道应该分配更高的比特精度。接下来，可以将这一问题建模为一个强化学习任务，并使用“演员-评论家（Actor-Critic）”模型来优化每个通道的比特精度分配策略。模型通过强化学习来决定不同 QBN 设置的比例，并通过调整整个通道的比特精度来满足压缩比和精度之间的平衡。在模型训练过程中，强化学习根据网络的性能反馈和资源约束，逐步地调整模型比特精度分配，最终找到最优的 QBN 比例。

演员-评论家模型由两个主要部分组成，演员网络（Actor）负责根据当前的状态生

成动作，即为每个通道选择合适的比特精度。在本章方法中，演员网络接收根据 Hessian 矩阵的 Trace 排序后的通道重要性信息作为输入，输出为相应的 QBN。演员网络的目标是学习如何根据当前状态最大化奖励，最终得到优化的精度分配策略。评论家网络（Critic）负责评估演员网络产生的动作的质量。评论家网络通过估计动作的价值，向演员网络进行反馈，从而指导演员网络做出更好的决策。评论家网络的输入是当前的状态和演员网络选择的动作，该网络输出演员网络动作对应的价值，用于指导演员网络在训练过程中不断调整其策略。具体的根据“演员-评论家”网络分配比特精度的示意图如下：

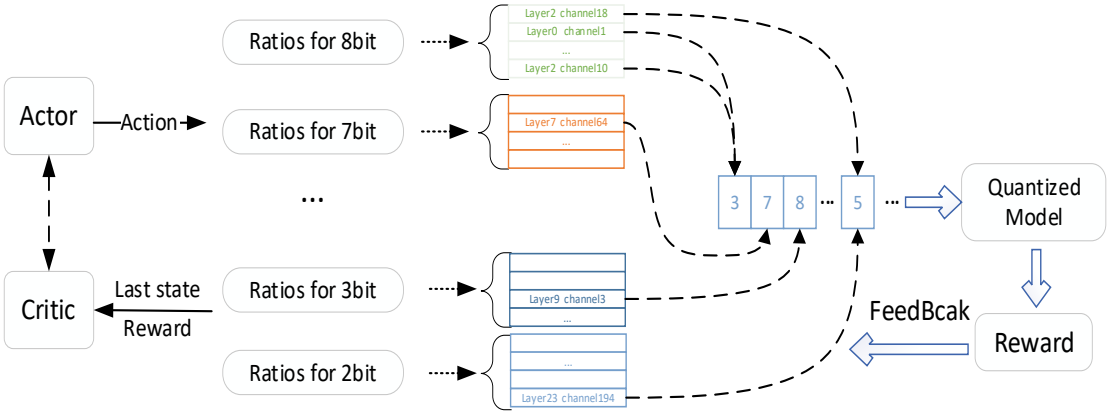


图 4-3 基于强化学习的比特分配框架

在每个训练步骤中，演员-评论家模型根据当前状态选择合适的比特精度分配策略。状态空间通过引入一个 5 维特征向量 Q_k 来表示：

$$[k, n_{re}, s_i, e_i, a_{k-1}] \tag{4-10}$$

其中，k 表示通道的索引，n_{re} 表示剩余参数的数量，s_i 和 e_i 分别表示 Hessian 矩阵 Trace 排序列表中某个通道的比特精度范围的起始位置和结束位置，这两个值定义了该通道在量化过程中所应分配的比特精度范围。a_{k-1} 表示上一步演员网络的动作，即上一个比特分配精度。这些信息共同构成了状态空间，供演员网络做出决策。在本章中，将这些特征向量做归一化处理以确保在运算时处于同样的数据规模

在实际训练过程中，本章使用连续的搜索空间来确定不同量化比特数的比例。演员-评论家模型会在多个时间步骤（time step）中迭代地学习并调整比特精度的分配策略。每一次决策都会基于当前的状态信息进行，即根据当前通道的 Hessian 矩阵的 Trace 排序和前一步的比特精度分配结果，进行下一步的量化策略调整。在每个时间步中，演员网络选择当前比特精度的比例，并通过评论家网络评估这一选择是否有助于优化模型的性能和资源消耗。

由于所有的比例和必须严格为 100%，本章模型量化的初始过程为：首先，搜索 2 位量化的比例，并根据排序后的 Hessian 矩阵的 Trace 来量化相应的通道。接着，在剩余通道中搜索 3 位量化的比例，并相应地量化这些通道。然后，以此类推，量化其他位数的通道。最后，将剩余权重量化为 8 位。

为了有效地确定各个比特量化比例，需要设置模型训练的奖惩机制。奖励函数会在每一步根据当前选择的比特精度比例计算奖励。如果分配的比例导致资源超标或者精度不符合要求，奖励会下降，迫使模型调整策略。而当模型在训练后期能够达到更高精度和更小资源消耗，奖励则会增加。这种反馈机制会促使模型在训练过程中不断优化比特精度的分配策略。在演员-评论家模型训练过程中，两个网络共享相同的网络架构，对于演员网络，扩展了一个额外的 sigmoid 层，用于输出向量的归一化。在学习率设置上，演员网络学习率设置为10^{-4}，评论家网络设置为10^{-3}。网络使用批量大小为 128 进行训练，重放缓冲区设置为 600。网络学习过程中，首先进行 50 个 episode 的探索训练，使用常数噪声 0.5 用于辅助学习，然后在后续 600 个 episode 回合中使用指数衰减噪声让模型对实际知识进行学习。

4.3 实验及结果分析

4.3.1 实验设置

为了系统地评估基于通道的混合精度量化算法的性能，本章构建了一个多维度的实验验证体系，涵盖了模型选择、数据集选择、性能评估等多个方面。实验的模型选择策略旨在验证算法在不同模型复杂度和架构特性下的适应性，并确保在多种实际应用场景中的鲁棒性和泛化能力。

在模型选择方面，本章采用了两种具有代表性的对比架构进行评估：其一为第三章中提出的未剪枝的基础模型（基于 MobileNetV2 改进架构），并使用 AffectNet 数据集进行训练和测试；其二为经典的 ResNet-50 模型，使用 ImageNet 数据集进行评估。这样设计的目的在于验证所提出的混合精度量化算法在面对不同复杂度的模型和多样化的任务时的表现，尤其是在面对 AffectNet 数据集（专注于面部表情识别）时，算法的通用性和鲁棒性。

在实验中，选取了 AffectNet 和 ImageNet 这两个数据集作为基准测试平台。AffectNet 数据集具有丰富的情感分类标注，适合用于情感分析类任务；而 ImageNet 则是一个标

准的图像分类数据集，广泛用于测试模型的通用性与性能。这两个数据集在数据分布的多样性和标注一致性上具有显著优势，为量化算法在实际应用中的鲁棒性提供了可靠的验证基础。

在性能指标方面，本章从任务性能、存储效率和部署可行性三个维度进行全面评估。具体评估指标如下：

（1）分类准确度：在 AffectNet 数据集上进行实验与验证时，本章使用 7 类表情识别精度作为主要性能指标，这些表情分别为中性、快乐、悲伤、惊讶、恐惧、厌恶和愤怒。分类准确度是衡量量化后模型在保留任务精度方面的关键指标，特别是在情感分析等任务中，模型在不同类别上的表现将直接影响到其应用效果。

（2）权重压缩比（W-Comp）：该指标用于衡量量化模型相较于基准模型的存储空间压缩效果。较高的压缩比意味着量化后的模型在存储方面节省了更多空间，从而更适合在资源受限的边缘设备上部署。通过该指标，可以评估算法在压缩效率方面的优势，尤其是在存储要求严格的实际应用中。

（3）模型大小：记录经过混合精度量化后的模型存储体积（单位为 MB）。该指标直接反映了量化后的模型在存储上的实际需求，是在边缘设备部署时的重要参考指标。更小的模型大小使得部署更加灵活，能够有效减少存储资源的消耗，特别是在内存和存储有限的设备上。

此外，为了进一步探讨剪枝与量化的结合效果，本章还分别对剪枝前后的模型进行了量化后的结果对比。剪枝操作通常能够减少模型的冗余，进而提高量化后的效率和存储节省。通过对比剪枝前后的量化结果，本文揭示了剪枝和量化结合对模型大小和精度的影响，并为量化优化提供了理论依据与实践指导。

通过上述实验设计和指标评估，本章不仅验证了基于通道的混合精度量化算法在多个不同模型和数据集上的有效性，还综合评估了其在存储压缩、精度保持和部署可行性等方面的表现，为后续的优化研究和实际应用提供了参考。

4.3.2 结果分析

本节对基于通道的混合精度量化算法在多种模型和数据集上的实验结果进行了系统的分析，重点关注了分类精度、存储效率（包括权重压缩比与模型大小）以及部署可行性等关键维度。通过对比实验，本章评估了该量化方法在人脸表情识别任务中的适用性与优越性，为其在实际应用中的潜力提供了科学依据。

表 4-1 展示了在 ImageNet 数据集上，使用 ResNet-50 模型并采用不同量化方法优化基线模型后的效果对比。实验结果表明，虽然本章所提出的方法对模型进行了大规模的压缩，但是精度仅仅出现了轻微下降。这一结果表明，所提出的方法在移动设备和边缘设备上具有良好的适用性和实用价值。与其他量化方法相比，本章提出的量化方法在精度损失、权重压缩比以及模型大小方面均表现出较为明显的优势，进一步验证了其在高效压缩和精度保持之间的优良平衡。

表 4-1 ResNet50 在 ImageNet 上的量化结果

Method	ACC	ACC drop	W-Comp	Size(MB)
BaseLine	76.45	NA	1.00x	97.8
Uniform[71]	73.93	2.52	8x	12.23
HAWQ-V2	74.82	1.63	12.24x	7.99
AutoQ[72]	74.12	2.33	14.48x	6.75
HAWQC	74.88	1.57	14.48x	6.75

表 4-2 展示了在 AffectNet 数据集上，使用未剪枝的 MobileNetV2 模型并采用不同量化方法优化后的效果对比。实验结果表明，所提方法在表情识别任务中具有较好的有效性：在仅轻微牺牲模型精度的情况下，成功实现了模型体量的大幅下降，验证了模型压缩的有效性以及其在实际部署中的可行性。与其他量化压缩方法相比，本章提出的量化方法在基于 MobileNetV2 模型和 AffectNet 数据集的实验中，仍然展现出了较为优越的性能。

表 4-2 MobileNetV2 在 AffectNet 上的量化结果

Method	ACC	ACC drop	W-Comp	Size(MB)
BaseLine	65.51	NA	1.00x	13.51
Uniform	62.43	3.08	8x	1.69
AutoQ	62.92	2.59	9.93x	1.36
HAWQC	63.13	2.38	9.93x	1.36

为了进一步地探讨结构剪枝后的模型进行量化的可行性，本章将剪枝后的最终模型输入到混合精度量化方法进行压缩，并与未剪枝的 MobileNetV2 模型进行性能上的对比。对比结果如图 4-3 所示。实验表明，在剪枝之后再进行量化是可行的。剪枝与量化相结合不仅有效地减小了模型体积，还在一定程度上保持了模型的精度，验证了该方法在实际应用中的可行性和优势。

表 4-3 剪枝前后模型在量化后的性能差异

Model	ACC	ACC drop	W-Comp	Size(MB)
MobileNetV2	63.13	2.38	9.93x	1.36
Pruned MobileNetV2	63.35	2.16	7.65x	1.36

通过在 ImageNet 和 AffectNet 数据集上进行实验，本章验证了所提混合精度量化方法在不同模型和任务中的有效性与优势。实验结果表明，尽管模型进行了大规模的压缩，精度仅出现了轻微下降，这表明所提出的量化方法能够在压缩模型体积的同时，保持较高的精度，尤其适用于资源受限的移动设备和边缘设备。与其他量化方法相比，本章提出的方法在精度损失、权重压缩比以及模型大小方面均表现出较为明显的优势，进一步验证了其在高效压缩和精度保持之间的优良平衡。

4.4 本章小结

本章详细探讨了神经网络模型的混合精度量化算法，提出了一种基于 Hessian 矩阵感知的通道量化方法（HAWQC）。该方法通过为不同通道的权重和激活值分配不同比特数，在确保模型精度的前提下，有效地提高了模型性能并优化了存储效率，从而实现了更好的压缩效果。

同时，本章还介绍了利用强化学习进行比特分配的方法，采用“演员-评论家（Actor-Critic）”模型优化量化策略，进一步改善了量化过程中的精度与计算效率的平衡。为了验证提出方法的有效性，本章还设计了一系列对比实验，实验结果表明，本章提出的混合精度量化方法在不同模型和数据集上均表现出了存储效率的提升，并且在相同压缩比下，模型的精度损失较小，优于现有的其他方法。

第五章 人脸表情识别安卓应用

为验证面部表情识别模型在实际应用中的可行性，本章开发了一款安卓端表情识别应用（FER Application，FER APP）。该应用基于在 AffectNet 数据集上训练的轻量级人脸表情识别模型，能够将输入的图像分类为“开心”、“愤怒”、“厌恶”、“悲伤”、“轻蔑”、“恐惧”、“中性”和“惊讶”这八种情绪。由于 PC 端训练的模型不能直接在安卓设备上使用，需要将 PyTorch 模型（.pt 格式）转换为适合 PyTorch Mobile 的 PTL 格式（.ptl 格式）。接着，使用 PyTorch Android Lite 解释器在安卓设备上进行推理。为实现这一目标，使用 Android Studio 开发了表情识别应用，使得应用能够对静态图像和实时摄像头视频流中的人脸表情进行自动检测和识别。通过此过程，成功展示了轻量级模型在移动设备上的应用潜力。

5.1 系统设计方案

5.1.1 系统开发流程

如图 5-1 所示，系统的构建主要包含两个关键开发阶段：首先是模型训练阶段，该阶段的研究内容已在第三、四章详细阐述，重点在于构建并优化轻量级面部表情识别模型，为后续系统实现奠定算法基础。

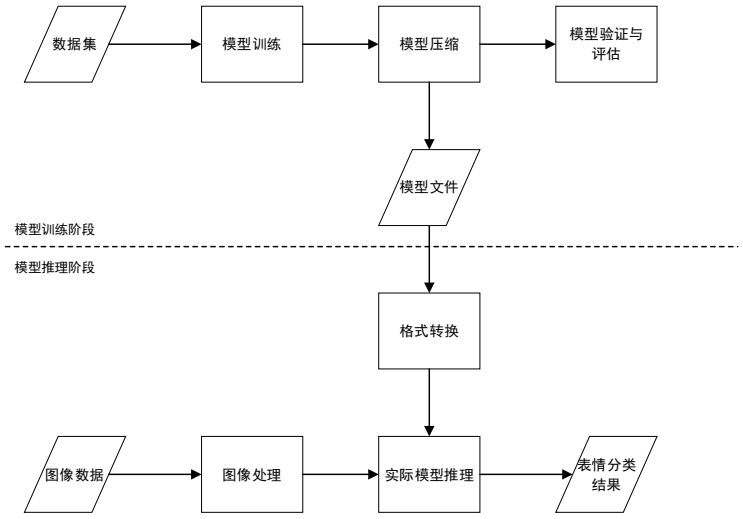


图 5-1 开发流程示意图

在模型推理阶段，为了将模型成功部署到移动端并实现高效推理，需要对网络模型进行适配和转换操作。由于人脸识别模型是在 PC 端使用 PyTorch 框架训练得到的，因

此直接将其应用于移动设备存在不同端的差异限制，必须模型格式进行转换以适应移动设备。具体而言，需要将原始的 PC 端模型（.pt 格式）转换为 PyTorch Mobile 支持的 PTL 格式（.ptl 格式）。这种转换是为了确保模型在移动设备上能够顺利运行，同时也能有效地利用设备的计算资源。在转换过程中，首先会使用 PyTorch 提供的工具将训练好的 PyTorch 模型进行格式转换，生成一个适合移动端使用的推理模型。

完成模型转换后，接下来的步骤是将转换后的模型集成到安卓应用中。在应用运行时，首先需要对输入的图像进行必要的预处理，包括对图像进行尺度变换、区域截取以及像素值归一化等方法，将原始图像转换为符合深度学习模型输入规范的标准化数据。之后，应用会调用已经集成的 PyTorch Mobile Lite 对模型进行推理，利用移动设备的计算能力进行实时的表情识别。在推理过程完成后，应用将输出识别结果，例如检测到的情绪类别，如“开心”、“愤怒”、“厌恶”、“悲伤”等。通过这一系列的转换与适配操作，最终能够实现高效的表情识别功能，确保模型能够在安卓设备上顺利运行，并提供快速、准确的推理结果。

5.1.2 开发环境

深度学习模型通常是在大型服务器端框架（如 PyTorch 和 TensorFlow）上进行构建和训练的。然而，由于这些框架的规模庞大以及依赖环境复杂，它们并不适用于在移动端设备上进行直接部署。因此，需要将在 PC 端训练完成的深度学习模型进行格式转换和结构优化，使其能够适配移动端推理框架，并在适配的移动端深度学习框架上进行推理。通过这种方式，能够克服框架规模和依赖环境的限制，实现移动端上的高效推理。

随着移动设备计算能力的迅速提升，市场上涌现了多个专为移动端和边缘设备设计的深度学习框架，如 TensorFlow Lite[73]、Core ML[74]、NCNN[75]和 MNN[76]。由 Google 公司团队研发的 TensorFlow Lite 框架，其核心优势在于提供了完整的模型转换工具链，能够将 TensorFlow 训练模型高效地转换为适用于移动端或者边缘端的轻量化版本，显著降低了模型部署的复杂度。Apple 公司团队推出的 Core ML 框架凭借其与 iOS 系统的深度集成，实现了终端设备的离线机器学习能力，该框架自 2017 年发布以来，已在苹果生态系统中得到广泛应用。腾讯公司的团队开发的 NCNN 框架以其轻量化和高兼容性著称，该框架不依赖任何第三方库，能够支持主流卷积神经网络架构，为移动端 AI 应用提供了灵活的技术选择。凭借其在移动设备上的卓越性能，NCNN 已成为一个优秀的国产开源框架。

系统开发采用 PyTorch Mobile[77]作为移动端深度学习框架，该框架由 Facebook 人工智能研究团队（FAIR）开发，是 PyTorch 生态系统的重要组成部分。PyTorch 框架最初由 FAIR 团队于 2016 年发布，为满足移动端深度学习推理的需求，研究团队于 2019 年推出了 PyTorch Mobile 扩展模块。该框架的主要优势在于能够简化 PyTorch 模型向 iOS 和 Android 平台的部署流程，并为移动端推理任务提供高效支持。PyTorch Mobile 的推出使得计算机视觉、自然语言处理等复杂深度学习任务在移动设备上的实现成为可能。该框架提供了一系列针对移动设备的优化工具，有效解决了移动端计算资源受限的问题，从而确保了深度学习模型在移动设备上的高效运行。通过使用 PyTorch Mobile，开发者可以便捷地将训练完成的 PyTorch 模型部署到移动设备上，并能够实现较为高效的推理计算。

人脸表情识别 APP 的集成开发环境采用 Android Studio，这是谷歌公司发布的官方安卓应用开发平台。Android Studio 作为当前 Android 平台的主流开发工具，为应用程序开发提供了全面的技术支持。该开发环境集成了代码编写、智能补全、实时调试和布局预览等核心功能，这些功能的有机结合大幅提升了开发效率。在应用构建方面，开发者可以通过 Gradle 构建工具定制项目结构，灵活生成适用于各类 Android 设备的应用程序包。此外，该平台配备的多功能模拟器能够仿真不同品牌和型号的 Android 设备，涵盖手机、平板和智能电视等多种终端类型，为应用的兼容性测试和性能调优提供了可靠保障。这些功能的系统集成使得 Android Studio 能够为移动应用开发提供全生命周期的支持，不仅简化了开发流程，也确保了应用质量。

图 5-2 展示了 PC 端模型的处理流程。首先，网络模型在计算机平台上进行训练，训练代码采用 Python 编程语言，并基于 PyTorch 框架实现。训练完成后，将生成的 PyTorch 模型转换为适用于移动设备的 PTL 格式。此格式转换是模型从训练环境到移动端部署的关键步骤，确保了模型能够高效地迁移并在移动端进行应用，从而实现模型在不同平台上的一致性和优化性能。

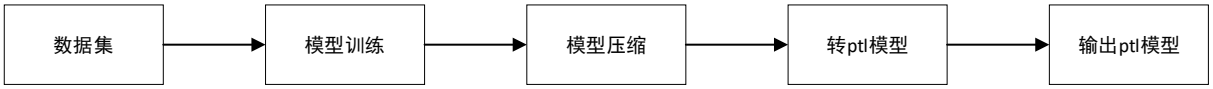


图 5-2 PC 端输出 PTL 模型

图 5-3 展示了基于 PyTorch Mobile 推理引擎的移动端的处理流程。在推理阶段，系统首先对输入数据执行与训练阶段相同的预处理操作，以确保数据处理的一致性。这些步骤包括图像缩放和像素归一化等，以确保输入数据符合模型的要求。

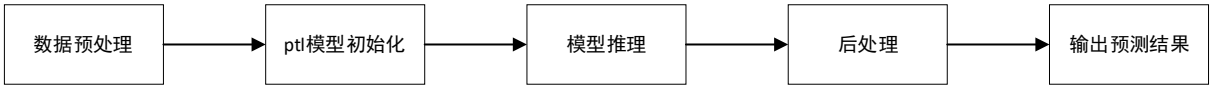


图 5-3 安卓端使用 PTL 模型推理过程

5.1.3 系统概述

开发的面部表情识别应用支持 Android 9.0 及以上版本，系统架构如图 5-4 所示。应用采用模块化设计，包含两个主要功能：静态图片识别和动态实时视频分析。静态图片识别功能允许用户选择本地图像进行面部表情分析，而实时视频分析功能通过设备摄像头实现连续帧的面部表情识别。两者均配备图形化结果显示界面，便于用户直观查看识别结果。

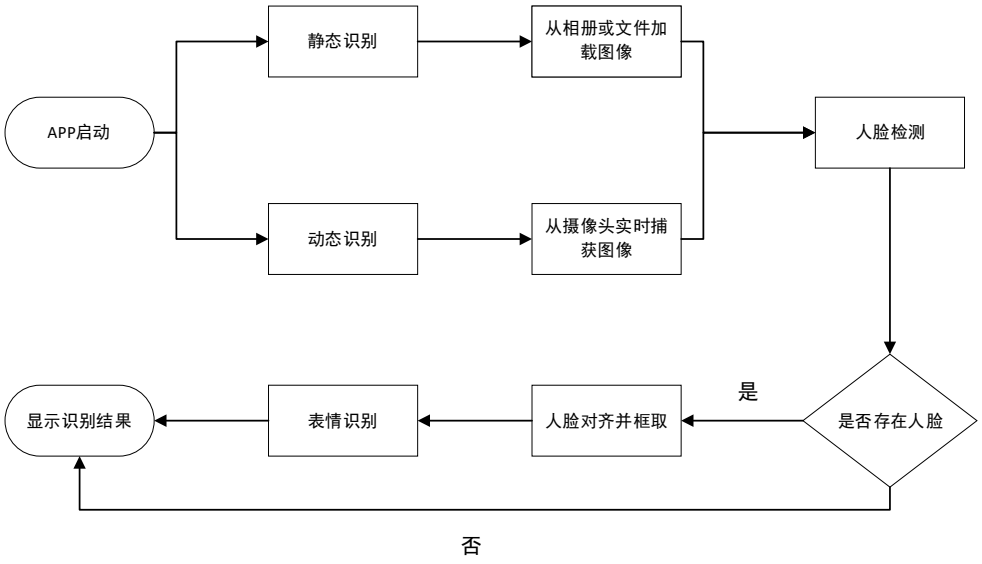


图 5-4 APP 的总体流程图

在静态图像识别功能中，用户可以通过应用程序访问设备相册或文件系统，选择待被分析的图片文件并启动识别功能。系统首先执行人脸检测算法，当系统未定位到人脸时，界面仅显示原始图像，无任何识别信息；若多任务级联卷积网络（Multi-task Cascaded Convolutional Networks，MTCNN）[78]算法成功识别出人脸区域，系统将继续执行后续处理步骤：首先对人脸区域进行空间校正，接着绘制人脸边界框，并进行表情分类分析。分析结果将实时显示在应用界面上。这种层次化的处理流程确保了系统的鲁棒性，并优化了用户的交互体验。

在实时识别模块中，系统通过相机实时捕捉图像并进行人脸检测。相机预览由 CameraX 库提供，配置为前置摄像头。为了处理相机捕捉的图像，系统将其转换为 Bitmap

对象，具体步骤如下： 首先，从相机图像获取字节数据，数据形式为 YUV 格式。接着，将 YUV 数据转换为 JPEG 格式，然后通过 BitmapFactory 工具类将 JPEG 字节数组转换为 Bitmap 对象。若相机图像的方向与设备屏幕不一致，系统会使用 Matrix 对 Bitmap 进行旋转，以确保图像正确显示。一旦检测到人脸，系统会进行人脸对齐处理并进行表情识别，最终将识别结果显示在 APP 的 UI 界面上。实时识别模块的操作过程与静态识别模块类似。

5.2 系统功能实现

5.2.1 系统页面布局

本小节介绍了系统的整体架构。系统的主页面布局如图 5-5 所示，功能划分为静态表情图像识别模块和基于摄像头的动态图像识别模块。

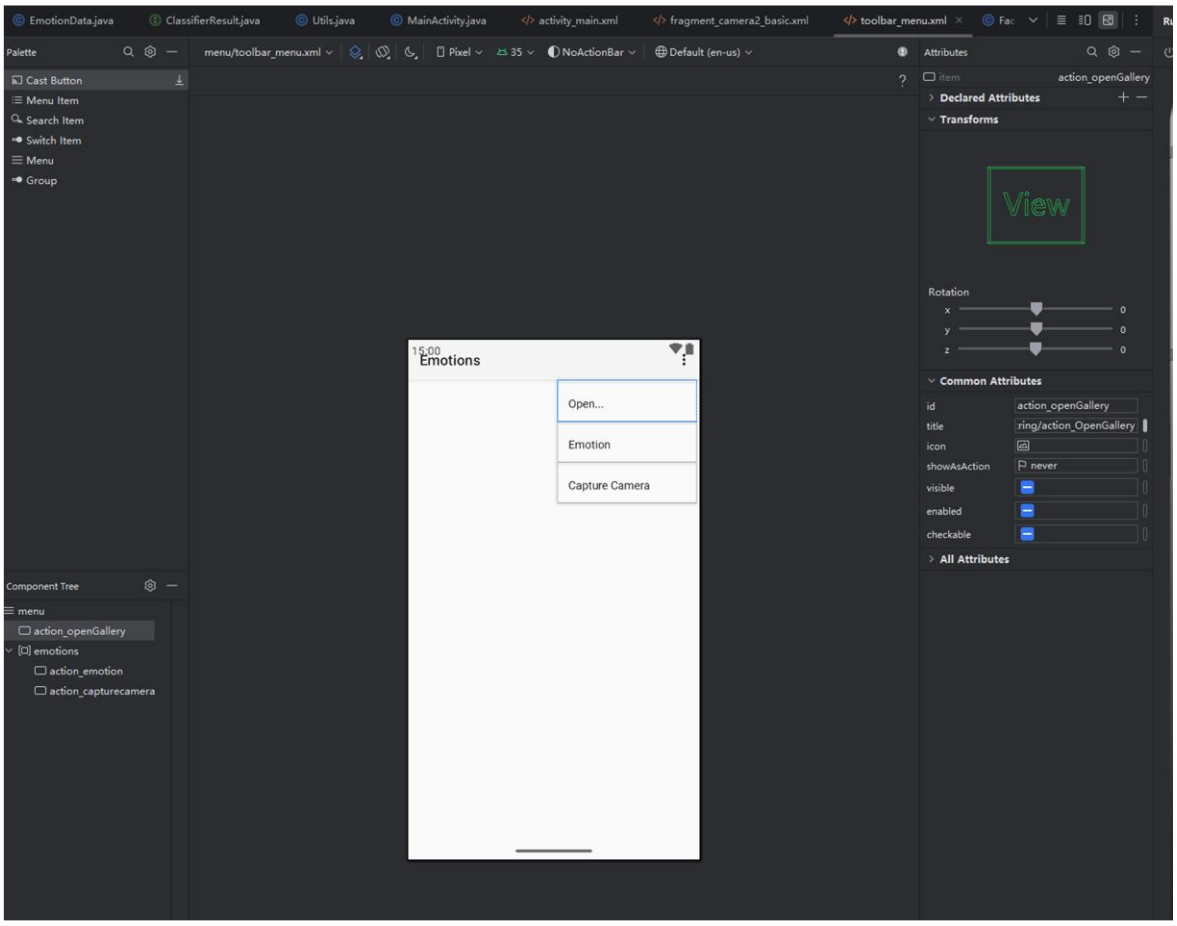


图 5-5 系统主页面布局

在未进行任何按钮操作时，图像显示区域及表情识别结果将保持为空白状态。用户需要通过从系统文件或相册中上传图片，才能进行静态表情识别。此操作可通过点击“Open...”按钮控件来完成。上传图像后，用户可按下“Emotion”按钮控件启动表情识别过

程，系统将对上传的图像进行处理并展示识别结果。通过这种交互方式，用户能够方便地选择图像并实时获取表情识别的反馈。

当用户点击“Capture Camera”按钮时，将切换至动态识别模式。在该模式下，用户可以在页面上看到摄像头实时传出的图像信息。此时，系统将启动实时图像捕捉和表情识别功能。如果用户再次点击此处按钮“Stop Camera”，动态识别将停止。这一操作使用户能够灵活控制实时识别过程，确保在需要时可以启动或停止识别功能。

5.2.2 基于静态图像的人脸表情识别

本节将详细介绍静态图像识别模块的处理流程。如图 5-6 所示，基于静态图像的表情识别系统包括六个处理环节：图像加载、图像对齐、图像缩放、人脸检测与对齐、表情识别和结果显示。该处理流程涵盖了静态图像表情识别的各个关键步骤，每个环节都经过优化设计，以确保系统的准确性和稳定性。

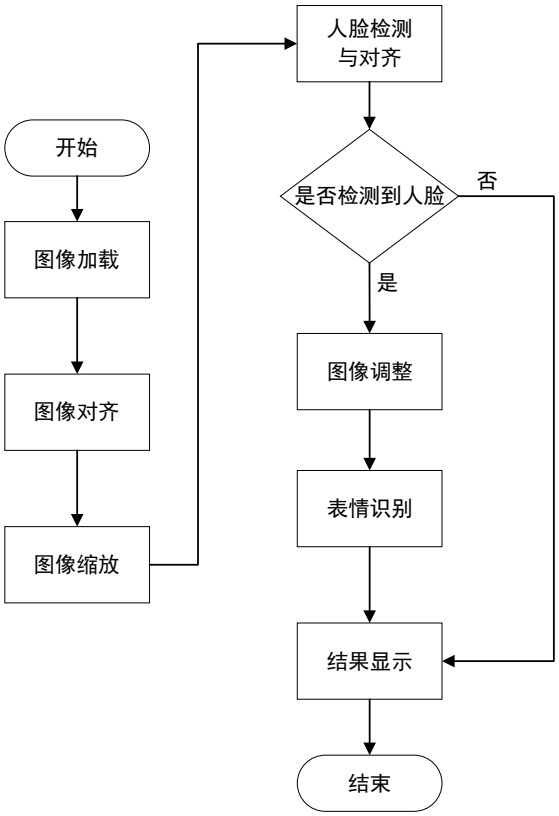


图 5-6 系统静态图像识别流程

（1）图像加载：图像加载是从客户选择的文件中获取数据，并转换为可以处理的 Bitmap 文件类型。当用户选择图片之后，程序通过获取图像的 Uri 来访问图像文件。接着，程序将图像数据读取进内存，并将其解码为 Bitmap，保证从图片中提取的字节数据能够正确转换为 Bitmap 对象。

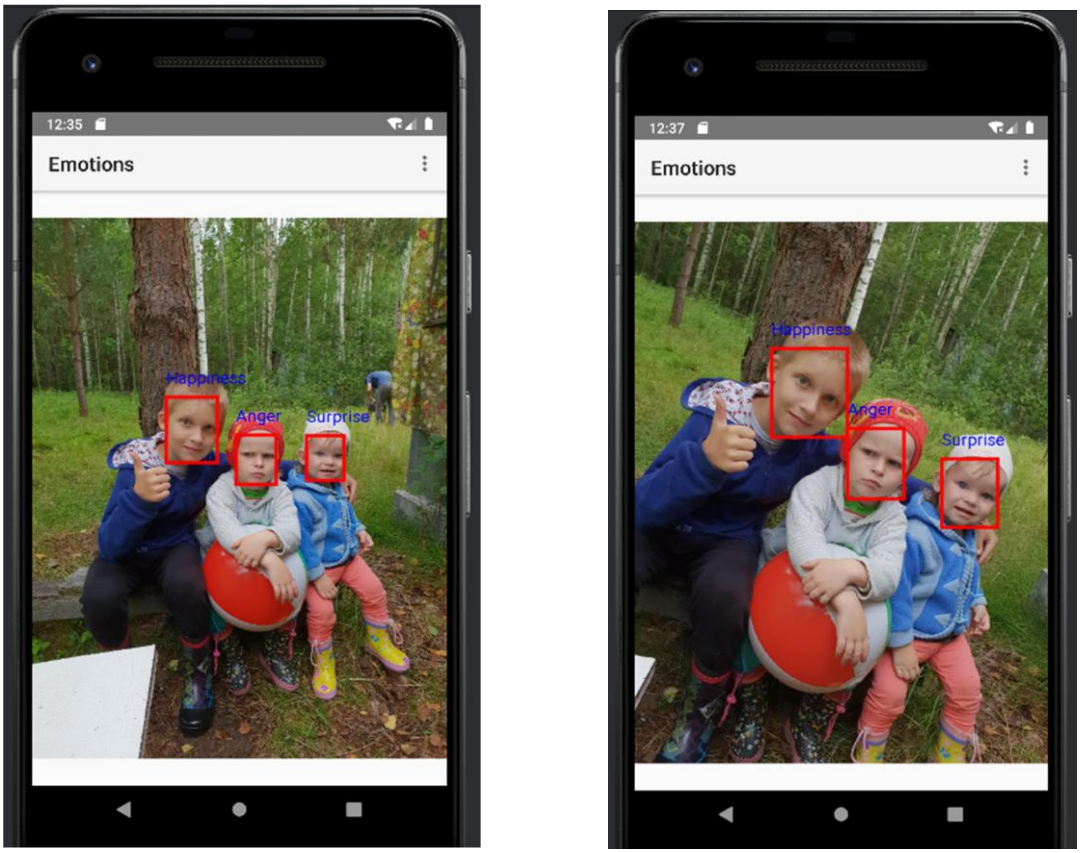
（2）图像对齐与缩放：许多相机拍摄的图像可能包含 EXIF 信息（可交换图像数据），其中包含图像的旋转信息。为了确保图像的显示方向正确，程序通过检查图像的 EXIF 信息确定图像是否需要旋转。旋转的角度包括 0°、90°、180° 和 270°。如果旋转角度不是默认的 0°，则程序会将图像旋转到正确的位置。这样，无论原始图像拍摄时的方向如何，程序始终能确保显示出正确的图像。为了适应人脸检测模型的输入要求，图像需要被适当缩放。首先，程序会设定一个最小尺寸 minSize，值为 600.0px，代表图像的最小边长。然后根据图像长宽和最小边长计算出图像的缩放比例，从而对图像进行缩放。缩放的目是保证在接下来的人脸检测中能够在图像中识别出最够清晰的特征。

（3）人脸检测与对齐：人脸检测是整个图像处理流程中的关键步骤。程序使用 MTCNN 进行人脸检测。该方法调用 MTCNN 模型对图像进行处理，返回一个 Box 对象，其中每个 Box 表示一个检测到的人脸的边界框。通过人脸检测模型，程序能够有效地定位图像中所有人脸区域，并为每个区域根据 Box 对象绘制一个矩形框。然而，MTCNN 输出的边界框是相对于缩放后的图像的，而原始图像的尺寸可能有所不同。因此程序需要根据缩放比例调整边界框的位置和大小，以便准确还原到原始图像的坐标系中。MTCNN 的人脸对齐是在检测到的每个人脸框的基础上进行的。具体来说，人脸对齐通常是通过将检测到的面部关键点与标准的参考模型对齐来完成。MTCNN 通过标准关键点位置和仿射变换实现这一目标。

（4）表情识别：在完成人脸区域提取与标准化预处理后，系统将处理后的面部图像输入基于深度学习的表情分类模型。该分类器采用经过迁移学习优化的卷积神经网络架构，通过前向传播计算获取表情特征的空间分布。模型推理完成后，输出层通过 Softmax 函数生成八维概率向量（对应八种表情类别），系统选取置信度最高的类别作为最终识别结果。

（5）结果显示：如果没有检测到人脸，图像上不会框出人脸信息也不会显示预测结果，如果检测到人脸，根据人脸检测模型输出的人脸框和表情识别模型输出的表情标签信息。最后在原始图片上绘制出人脸框，使用户可直观地看到应用检测到的人脸位置。同时将表情识别结果绘制在人脸框旁边。

通过上述五个步骤，该应用可以实现对静态图片的表情识别功能。如图 5-7 (a)所示，当未检测到人脸时，应用会直接输出对应提示信息。检测到人脸的效果图如图 5-7 (b)所示。



（a）　　（b）

图 5-7 静态识别的效果图：（a）人脸图像识别；（b）有旋转角度的人脸图像识别

5.2.3 基于实时摄像头信号的人脸表情识别

在论文 5.2.2 小节中，基于静态图像的人脸识别流程与基于实时摄像头信号的人脸识别流程具有相似性，二者的主要区别在于图像获取方式。基于静态图像的识别方法通常通过加载系统存储中的图像文件进行处理，而基于实时摄像头信号的识别方法则是通过捕获实时摄像头信号流中的一帧图像，进而进行表情识别。这一差异主要体现在数据来源的实时性上。基于摄像头信号的动态人脸表情识别的流程如图 5-8 所示。

系统首先通过摄像头实时捕捉图像，借助 Android 的 CameraX API 对视频流进行处理。每一帧图像在提取后会传递至后续处理模块。在 Android 程序中，使用 CameraX 进行图像分析时，为避免主线程阻塞并保持应用流畅性，图像处理任务被分配至后台线程。程序通过 HandlerThread 创建一个后台线程，专门用于图像分析任务，确保图像处理不会影响 UI 线程的响应。每当捕获到新的一帧图像时，图像处理函数会调用分析器对图像进行处理，图像被转换为 Bitmap 格式，并用于表情识别。如图 5-9 所示，使用摄像头的效果图展示了系统的运行状态。

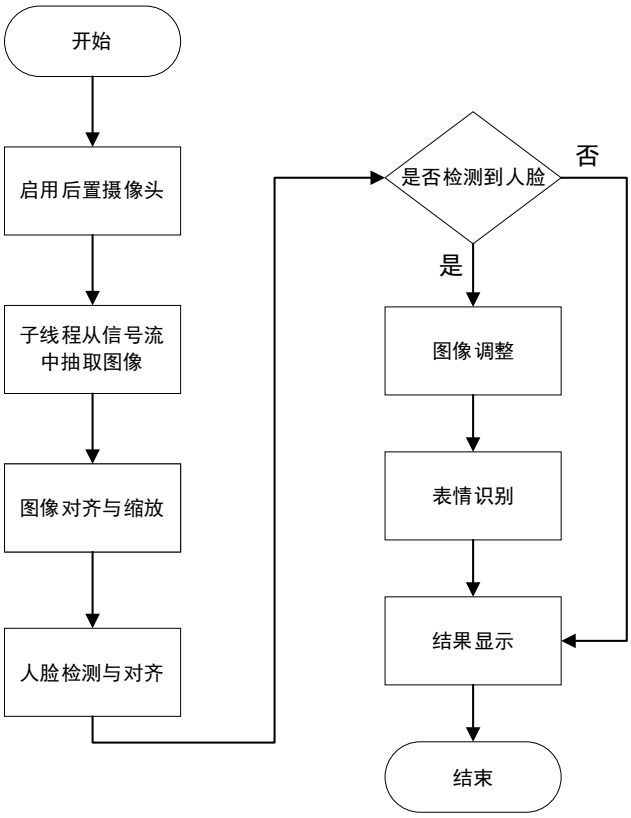
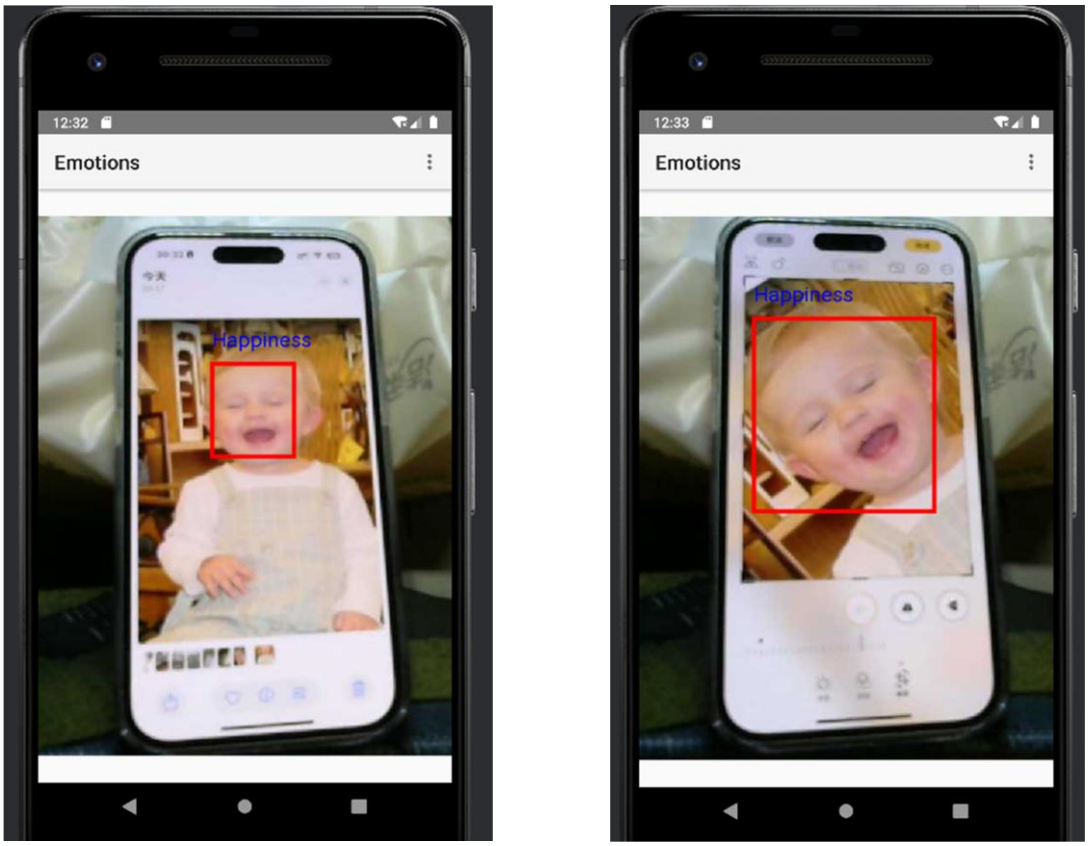


图 5-8 系统静态图像识别流程



（a） （b）

图 5-9 实时动态图像识别的效果图：（a）人脸图像识别；（b）有旋转角度的人脸图像识别

5.2.4 应用测试

为系统评估移动端表情识别应用的鲁棒性，本研究设计并实施了基于 Android 平台的仿真实验。考虑到真实场景数据采集的局限性，实验选用 AffectNet 数据集的测试子集作为基准数据。该数据集包含从互联网收集的八类基本表情样本，涵盖“开心”、“中性”、“惊奇”、“轻蔑”、“恐惧”、“悲伤”、“厌恶”和“愤怒”等类别，具有较好的样本多样性和场景代表性，能够较为全面地反映真实场景中的表情变化和分布特征。为控制数据质量，实验数据为从测试集中的随机选取的 400 张图像（每类 50 张），图 5-10 展示了部分样本图像，表 5-1 则详细记录了实验结果，包括各类表情的识别准确率。该实验有效模拟了实际表情识别应用场景，通过严格的实验流程和数据分析，为评估算法在复杂表情数据上的泛化能力提供了可靠的实验依据。

图 5-10 所抽取图像集的示例图像

表 5-1 测试结果

表情类别	正确数（张）	准确率（%）
开心	45	90.00
中性	40	80.00
悲伤	31	62.00
轻蔑	27	54.00
惊奇	41	82.00
恐惧	30	60.00
厌恶	26	52.00
愤怒	33	66.00
总计	273	68.25

5.3 本章小结

本章实现了面部表情识别模型在移动设备上的部署与应用。利用 PyTorch Mobile 框架，将训练好的 FER 模型成功移植到 Android 平台，并开发了配套的应用程序。本章详细描述了应用开发的技术路径、环境配置步骤及功能实现过程，并展示了实际运行效果。该应用程序支持两种处理模式：既能够分析本地存储的静态图像，也可以实时处理摄像头捕获的视频流。系统应用整合了人脸检测、关键点定位和表情分类等多个处理步骤，并通过图形界面实时显示分析结果。实验结果证实，该系统不仅验证了前述轻量级表情识别方法的有效性，也为深度学习模型在移动端的实际应用提供了实践参考。

第六章 总结与展望

6.1 本文总结

随着人工智能技术的不断进步，人机交互模式正在从认知智能向情感智能演进。面部表情识别网络的轻量化研究具有显著的理论意义和实际应用价值。通过开发高效的表情识别模型，可以使智能设备更准确地识别和理解人类情感状态，从而促进人机交互的自然化和智能化。然而，当前主流的深度学习表情识别技术虽然在准确度上表现优异，但对硬件的要求较高，内存占用大，实时性较差，这使得它们在低算力的设备上的应用受限。因此，如何在保证识别精度的同时，降低模型的计算复杂度和内存消耗，成为实现移动端和嵌入式设备上应用的关键。本文聚焦于轻量级面部表情识别算法及其工程应用，核心研究工作如下：

（1）通过引入基于门控剪枝的轻量级网络模型构建方法，本文优化了人脸表情识别模型的计算效率和性能。针对当前深度学习神经网络在资源受限环境下面临的计算量大、存储需求高的问题，提出了一种结合迁移学习与模型剪枝压缩的方案。该方案将模型构建与压缩过程分为三个阶段。首先，选取几种主流的轻量级神经网络模型进行人脸识别预训练，并通过性能评估筛选出最优的基础模型。接着，将该最优基础模型迁移至表情识别任务中，并在此阶段通过优化损失函数和调整训练策略以提升识别精度。最后，采用细粒度通道剪枝技术对迁移学习后的模型进行压缩，从而降低模型复杂度并提高计算效率。基于此，本文进一步提出了一种“惩罚-奖励-决断”三阶段的迭代剪枝框架，用于在剪枝过程中逐步恢复模型精度。实验结果表明，该方法在尽量减小精度损失的同时，能够有效实现模型压缩。

（2）通过引入混合精度通道量化方法，本文从硬件支持的角度出发，进一步地提升了表情识别网络的计算效率和存储效率。针对传统量化方法在处理网络通道敏感性差异方面的局限，本文提出了一种优化的混合精度通道量化方案。该方法基于通道敏感性，为每个通道的权重和激活值分配不同比特位数，从而实现模型的量化压缩。此外，本文还引入了强化学习中的“演员-评论家”网络模型，通过优化比特分配策略，进一步平衡量化过程中的精度与计算效率。实验结果表明，所提出的混合精度通道量化方法在多种模型和数据集上展现出显著的存储效率提升。在相同压缩比下，模型的精度损失较小，并

且优于一些现有的量化方法。

（3）设计并实现了一款基于 Android 平台的面部表情识别系应用。通过将桌面端训练的 Pytorch 模型转换为移动端支持的 PTL 格式，并集成 Pytorch Android Lite 推理框架，实现了模型在移动设备上的高效部署。系统采用 MTCNN 算法完成人脸区域定位，结合本文提出的轻量级表情分类模型进行特征分析。利用 Android Studio 开发环境，该系统具备静态图像分析和实时视频流处理双重功能，成功实现了移动端表情识别技术的工程化应用。该系统的开发为深度学习模型在移动终端的部署提供了可行的技术方案，对推动表情识别技术的实际应用具有参考价值。

6.2 研究展望

本文针对现有表情识别模型存在的计算复杂度高、存储需求大等问题，提出了相应的优化策略，并通过系统性实验验证了所提方法在提升模型效率和性能方面的有效性。然而，本文研究仍存在若干局限性需要进一步探讨。本节将深入分析当前方法的不足，并展望未来的研究重点与潜在改进方向。

（1）在人脸表情识别任务中，遮挡和姿态变化是影响识别效果的重要因素。在对实验数据集的观察过程中，面部表情图像常常受到诸如发型或装饰物等遮挡，这会干扰表情识别网络的准确性。此外，采集到的人脸姿态往往不符合标准，头部姿态的变化会显著影响人脸检测和对齐算法的性能，进而对表情识别结果造成较大干扰。因此，遮挡和姿态变化问题仍是人脸表情识别领域中的普遍挑战，所提出的方法亟需进一步改进以应对这些问题。

（2）随着智能设备的普及，尤其是嵌入式设备的广泛应用，部署表情识别模型到这些低功耗、高效能的设备上显得尤为重要。因此，在安卓设备上的模型部署可以进一步扩展到嵌入式设备上，为了更符合嵌入式设备的部署要求，需将 pt 模型转换为 tflite 模型。在嵌入式设备上部署深度学习模型时，tflite（TensorFlow Lite）作为一种专为移动和嵌入式设备优化的框架，其轻量化程度和硬件适配性尤为突出。tflite 具有小巧的文件尺寸、较低的计算复杂度和更低的内存占用，非常适合在资源有限的嵌入式环境中运行。为了进一步提升模型部署的灵活性和兼容性，可能需要将 pt 模型首先转换为 onnx 格式，再通过转换工具转换为 pb 文件格式，最后转为 tflite 文件。这一转换过程不仅能确保跨平台的兼容性，还能在不同的深度学习框架之间实现更高效的模型迁移和优化，但是如何无损进行模型转换还需要进一步研究。

（3）尽管本文通过“惩罚-奖励-决断”三阶段迭代剪枝框架和“演员-评论家”强化学习网络，实现了模型精度与模型大小之间的平衡，但目前的评判方法仍然依赖于主观裁定，未能充分考虑特定设备的硬件条件。在未来的研究中，结合具体设备的存储空间和计算能力来动态调整模型结构是必须的，因此，如何提出一种基于客观硬件资源约束的模型动态压缩方法，是一个值得深入探讨的研究方向。

（4）系统程序仍有进一步优化的空间。在人脸表情识别 APP 的测试和使用时发现，程序的整体耗时受到多种因素的影响，不仅需要考虑模型推理的时间，还应重视图像数据的预处理过程。事实上，原始数据的预处理步骤有时也会显著增加程序的运行时间。因此，未来的优化方向可以考虑在硬件设备上实现图像数据预处理的硬件加速，以进一步提升程序的运行效率。

参考文献

[1] Mehrabian A, Ferris S R. Inference of attitudes from nonverbal communication in two channels[J]. Journal of consulting psychology, 1967, 31(3): 248.

[2] Yildirim S, Chimeumanu M S, Rana Z A. The influence of micro-expressions on deception detection[J]. Multimedia Tools and Applications, 2023, 82(19): 29115-29133.

[3] Melinda M, Andriyani N A C, Nurdin Y, et al. Deep learning performance analysis for facial expression based autism spectrum disorder identification[J]. Radioelectronic and Computer Systems, 2024, 2024(2): 30-40.

[4] Talaat F M. Real-time facial emotion recognition system among children with autism based on deep learning and IoT[J]. Neural Computing and Applications, 2023, 35(17): 12717-12728.

[5] Valagkouti I A, Troussas C, Krouska A, et al. Emotion recognition in human–robot interaction using the NAO robot[J]. Computers, 2022, 11(5): 72.

[6] Ekman P, Friesen W V. Constants across cultures in the face and emotion[J]. Journal of personality and social psychology, 1971, 17(2): 124.

[7] Ekman P, Friesen W V. Facial action coding system[J]. Environmental Psychology & Nonverbal Behavior, 1978.

[8] 胡敏, 胡鹏远, 葛鹏, 等. 基于面部运动单元和时序注意力的视频表情识别方法[J]. 计算机辅助设计与图形学学报, 2023, 35(1): 108-117.

[9] Naveen P. Occlusion-aware facial expression recognition: A deep learning approach[J]. Multimedia Tools and Applications, 2024, 83(11): 32895-32921.

[10] 邵志文, 周勇, 谭鑫, 等. 基于深度学习的表情动作单元识别综述[J]. 电子学报, 2022, 50(8): 2003-2017.

[11] 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.

[12] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]//International conference on machine learning. PMLR, 2019: 6105-6114.

[13] Ma N, Zhang X, Zheng H T, et al. Shufflenet v2: Practical guidelines for efficient cnn

architecture design[C]//Proceedings of the European conference on computer vision (ECCV). 2018: 116-131.

[14] Barros P, Churamani N, Sciutti A. The facechannel: a fast and furious deep neural network for facial expression recognition[J]. SN Computer Science, 2020, 1(6): 321..

[15] Chen S, Liu Y, Gao X, et al. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices[C]//Chinese conference on biometric recognition. Cham: Springer International Publishing, 2018: 428-438.

[16] Yang D, Wang Y, Wei R, et al. An efficient multi-task learning CNN for driver attention monitoring[J]. Journal of Systems Architecture, 2024, 148: 103085.

[17] Ab Wahab M N, Nazir A, Ren A T Z, et al. Efficientnet-lite and hybrid CNN-KNN implementation for facial expression recognition on raspberry pi[J]. IEEE Access, 2021, 9: 134065-134080.

[18] Chowdary M K, Nguyen T N, Hemanth D J. Deep learning-based facial emotion recognition for human‐computer interaction applications[J]. Neural Computing and Applications, 2023, 35(32): 23311-23328.

[19] Li Z, Imai J, Kaneko M. Facial-component-based bag of words and phog descriptor for facial expression recognition[C]//2009 IEEE International Conference on Systems, Man and Cybernetics. IEEE, 2009: 1353-1358.

[20] Salmam F Z, Madani A, Kissi M. Facial expression recognition using decision trees[C]//2016 13th international conference on computer graphics, imaging and visualization (CGiV). IEEE, 2016: 125-130.

[21] Hsieh C C, Hsih M H, Jiang M K, et al. Effective semantic features for facial expressions recognition using SVM[J]. Multimedia tools and applications, 2016, 75: 6663-6682.

[22] Sharafeldeen A, Elsharkawy M, Khaled R, et al. Texture and shape analysis of diffusion‐weighted imaging for thyroid nodules classification using machine learning[J]. Medical Physics, 2022, 49(2): 988-999.

[23] Murugappan M, Mutawa A. Facial geometric feature extraction based emotional expression classification using machine learning algorithms[J]. Plos one, 2021, 16(2): e0247131.

[24] Gomes Ataide E J, Ponugoti N, Illanes A, et al. Thyroid nodule classification for

physician decision support using machine learning-evaluated geometric and morph ological features[J]. Sensors, 2020, 20(21): 6110.

[25] Lucey P, Cohn J F, Kanade T, et al. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression[C]//2010 ieee computer society conference on computer vision and pattern recognition-workshops. IEEE, 2010: 94-101.

[26] Yu Z, Zhang C. Image based static facial expression recognition with multiple deep network learning[C]//Proceedings of the 2015 ACM on international conference on multimodal interaction. 2015: 435-442.

[27] Goodfellow I J, Erhan D, Carrier P L, et al. Challenges in representation learning: A report on three machine learning contests[C]//Neural information processing: 20th international conference, ICONIP 2013, daegu, korea, november 3-7, 2013. Proceedings, Part III 20. Springer berlin heidelberg, 2013: 117-124.

[28] Li Y, Zeng J, Shan S, et al. Occlusion aware facial expression recognition using CNN with attention mechanism[J]. IEEE transactions on image processing, 2018, 28(5): 2439-2450.

[29] Kosti R, Alvarez J M, Recasens A, et al. Context based emotion recognition using emotic dataset[J]. IEEE transactions on pattern analysis and machine intelligence, 2019, 42(11): 2755-2766.

[30] Siqueira H, Magg S, Wermter S. Efficient facial feature learning with wide ensemble-based convolutional neural networks[C]//Proceedings of the AAAI conference on artificial intelligence. 2020, 34(04): 5800-5809.

[31] She J, Hu Y, Shi H, et al. Dive into ambiguity: Latent distribution mining and p-airwise uncertainty estimation for facial expression recognition[C]//Proceedings of t he IEEE/CVF conference on computer vision and pattern recognition. 2021: 6248-6257.

[32] Wen Z, Lin W, Wang T, et al. Distract your attention: Multi-head cross attention network for facial expression recognition[J]. Biomimetics, 2023, 8(2): 199.

[33] 周嘉雄.基于 3D 模型的人脸表情识别研究[D].吉林大学,2023.

[34] 张洁.基于残差网络的面部表情识别研究[D].西安工业大学,2023.

[35] 丰芳宇.基于残差网络的人脸表情识别方法研究[D].广西师范大学,2023.

[36] Yu J, Wei Z, Cai Z, et al. Exploring Facial Expression Recognition through Semi-Supervised Pre-training and Temporal Modeling[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024: 4880-4887.

[37] Tao H, Duan Q. Hierarchical attention network with progressive feature fusion for facial expression recognition[J]. Neural Networks, 2024, 170: 337-348.

[38] Gong W, Qian Y, Zhou W, et al. Enhanced spatial-temporal learning network for dynamic facial expression recognition[J]. Biomedical Signal Processing and Control, 2024, 88: 105316.

[39] Nan Y, Ju J, Hua Q, et al. A-MobileNet: An approach of facial expression recognition[J]. Alexandria Engineering Journal, 2022, 61(6): 4435-4444.

[40] 刘劲, 罗晓曙, 徐照兴. 权重推断与标签平滑的轻量级人脸表情识别[J]. Journal of Computer Engineering & Applications, 2024, 60(2).

[41] Martindez-Diaz Y, Luevano L S, Mendez-Vazquez H, et al. Shufflefacenet: A lightweight face architecture for efficient and highly-accurate face recognition[C]//Proceedings of the IEEE/CVF international conference on computer vision workshops. 2019: 0-0.

[42] Zou W, Zhang D, Lee D J. A new multi-feature fusion based convolutional neural network for facial expression recognition[J]. Applied Intelligence, 2022, 52(3): 2918-2929.

[43] 许大良.基于标签置信估计与知识蒸馏的表情识别算法研究[D].华中师范大学,2022.

[44] 姜慧明.基于生成对抗网络与知识蒸馏的人脸修复与表情识别[D].吉林大学,2020.

[45] 张宏丽, 白翔宇. 利用优化剪枝 GoogLeNet 的人脸表情识别方法[J]. Journal of Computer Engineering & Applications, 2021, 57(19).

[46] 唐玉梅,李丹杨,吴亚婷,等.基于粗糙集和集成剪枝的人脸表情识别方法[J].智能计算机与应用,2023,13(04):20-26.

[47] Koonce B, Koonce B. ResNet 50[J]. Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization, 2021: 63-72.

[48] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[49] Längkvist M, Karlsson L, Loutfi A. Inception-v4, inception-ResNet and the impact of residual connections on learning[J]. Pattern Recognit. Lett, 2014, 42(1): 11-24.

[50] Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural network[J]. Advances in neural information processing systems, 2015, 28.

[51] Liu Z, Mu H, Zhang X, et al. Metapruning: Meta learning for automatic neural network channel pruning[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 3296-3305.

[52] Zeng W, Xiong Y, Urtasun R. Network automatic pruning: Start nap and take a nap[J]. arXiv preprint arXiv:2101.06608, 2021.

[53] Han S, Mao H, Compression W J D D. Compressing deep neural networks with pruning[J]. Trained Quantization and Huffman Coding, 2016.

[54] Qin H, Gong R, Liu X, et al. Forward and backward information retention for accurate binary neural networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 2250-2259.

[55] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2704-2713.

[56] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.

[57] 高晗,田育龙,许封元,等.深度学习模型压缩与加速综述[J].软件学报,2021,32(01):68-92.

[58] Romero A, Ballas N, Kahou S E, et al. Fitnets: Hints for thin deep nets[J]. arXiv preprint arXiv:1412.6550, 2014.

[59] 吕君环.混合比特位宽卷积神经网络模型量化方法研究[D].北京交通大学,2022.

[60] Mollahosseini A, Hasani B, Mahoor M H. Affectnet: A database for facial expression, valence, and arousal computing in the wild[J]. IEEE Transactions on Affective Computing, 2017, 10(1): 18-31.

[61] Li S, Deng W, Du J P. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2852-2861.

[62] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image data -base[C]//2009 IEEE conference on computer vision and pattern recognition. Ieee,

2009: 248-255.

[63] Han D, Yun S, Heo B, et al. Rethinking channel dimensions for efficient model design[C]//Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. 2021: 732-741.

[64] Liu Z, Li J, Shen Z, et al. Learning efficient convolutional networks through network slimming[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2736-2744.

[65] Wang H, Qin C, Zhang Y, et al. Neural pruning via growing regularization[J]. arXiv preprint arXiv:2012.09243, 2020.

[66] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2704-2713.

[67] Shen S, Dong Z, Ye J, et al. Q-bert: Hessian based ultra low precision quantization of bert[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 8815-8821.

[68] Li Y, Gong R, Tan X, et al. Brecq: Pushing the limit of post-training quantization by block reconstruction[J]. arXiv preprint arXiv:2102.05426, 2021.

[69] Dong Z, Yao Z, Arfeen D, et al. Hawq-v2: Hessian aware trace-weighted quantization of neural networks[J]. Advances in neural information processing systems, 2020, 33: 18518-18529.

[70] Pollex R L, Hegele R A. Hutchinson–Gilford progeria syndrome[J]. Clinical genetics, 2004, 66(5): 375-381.

[71] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2704-2713.

[72] Wang K, Liu Z, Lin Y, et al. Haq: Hardware-aware automated quantization with mixed precision[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 8612-8620.

[73] 李双峰.TensorFlow Lite:端侧机器学习框架[J].计算机研究与发展,2020,57(09):1839-1853.

[74] Thakkar M. Beginning machine learning in ios: CoreML framework[M]. Apress, 2019.

[75] Gong R, Liu X, Jiang S, et al. Differentiable soft quantization: Bridging full-precision and low-bit neural networks[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2019: 4852-4861.

[76] Lv C, Niu C, Gu R, et al. Walle: An {End-to-End},{General-Purpose}, and {Large-Scale} Production System for {Device-Cloud} Collaborative Machine Learning[C]//16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22). 2022: 249-265.

[77] Deng Y. Deep learning on mobile devices: a review[J]. Mobile Multimedia/Image Processing, Security, and Applications 2019, 2019, 10993: 52-66.

[78] Wang R, Tian J, Jin C. Joint Face Detection and Alignment Using Focal Loss-Based Multi-task Convolutional Neural Networks[C]//Biometric Recognition: 14th Chinese Conference, CCBR 2019, Zhuzhou, China, October 12–13, 2019, Proceedings 14. Springer International Publishing, 2019: 266-273.