单位代码	10602
学　　号	2016011644
分 类 号	TP391
密　　级	公开

广西師範大學
GUANGXI NORMAL UNIVERSITY

硕士专业学位论文

基于计算机视觉的手势识别方法研究

Research on Gesture Recognition Method Based on Computer Vision

学　　院：电子工程学院

学位类别：工学硕士

领　　域：电子与通信工程

年　　级：2016 级

研 究 生：莫伟珑

指导教师：罗晓曙 教授

完成日期：2019 年 4 月

基于计算机视觉的手势识别方法研究

专业名称：电子与通信工程

申 请 人：莫伟珑

指导教师：罗晓曙 教授

论文答辩委员会

主席：

委员：

基于计算机视觉的手势识别方法研究

研究生姓名：莫伟珑　　导师姓名：罗晓曙 教授
专业：电子与通信工程　研究方向：图像处理 年级：2016 级

摘 要

随着现代技术的发展和计算机性能的飞速提升，人机交互逐渐变得自然化和多样化，因此，研究灵活、自然的人机控制方式具有重要的意义，尤其是依靠手势识别的控制方式。如果用计算机视觉的方式进行手势识别而不是用数据手套、遥控器这些硬件设备，将极大方便人们的生活需求。

计算机视觉通过使用算法实现对人类视觉的模拟仿真，可应用于控制、医学图像分析、目标检测等方面，亦可应用到人工智能、图像处理和科学计算等多个领域，最终的目是让计算机可以像人类那样通过视觉去看和理解现实的场景。让计算机视觉去理解人们手势传递的信息，可以减少接触式器件带来的束缚。进行手势识别是一个比较有挑战性的技术，传统的图像处理难以做到令人满意的效果，因为人手的变形具有不定性，还要考虑目标定位、运动状态、动作序列、环境干扰和被遮挡等多种问题的困扰，这也让手势的识别成为计算机视觉领域长期研究的重点和难点。

本文基于深度学习 Tensorflow 框架和 Ubuntu16.04 操作系统，通过两种方式实现了手势识别，仿真实验结果表明：提出的改进算法达到了良好的手势识别效果。本文完成的主要研究工作内容如下：

（1）对卷积神经网络的基础理论做了阐述，包括卷积神经网络的原理、卷积层和反卷积层的特征输出、目标函数的优化以及残差技术原理，为后续手势图像识别算法的研究和实现奠定了理论基础。

（2）提出了基于改进胶囊网络与算法的手势图像分割与识别方法，即通过手势分割、手势定位和手势识别的方式实现了手势识别。首先利用改进的胶囊网络算法，再结合 U-net 网络和残差技术，实现了 U 型残差胶囊分割网络，完成了对手势图像的分割；然后对矩阵胶囊网络进行结构优化，完成了对不同类别的手势图像的识别；最后通过数据集的训练和测试，得出了手势图像分割的性能指标、P-R 曲线、ROC 曲线、网络中间层特征可视化图、手势图像识别的混淆矩阵、手势图像识别 loss 曲线和识别准确率曲线。通过仿真结果表明，提出的改进模型达到了较好的识别效果。

（3）研究了基于 Tiny Yolo v3、Deep-sort 和 DenseNet 多模型与算法综合的方法实现手势识别。具体研究分析了 Tiny Yolo v3 检测算法、Deep-sort 目标跟踪算法和 DenseNet 算法，通过制作数据集进行模型的训练，首先训练好 Tiny Yolo v3、Deep-sort 和 DenseNet 三个模型，然后将几种模型和算法进行有机整合，实现了一种在复杂背景下对手势的检测、

追踪和姿态的识别方法，算法流程速度达到每秒 7~8 帧。

关键词：手势识别；胶囊网络；Tiny Yolo v3 检测；Deep-sort 目标跟踪；DenseNet 网络

Research on Gesture Recognition Method Based on Computer Vision

Master Candidate: Weilong Mo Supervisor: Prof. Xiaoshu Luo Grade: 2016
Major: Electronics and Communication Engineering
Research Area: Image Processing

Abstract

With the development of modern technology and the rapid improvement of computer performance, human-computer interaction has gradually become naturalized and diversified. Therefore, it is of great significance to study flexible and natural human-machine control methods, especially relying on gesture recognition. If you use computer vision to perform gesture recognition instead of using hardware devices such as data gloves and remote controls, it will greatly facilitate people's needs.

Computer vision can realize simulation of human vision by using algorithms. It can be applied to control, medical image analysis, target detection, etc. It can also be applied to many fields such as artificial intelligence, image processing and scientific computing. the ultimate goal is to make the computer can be like Humans see and understand realistic scenes through vision. Allowing computer vision to understand the information conveyed by people's gestures can reduce the constraints imposed by contact devices. Gesture recognition is a more challenging technique. Traditional image processing is difficult to achieve excellent results because the deformation of the human hand is uncertain, and the target positioning, motion state, motion sequence, environmental interference, and easily obscured by objects are also considered. The confusion of various problems, which makes the recognition of gestures become the focus and difficulty of long-term research in the field of computer vision.

Based on the deep learning Tensorflow framework and Ubuntu16.04 operating system, the gesture recognition is realized in two ways. The simulation results show that the proposed improved algorithm achieves good gesture recognition. The main research work completed in this paper is as follows:

(1) The basic theory of convolutional neural networks is expounded, including the principle of convolutional neural networks, the characteristic output of convolutional and deconvolution layers, the optimization of objective functions and the principle of residual techniques.It lays the theoretical foundation for the research and implementation of subsequent gesture image recognition algorithms.

(2) The gesture image segmentation and recognition method based on improved capsule network and algorithm is proposed. The gesture recognition is realized by gesture segmentation,

gesture positioning and gesture recognition. Firstly, the improved capsule network algorithm is combined with the U-net network and residual technology to realize the U-shaped residual capsule segmentation network, and the segmentation of the gesture image is completed. Then, the structure of the matrix capsule network is optimized, and the recognition of different types of gesture images is completed;Finally, through the training and testing of dataset, the performance index, P-R curve, ROC curve,the visualization of middle network layer feature , gesture image recognition confusion matrix, gesture image recognition loss curve and recognition accuracy curve of gesture image segmentation are obtained.The simulation results show that the proposed improved model achieves better recognition results.

(3)The method of multi-model and algorithm synthesis based on Tiny Yolo v3, Deep-sort and DenseNet is studied to realize gesture recognition. The specific research analyzes the Tiny Yolo v3 detection algorithm, the Deep-sort target tracking algorithm and the DenseNet algorithm. By making the data set and training the model, firstly train the three models Tiny Yolo v3, Deep-sort and DenseNet, then these three models and algorithm is integrated to realize a method for detecting, tracking and recognizing gestures in a complex background. The algorithm is up to 7~8 frames per second.

Keywords: gesture recognition; capsule network; Tiny Yolo v3 detection; Deep-sort multi-target tracking; DenseNet

目录

第 1 章 绪论

1.1 课题研究背景及意义

随着 21 世纪以来人工智能[1]的发展与大数据[2]时代的出现，计算机视觉[3]也得到了快速的发展。人与机器的交互变得更加频繁，人们不再满足传统的人机交互方式，对人机交互方式提出了愈来愈高的要求。人机交互[4]成为了人们生活中不可或缺的一部分，尤其是手势识别，由于其安装方便，成本低廉，使得人们在生活中可以用一种简单、直接的方式进行人机交流。

近几十年以来，手势识别技术得到了长足的发展。由于手势交互十分灵活，为了满足实际应用的需要，研究基于机器视觉的人机手势交流[5]方法具有重要的应用价值，将大大提高相关产品的智能化水平。经过多年的发展后，手势识别已经提出了很多方法。手势识别的手段包括有基于传感器的方法、基于无线信号的方法以及基于机器视觉的方法。其中基于传感器的方法是在智能设备内部安置了惯性传感器，如数据手套，虽然该方法在一定程度上弥补了基于视觉和语音在不理想环境下的不足，但是在应用中佩戴的机器部件过多，缺乏灵活性。基于无线信号的方法如 Wifi 信号感知[6]，利用无线感知模块[7]来提取有用的信息实现手势识别，但这种方式至今为止也没有成熟的技术方案。目前基于视觉的手势识别方法，比如结合图像处理方式利用肤色模型分割[8]出手势再结合卷积神经网络（CNN）进行识别是一个研究热点。

通过计算机视觉的方法进行手势识别已经成为一门新兴的技术。手势的分类和识别对于人类来说是一项容易的任务，然而对于计算机视觉来说，手势识别仍然存在着巨大的挑战。第一，同一种手势的复杂多变以及拍摄的角度不同，让机器识别更加困难；第二，室内外受各种光线的影响，会导致不清晰；第三，手可能会被另一些物体遮挡住；第四，在手的运动过程中，能做实时识别十分重要，特别是手语翻译，手势云台控制等需要进行一连串的动作识别，因此每一帧图像能够给系统做出的反应时间极短，这就要求系统能实时完成手势识别，并最终得出手势行为正确结果。第五，高精度的识别手段要依赖高成本的设施，从而让手势识别难以推广。手势识别领域涉及了云台控制、虚拟现实[9]和对聋哑人的手势手语的翻译[10]等，因此，在日常生活中寻找一种安全高效、灵活便捷的手势识别方法十分有必要，对于人们的生活和生产具有重要的意义。

1.2 国内外手势识别的研究现状

关于手势识别的研究，从上世纪 80 年代起就开始了。国外对手势识别的研究比较早，VPL 公司在 1987 年里，开发出了一款手势识别传感手套，当使用者的手把它戴上之后，能够做到实时在 VR（虚拟现实）系统进行交互[11]。例如，当使用者把手套戴上后，可以 VR 在中进行对目标的抓取，操作和移动等。机器通过安装在里面的传感器传输的信号进

行跟踪和识别出用户的手势姿态，从而能够执行相应的操作。

在 2006 年，Yoruk E 等对于大约 500 名受试者用平板扫描仪采集用户的右手图像，并计算手的旋转和平移，以及单独手指的图像轮廓中的豪斯多夫距离来进行手势分类[12]，实验结果取得了令人满意的结果。同年，Roy 等人成功地开发了用于身份验证或识别的手识别技术[13]，提出了一种结合颜色、纹理和形状信息的无接触生物识别方法。首先，分割集成了皮肤颜色组件和表单模型。然后，将手指的几何特征和被分析手掌的纹理进行卷积，使认证过程融合。实验结果表明，对 16 个志愿者进行测试，错误识别率低于 2%。

微软公司在 2010 年推出了设备 Kinect[14]，使得研究人员可以利用 Kinect 设备采集彩色图片和深度数据，并对手势进行分割和识别。同年，Hitoshi Hongo 等提出了一种多摄像头系统[15]，可以跟踪多个手，并聚焦于手势进行识别。立体摄像机通过设定的标准肤色方法检测手部。然后估计目标的距离，如果目标的大小不适合识别，跟踪摄像头将获取其缩放图像，最终通过四个方向的特征来识别面部和手势，但该方法容易受到如光线等因素影响，只能在特定的理想环境下才有较好的识别效果。

近些年来，深度学习技术越来越受到研究者的重视，由于其鲁棒性好，准确率高，因此逐渐成为手势识别的主流方法。Devineau G 等采用并行卷积的方法[16]对手骨关节位置序列进行处理，并对该模型在手势序列分类任务中的性能进行了研究。模型只使用手部骨骼数据，没有深度图像。实验对 SHRC 2017 三维形状检索竞赛的 DHG 数据集进行了测试，模型实现了 14 个手势类案例的 91.28%的分类准确度，28 个手势类案例的 84.35%的分类准确度，实验结果表明此方法有较强的鲁棒性，更加适应现实中的情况。

国内早期也采用数据手套的方式进行手势识别，吴江琴等人使用数据手套识对手势动作进行了采集，使用神经网络和决策树的方式进行训练，能够别出 25 种手势字母，但是这种基于数据手套的方法会有较高的硬件成本，而且人机交互并不自然，局限性比较大。

在 2002 年，张良国等人使用基于 Hausdorff 距离的手势技术[17]，该技术使用边缘特征像素点作为特征点，实现了多达 30 个字母手势进行识别，最终的识别率为 96.7%，但是不同的手势的角度、距离等带来的问题会导致手势识别精度不高。

在 2011 年，李文生基于粒子群优化神经网络实现了动态手势识别[18]，为了能提高手势学习的训练速度和手势图像识别准确率，通过定义动态手势模型，并提取动态手势特征向量作为神经网络的输入，并使用粒子群算法训练 BP 神经网络的权值，最终实现手势分类。结果表明，该方法比传统的 BP 神经网络，不但能大幅度提高网络的精度以及泛化能力，还能更好的对手势动作进行识别。

在 2014 年，刘耀杰等人基于骨架运算的实现了动态手势定位和识别[19]，采用 ViBe 算法对背景进行建模再结合肤色检测，将得到的结果再对手势目标区域进行检测。最后利用形态学骨架运算提取出图像特征，并根据计算出的图像特征的坐标形态推算出手势类别。尽管该方法取得良好的识别效果，但时间复杂度高。

在 2017 年，刘杨俊武提出一种基于关键帧和局部极值的动态手势特征提取算法[20]，

算出手势轮廓并确定指尖特征，最终将特征采用 DTW 算法实现动态手势识别，但是这种人工设计特征的方法在增加手势类别的情况下，识别率会大幅下降。

在 2018 年，杜新怡使用 Kinect 深度摄像头来获取深度图像[21]，然后使用 Canny 算子对分割后的图像进行边缘检测，并求出手势轮廓以及相应的重心矩，最后根据指尖坐标点识别出石头剪刀布，但这种方式只能做一些简单的识别，仍然不能满足日常生活的应用。吴晴使用 CNN 进行特征提取，并使用 SVM 进行分类[22]，对 Sebastien Marcel 手势数据集和 Jochen Triesch 手势数据集进行识别，分别达到了 98.7%和 98.6%的识别精度，该方法减少了模型的复杂度，取得了良好的效果，但还需要进一步改进。

1.3 主要研究工作

本文基于深度学习 Tensorflow 框架和 Ubuntu16.04 操作系统，研究两种方式实现了手势识别，并比较两者的性能优劣。本文完成的主要研究工作有以下几个方面：

（1）阐述手势识别在国内外研究发展历程、研究现状和研究的意义，并简述了本文的总体框架设计。

（2）对卷积神经网络的基础理论做了阐述，包括卷积神经网络的原理、目标函数的优化、卷积层和反卷积层的特征输出以及残差技术原理，为后续手势图像识别算法的研究和实现奠定了理论基础。

（3）设计了基于改进胶囊网络与算法的手势图像分割与识别方法，即通过手势分割、手势定位和手势识别的方式实现了手势识别。

（4）设计了基于 Tiny Yolo v3、Deep-sort 和 DenseNet 多模型与算法综合的手势识别方法，实现了一种在复杂背景下对手势的检测、追踪和姿态的识别，算法流程速度达到每秒 7~8 帧。

（5）对本文两种手势识别算法的优缺点做了比较和总结，并对未来的算法的改进和功能的进一步完善进行了展望。

1.4 论文的组织结构

本文一共分为 5 章，各个章节具体安排如下：

第 1 章阐述手势识别在国内外的研究发展历程、研究现状和研究的意义。

第 2 章阐述对卷积神经网络的基础理论，包括卷积神经网络的原理、卷积层和反卷积层的特征输出、目标函数的优化以及残差技术原理。

第 3 章研究基于改进胶囊网络的手势分割和改进矩阵胶囊网络的手势识别方法。实现整个视频流截帧、图像增强、手势图像分割、手势定位的计算和手势具体分类的流程。

第 4 章研究基于 Tiny Yolo v3、Deep-sort 和 DenseNet 多模型和算法整合的手势识别算法,通过制作自己的数据集进行模型的训练,实现整个算法流程。

第 5 章对论文的完成的工作进行分析总结，分析本文算法的优缺点，并对今后的进一

步研究工作做了展望。

第 2 章 卷积神经网络相关理论

卷积是图像处理常用的操作，而卷积神经网络是一种多层感知器，利用局部感受和权重共享的方式实现对待识别图像特征的提取，最初用于对二维图形图像的识别，已经成为当前图像识别、检测和分割等计算视觉领域的研究热点。由于其网络结构本身可以提取图像的多层次特征，在很多的物体识别任务上都取得了很好的效果。深层卷积神经网络为了识别二维图像，设计了多层感知器，可以从大数据中自动学习图像数据的分布规律，具有强大的学习和特征表达能力，从而可大大提升图像的识别准确率。

2.1 卷积神经网络的原理

卷积层是卷积神经网络[23]进行特征抽取的关键部分。卷积层的输出如公式 2.1 所示，式中的 x_j^{l-1} 表示第 $l-1$ 层的第 j 个特征图，$w_{i,j}^{l-1}$ 表示卷积核矩阵，符号 * 表示卷积操作，b_i^{l-1} 表示第 $l-1$ 层的偏置，$y(\bullet)$ 表示非线性的激活函数[24]。

$$x_i^l = y(\sum_{j \in M_i} x_j^{l-1} * w_{i,j}^{l-1} + b_i^{l-1}) \tag{2.1}$$

池化层[25]可用于降维、实现非线性和可以扩大感知野。池化层如公式 2.2 所示，式中的 β_j^l 和 b_j^l 分别为权值和偏置参数，$down(\bullet)$ 和 $y(\bullet)$ 分别为下采样函数和激活函数。

$$x_i^l = y(\beta_i^l down(x_i^{l-1}) + b_i^{l-1}) \tag{2.2}$$

全连接层原理如公式（2.3）所示，全连接层的每个神经元将和上一层的全部神经元进行全连接，其主要目的是维度变换，把高维的特征变成低维的样本标记。每个神经元可使用 Relu[26]、Sigmoid[27]或 Tanh 等激活函数，最终输出结果为分类结果。

$$x_i^l = y(\sum_{j}^{n} x_j^{l-1} w_{i,j}^{l-1} + b_i^{l-1}) \tag{2.3}$$

网络通过前向传播得到预测输出值后，求出损失函数，再通过反向传播[28]对网络的参数进行更新，修正误差。当网络训练完成，即可得到输入与输出之间的非线性映射关系。目标函数定义为公式（2.4），式中的右边第一项是误差函数，常用的误差函数有均方误差函数、交叉熵损失函数;公式第二项是以 λ 为权重的 L2 正则化，用于避免训练后会造成过拟合的问题。

$$J(\mathbb{W}, b) = \frac{1}{n}\sum_{i=1}^{N} J(W, b; x^i, y^i) + \frac{1}{2}\lambda \| W \|^2 \tag{2.4}$$

2.2 卷积层和反卷积层的特征输出

卷积层使用了卷积核对上层的特征图做了卷积运算，从而输入相应的特征图，特征图按公式 2.5 计算出相应的尺寸大小，具体如图 2.1 所示。

$$w_l = \frac{w_{l-1} + 2 \times padding - Kernel\_Size}{Stride} + 1 \quad (2.5)$$

图 2.1 中下面的灰色格子是输入大小为 4×4 的输入图像，边界填充 *padding* 为 0，卷积核 *Kernel _ Size* 为 3，滑动步长 *Stride* 为 1，经过卷积运算最终输出了大小为 2×2 的特征图。

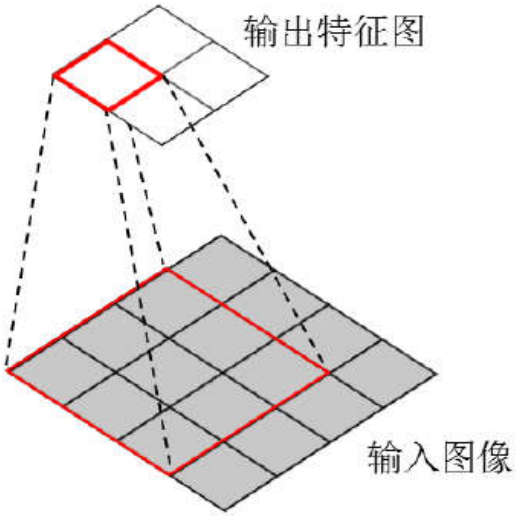


图 2.1 卷积过程

反卷积[29]也称为转置卷积，在某些领域，例如图像分割[30]，我们要通过用反卷积的方式实现将特征图还原到和原图一样的尺度大小。反卷积的方法使得图像可以放大，卷积核大小、补零操作及滑动步长决定了输出特征图的大小。图 2.2 给出了反卷积的过程，并按公式 2.6 取得相应的特征图大小。

$$w_l = Kernel\_Size\text{-}2 \times padding + Stride \times (w_{l-1} - 1) \quad (2.6)$$

图 2.2 中下面的灰色部分是大小为 2×2 的输入图像，其中 *padding* 为 0，*Kernel _ Size* 为 3， *Stride* 为 1，经过反卷积运算最终输出了上面部分大小为 4×4 的特征图。

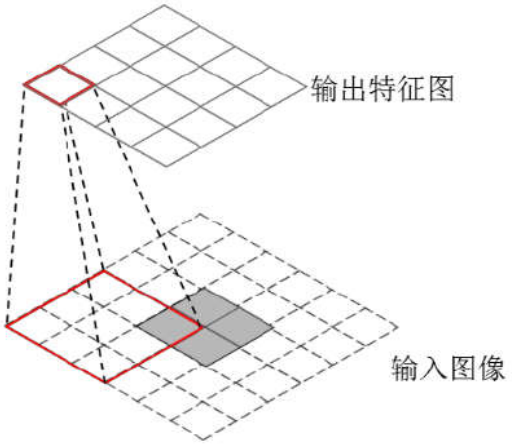


图 2.2 反卷积过程

2.3 目标函数的优化

为了让目标函数能快速的达到最小值，加快训练速度，反向传播需要选择合适的梯度下降法或动量优化法进行更新模型参数。常用的梯度下降法有批量梯度下降（Batch Gradient Descent，简称 BGD）、随机梯度下降法（Stochastic Gradient Descent，简称 SGD）和小匹量梯度下降（Mini-batch gradient descent，简称 MBGD）等。

梯度下降的过程为：先将给权重和偏移项按某种算法进行初始化（如高斯分布初始化、*xavier* 初始化和 *msra* 初始化等），再将权值不断优化使得损失值最小，设损失值为$J(\theta)$，θ为内部参数权重 W 和偏差 b，先通过反向传播得到参数更新公式 2.7，公式中的α为学习率（learning rate），θ_j为第j个特征。

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2.7}$$

再对公式（2.8）求偏导，过程如公式（2.8）所示：

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (\mathrm{h}_\theta(\mathrm{x}) - \mathrm{y})^2 \\ &= 2 \bullet \frac{1}{2} (\mathrm{h}_\theta(\mathrm{x}) - \mathrm{y}) \bullet \frac{\partial}{\partial \theta_j} (\mathrm{h}_\theta(\mathrm{x}) - \mathrm{y}) \\ &= (\mathrm{h}_\theta(\mathrm{x}) - \mathrm{y}) \bullet \frac{\partial}{\partial \theta_j} (\sum_{i=0}^{n} \theta_i x_i - y) \\ &= (\mathrm{h}_\theta(\mathrm{x}) - \mathrm{y}) x_j \end{aligned} \tag{2.8}$$

通过公式 2.7 和 2.8，如果计算第 i 个样本的反向传播，可以得到关于θ_j的公式（2.9）：

$$\theta_j = \theta_j + \alpha (\mathrm{y}^{(\mathrm{i})} - \mathrm{h}_\theta(\mathrm{x}^{(\mathrm{i})})) x_j^{(\mathrm{i})} \tag{2.9}$$

若有 m 个样本要进习，则可以使用 BGD 算法进行优化，BGD 算法一次性对整个数据集进行优化，即相当于一次性考虑完所有情形，再按最陡的方向更新，更新到最后为全局最优解，根据公式 2.9，可以得到如下 BGD 算法伪代码，代码中 *j=0,1,2,...n* 表示特征数。

repeat{

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} (\mathrm{y}^{(\mathrm{i})} - \mathrm{h}_\theta(\mathrm{x}^{(\mathrm{i})})) x_j^{(\mathrm{i})}$$

$$(\text{for every } j = 0, ..., n)$$

}

若样本数量 m 数值过大，例如可能为成千上万，一次性训练会导致更新非常慢，则可使用 *SGD* 算法，每次从一批样本中随机抽取一个样本去更新，这样训练速度也大大加快了，然而减少样本并不能实现全局更新，还会遭受噪声干扰导致准确率的下降，因此可以使用 *MBGD* 算法，该算法每次以一小批样本进行更新，MBGD 算法伪代码如下所示:

$$
\begin{aligned}
&repeat\{\\
&\quad for\ i = 1,11,21,31,...,\mathrm{m}\{\\
&\quad \theta_j = \theta_j + \alpha\frac{1}{10}\sum_{k=i}^{i+9}(\mathrm{y}^{(\mathrm{k})} - \mathrm{h}_\theta(\mathrm{x}^{(\mathrm{k})}))x_j^{(\mathrm{k})}\\
&\quad (\text{for every } j = 0,...,n)\\
&\quad \}\\
&\}
\end{aligned}
$$

上述算法的学习率α是预先人为设定的，而且各个参数的学习率也可能是不同的，没有进行自适应调整，这也大大影响了目标函数的训练速度。此外，学习还可能存在陷入局部最优以及鞍点的可能，因此可以考虑使用动量优化进行更新，动量优化基于梯度下降法进行优化，常用的动量优化算法例如有 Momentum，NAG（全称 Nesterov Accelerated Gradient）等。加入了自适应学习率的动量优化法会更加实用，比如有 AdaGrad（全称 Adaptive gradient algorithm）、RMSProp（全称 root mean square prop）和 Adam（全称 Adaptive Moment Estimation）算法等，Adam 算法对权值的更新如下面的伪代码所示：

$$
\begin{aligned}
&repeat\{\\
&\quad v_t = \beta_1 \times v_{t-1} - (1-\beta_1)\times \mathrm{g}_t\\
&\quad s_t = \beta_2 \times s_{t-1} - (1-\beta_2)\times \mathrm{g}_t^2\\
&\quad \Delta w_t = -\alpha\frac{v_t}{\sqrt{s_t+\varepsilon}}\times g_t\\
&\quad w_{t+1} = w_t + \Delta w_t\\
&\quad (\text{for every } w^j, j = 0,...,n)\\
&\}
\end{aligned}
$$

代码中的α为学习率，g_t为在 t 时刻的权值w^j的梯度，v_t为在 t 时刻权值w^j的指数平均，s_t为在t时刻权值w^j的梯度平方的指数平均，公式中的超参数β_1和β_2一般分别取 0.9 和 0.99，参数ε一般设置为 1e-10。

2.4 残差技术原理

卷积神经网络的深层传播过程会带来导致梯度消失[31]的退化问题，由于参数初始化会比较靠近 0，当多个权值连续相乘，最终那些靠前的层梯度会变得很小，因此学习也停滞了。图 2.3 的残差块[32]（Residual Block）通过加入一个抄近道（shortcut connection），把残差块的输入直接加到残差块的输出，图中残差块仅为一种最基本的形式，还有很多其它的方式。

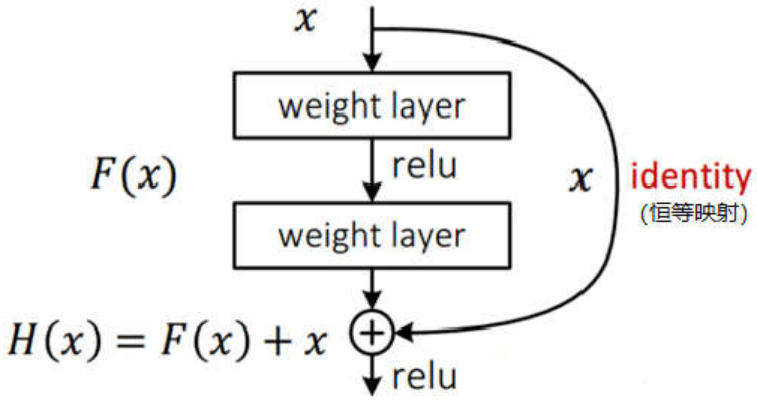


图 2.3 残差块结构

公式（2.10~2.13）给出了图 2.3 的残差块结构原理。式（2.10）的 x_{l+1} 为期望的下一层输出层，网络只要学习 x_{l+1} 和 x_l 之间的变化大小 $F(\mathrm{x}_l, \mathrm{w}_l)$ 即可，那些固定不动的东西不再需要学习，使得学习变得更加简单。

$$\mathrm{x}_{l+1} = x_l + F(\mathrm{x}_l, \mathrm{w}_l) \tag{2.10}$$

式子（2.11）给出两个残差块的连接。

$$\mathrm{x}_{l+2} = x_{l+1} + F(\mathrm{x}_{l+1}, \mathrm{w}_{l+1}) = x_l + F(\mathrm{x}_l, \mathrm{w}_l) + F(\mathrm{x}_{l+1}, \mathrm{w}_{l+1}) \tag{2.11}$$

式子（2.12）给出多个残差块的串联，从式子可以看出多个残差块串联后的网络的学习变成了加法，从而能大大加深网络结构的深度。

$$\mathrm{x}_L = x_l + \sum_{l}^{L-1} F(\mathrm{x}_i, \mathrm{w}_i) \tag{2.12}$$

式子（2.13）为残差网络的反向传播公式，由于公式中有常数 1 的存在，不会导致深层网络梯度消失，优化也更加容易。

$$\frac{\partial C}{\partial x_l} = \frac{\partial C}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial C}{\partial x_L}(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} F(\mathrm{x}_i, \mathrm{w}_i)) \tag{2.13}$$

2.5 本章小结

本章对卷积神经网络的基础理论做了阐述，包括卷积神经网络的原理、卷积层和反卷积层的特征输出、目标函数的优化以及残差技术原理，为后续手势图像识别算法的研究和实现奠定了理论基础。

第 3 章　基于改进胶囊网络与算法的手势图像分割与识别方法

已有的研究结果表明，胶囊网络[33]进行不同视角的手势图像分割和识别时比 CNN 更加有优势。在手势图像分割方面，使用现有胶囊网络的动态路由算法难以进行手势图像较深层次的特征提取，从而造成无法训练或训练效果不理想。在手势图像识别方面，现有的矩阵胶囊网络[34]收敛较慢，使用单一尺度通道造成识别率不高。直接使用 CNN 算法进行分割和识别参数量极大，这又增大了硬件开销，降低了手势图像识别率。

所以，本章提出了一种基于改进胶囊网络与算法的手势图像分割与识别方法，通过改进胶囊网络算法，并融合了 U-net[35]和残差技术，搭建了 U 型残差胶囊分割网络进行手势图像分割，解决了现有的矩阵胶囊网络收敛速度较慢，使用单一尺度通道造成识别率不高的问题，同时又避免了直接使用 CNN 算法进行手势图像分割和识别时参数量极大，造成极大增大硬件开销的技术问题。

3.1 实验环境

3.1.1 Tensorflow 框架

本章的仿真实验所使用的平台是在谷歌公司开发的机器学习框架 Tensorflow[36]完成的。Tensorflow 可以灵活、自由的搭建各种各样的卷积神经网络结构，在 GitHub 是上最受欢迎的机器学习库之一，该框架最开始是 Google Brain 小组开发的，其特点如下：

（1）Tensorflow 采用数据流图进行数值计算，不但支持单机模式，还支持分布式计算[37]，能充分利用硬件资源。

（2）Tensorflow 能实现卷积神经网络，不但能实现 caffe[38]不能实现的 RNN[39]以及 LSTM[40]等，还能实现深度学习以外的机器学习算法。

（3）拥有大量活跃的社区，国内有 Caicloud（才云）开发的 Tensorflow 中文社区。

（4）Tensorflow 提供了大量接口给开发者使用，可以使用 c++、python、java、go、c# 等编程语言在 Windows 或 Linux 环境下进行开发。相对于 caffe 框架而言，Tensorflow 能够非常直观的观察到计算的结果，提供了便捷的调试机制。

3.1.2 实验开发环境配置

本仿真实验中所使用的计算机配置包括有双 E5-2637 v4 CPU，64 位操作系统 Ubuntu 16.04，同时还使用了 GTX1080Ti 显卡、32GB 内存来加速训练。

此外，实验选用了 Python3.7 编程语言进行了开发，并在集成开发编译器 Pycharm 中实现所有功能，开发中使用了 opencv-pyhon-3.4.3、scikit-image-0.14 和 pillow-5.4 等开发库去处理图像，以及使用了 matplotlib-3.0、numpy-1.15、scikit-learn-0.19 和 pandas-0.23 等开发库分析和处理数据。

3.2 胶囊网络

3.2.1 胶囊网络原理

胶囊卷积层进行动态路由之前要按下面的公式（3.1）用输入特征乘以权重做变换，式中输入胶囊u_i乘以权值w_{ij}得$\hat{u}_{i|j}$，其中权值w_{ij}表示的是第 i 个特征属于第 j 个类别的权重，该权值经过训练使得可以遇到不同角度的手势也能够识别出来。

$$\hat{u}_{i|j} = w_{ij}u_i \tag{3.1}$$

动态路由公式为：

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ij})} \tag{3.2}$$

$$s_j = \sum_i c_{ij}\hat{u}_{j|i} \tag{3.3}$$

$$v_j = squash(s_j) = \frac{\| s_j \|^2}{1+\| s_j \|^2}\frac{s_j}{\| s_j \|} \tag{3.4}$$

$$b_{ij} = b_{ij} + \hat{u}_{i|j} \bullet v_j \tag{3.5}$$

其中，c_{ij}为动态路由耦合系数（即概率向量），b_{ij}初始化为 0，s_j是特征$\hat{u}_{i|j}$所对应的概率向量的加权之和；将公式（3.1）带入动态路由公式（3.2）到（3.5），通过反复迭代三次，就能实现输入是输出的聚类[41]结果。

3.2.2 胶囊网络的改进

针对原始胶囊网络用于图像分割时存在着难以正常在深层进行网络训练的问题，提出一种改进压缩函数 Squash 算法，改进后的压缩函数能够自适应调整激活值的数量以及防止梯度消失或爆炸，使得 U 型残差胶囊网络分割模型能够在深层网络进行正常训练，从而能有效正确输出手势分割图像。

当计算梯度运算遇到$log(\eta)$或除法运算时这些情形时，在η过小（趋于 0）的情况下导致数值无限大，从而导致程序训练时 loss 值无法收敛，网络将难以训练。公式（3.4）的$||s_j||^2$通常极小（幅值在 1e-20 到 1e-42 之间），从而导致激活值v_j也极小，当网络加深，迭代多次后v_j往往为 0 无法训练，改进后的压缩函数为公式（3.6），能得够调整v_j的数量级，从而能让模型正常训练。式中的atom为胶囊向量的长度，式子的$\frac{0.1}{\text{atom}}$能有效提高分割精度。

$$v_j = squash(s_j) = \frac{\| s_j \|^2}{\sqrt{|s_j|}+\| s_j \|^2}\frac{s_j}{\| s_j \|} + \frac{0.1}{\text{atom}} \tag{3.6}$$

3.2.3 残差胶囊模块

残差胶囊模块目的是为了加深网络训练，该结构上由两块胶囊卷积层组成，先把输入进

行批标准化，再输入到两个胶囊卷积层，第二层胶囊卷积层输出后再进行批标准化，两路输出相加再输出结果。

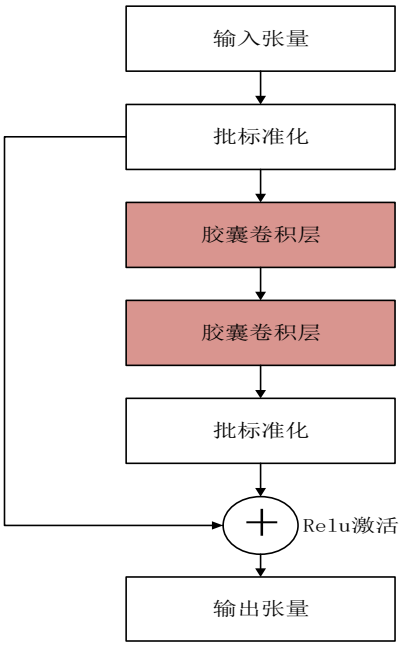


图 3.1 残差胶囊模块结构图

3.3 手势图像分割与识别方法整体流程图

根据前面有关的胶囊网络的基本理论概述，本章提出的改进胶囊网络与算法的手势图像分割与识别方法的整体流程图和步骤如下：

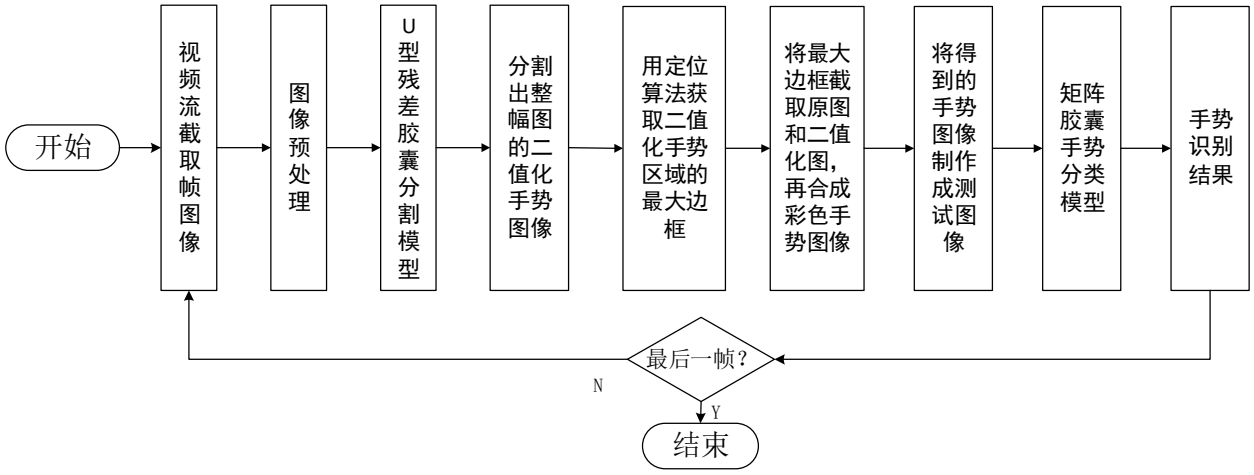


图 3.2 算法整体流程图

综上所述算法，进行图 3.2 的算法流程图实现手势识别，具体步骤如下：

步骤 1：拍摄和收集复杂背景下的手势图像，对所有图像的手势轮廓进行人工标注并生成标签图，再将原图和标签图进行图像增强处理；

步骤 2：用经过图像增强后的图像对 U 型残差胶囊网络进行训练，把复杂背景下的手势图像输入训练好后的 U 型残差胶囊网络分割出二值化手势图像；

步骤 3：将步骤 2 中分割出来的二值化手势图像经过图像定位得到矩形包围框，把包围框所对应的原图和分割图相乘最终得到所需要的手势图像；

步骤 4：利用不同手势形状的手势图像训练改进后的矩阵胶囊网络，输出训练好的模型，将步骤 3 得到的手势图像输入改进后矩阵胶囊网络模型，利用该模型分类出每种不同的手势，最终实现手势图像的识别。

3.4 数据集的使用

3.4.1 GTEA 数据集介绍

公共数据集 GTEA（Georgia Tech Egocentric Activity Datasets）为佐治亚理工学院为日常活动拍摄的数据集，该数据集包含 7 种类型的日常活动，每种活动包括 4 个不同的主题，在活动中把摄像机安装在被摄者戴的一顶帽子上进行拍摄。活动有：美式早餐、披萨、点心、希腊沙拉、意面沙拉、火鸡三明治和芝士汉堡。对每个活动都使用 ELAN 软件来图像标注，例如准备食物的任务，如做披萨，拍摄是一个比较短的时间片段，如在披萨饼皮上放酱汁，青椒切丁，洗蘑菇等。数据集总共有 1115 张图片，取 90%的图片用做训练集和校验集，10%用于做最终测试。

3.4.2 GTEA 数据集分割效果

图 3.3 和图 3.4 给出了从 GTEA 数据集抽取两张图片测试出的分割效果。

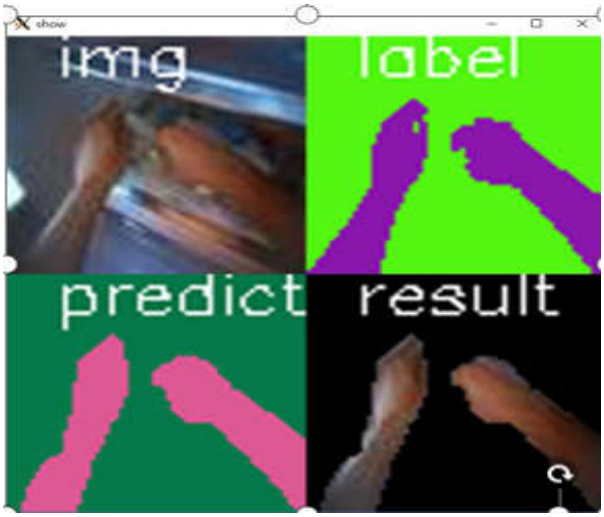


图 3.3 在阴暗环境下的最终分割效果

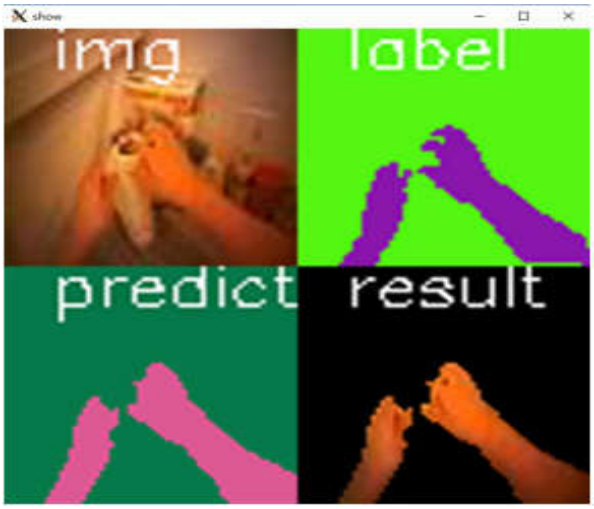


图 3.4 在和肤色颜色接近的昏黄环境下的最终分割效果

从图 3.3 和图 3.4 可以看出，本章模型能在这两种极端的环境下把手势正确的分割出来，做到了传统基于肤色分割算法无法达到的效果。

3.4.3 数据集的制作和图像增强

数据集的制作是用像机拍取大量室内外不同复杂场景下的手势图像，用 labelme 软件[45]对所有图像的手势轮廓进行人工标注，如图 3.5 所示，图中对手势轮廓进行了人工标注。

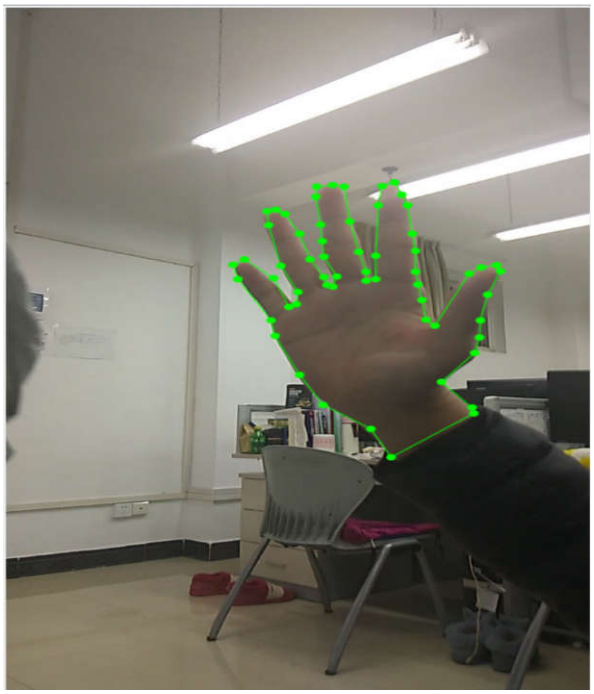


图 3.5 手势轮廓的标注

人工标注完成之后批量生成对应的二值化标签图，具体如图 3.6 所示。

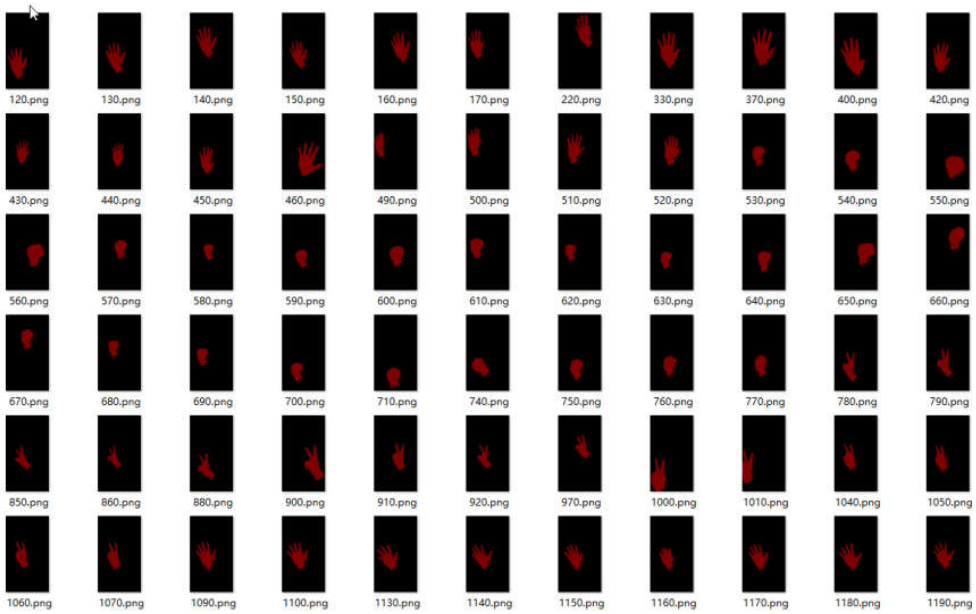


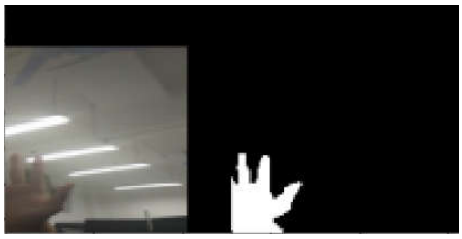
图 3.6 标注后输出的二值化图片

把标注后的图像在训练过程中进行图像增强，从而加强模型的泛化能力，图像增强效果如图 3.7 的(a)-(f)所示，图像增强需要同时对原图和标签图做同样的增强操作，图中分别进行了随机概率的旋转、翻转、仿射变换[46]、位移、缩放和椒盐噪声，训练过程会随机选择一种或多种图片增强的组合方式，每种增强方式的变化程度也是随机的。

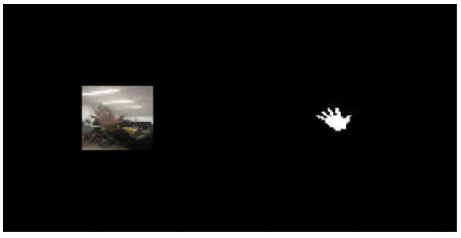
(a)随机旋转角度

(b)随机翻转

(c)随机仿射变换



(d)随机位移



(e)随机缩放

(f)随机椒盐噪声

图 3.7 图像增强效果

3.5 手势图像分割网络

3.5.1 U 型分割网络原理

U-net 用下采样和上采样的网络结构来实现图像分割。图 3.8 是经典的 U-net 结构，下采样用来逐渐展现环境信息，提取深层次特征，U-net 右侧结合下采样各层信息和上采样的输入信息来防止信息的丢失。上采样通过反卷积，由小尺寸变换到大尺寸，从而还原细节信息，并且逐步恢复图像精度，最终实现端到端的图像分割。

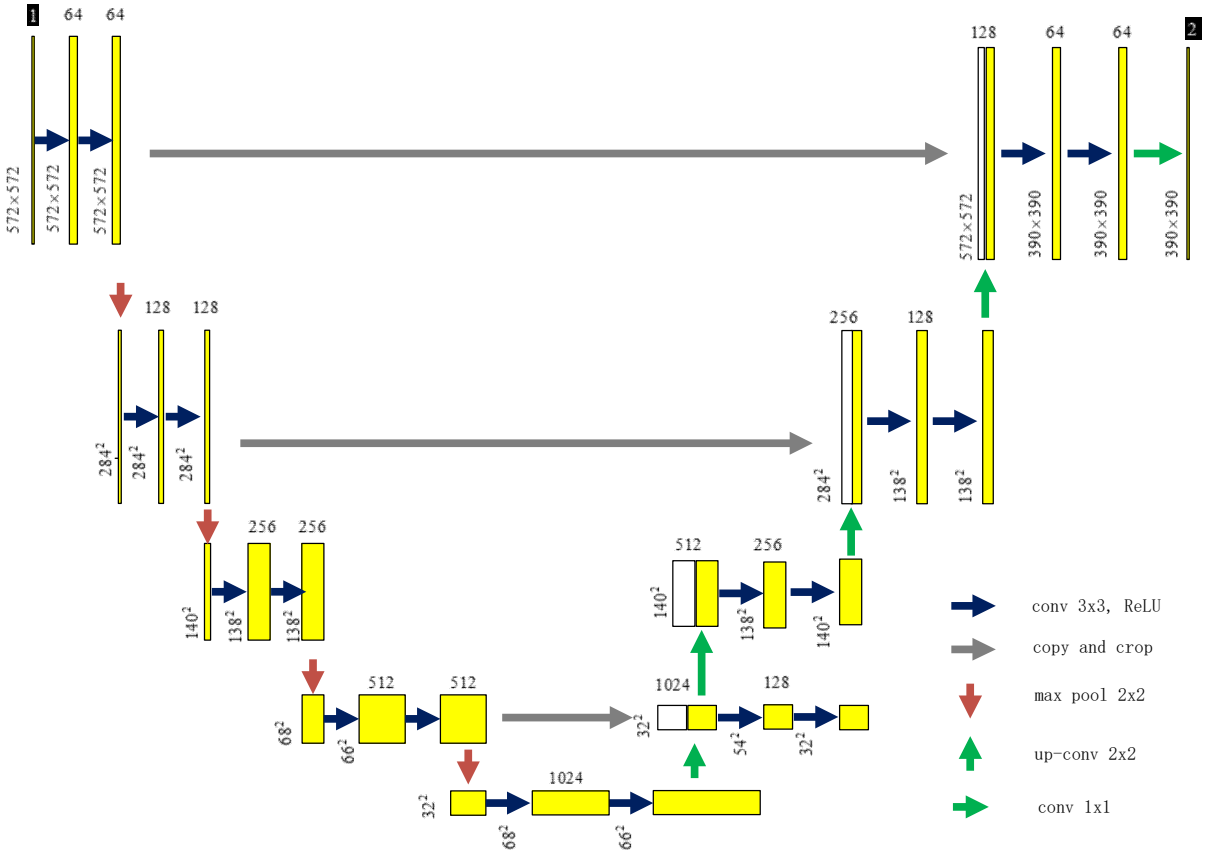


图 3.8 经典 U-net 模型

3.5.2 U 型残差胶囊分割网络

U 型残差胶囊网络分割模型结合了 U-net、深度残差技术和胶囊网络。把残差技术、胶囊网络做成一个残差胶囊结构模块，将学习转换成一个残差函数，随着网络深度的增加，能提取手势图像更丰富的深层次特征以及加快模型训练的收敛速度。

在图中 3.9，U 型深度残差胶囊分割网络模型主要由胶囊卷积层（CapsuleConv）、胶囊反卷积层（CapsuleDeConv）和残差胶囊块（Res_cap_block）组成。图中左侧部分使用胶囊卷积层和残差胶囊块对一张 128×128×3 的图像进行提取深层特征，下采样过程中特征层的大小变化依次为 128×128、64×64 和 32×32。网络最下面使用两个残差胶囊块作为中间层。右侧部分使用胶囊反卷积层进行上采样（放大图像），并把 U 型左侧提取的特征直接跳层拼接到右侧再进行提取特征，特征层的大小变化依次为 32×32、64×64 和 128×128，最终输出端

还原回和原图像大小一致的分辨率。网络最后的输出是对胶囊卷积层 L14_胶囊卷积层输出的 16 个通道取 F-范数（即对应元素的平方和再开方），得出一张 128×128 的单通道图片作为手势分割图像。

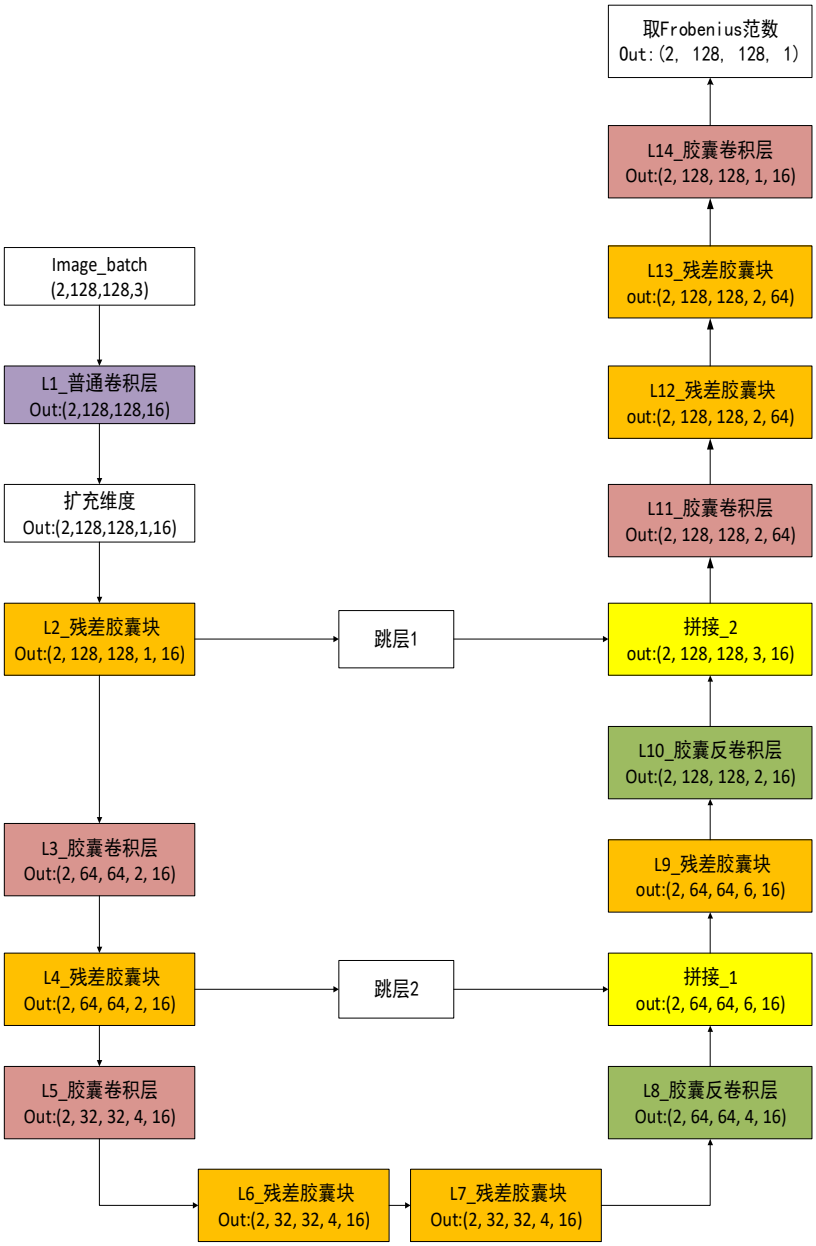


图 3.9 U 型残差胶囊分割网络结构图

U 型残差胶囊分割网络的训练目标函数使用三个 loss 函数组合进行训练，将网络输出的预测值和真实输入到 Loss 函数中，目标函数的如公式（3.7）所示：

$$\text{Loss} = Dice\_loss + \text{Binary\_cross\_entropy\_loss} + \alpha \times Focal\_loss \tag{3.7}$$

式中的 Loss 由 Dice_loss[42]、Binary_cross_entropy_loss[43]和 Focal_loss[44]三种 loss 组合而成，多 loss 组合能有效提升训练效果。

Binary_cross_entropy（二元交叉熵）的实现如公式（3.8）和（3.9）所示，式（3.9）中 $y \in \{0,1\}$。该式子专用于二分类交叉熵，当有两类时，为 0 或者 1，预测值输出先用 sigmod 做预处理，

将预测结果限定在 0~1 之间，

$$\hat{y}=\sigma(\hat{y})=\frac{1}{1+e^{\hat{y}}} \tag{3.8}$$

$$L_{logistic(\hat{y},y)}=-y log\hat{y}-(1-y)log(1-\hat{y}) \tag{3.9}$$

（3.10）式中 Dice loss 中的 A 和 B 分别为标签图和网络输出的预测图，该 loss 计算了 A 和 B 的相似度，当无限逼近相似时，dice_coef 的值为 1。

$$Dice\_loss=1-dice\_coef=1-2|A\cap B|/(|A|+|B|) \tag{3.10}$$

为了有效组合这三个 loss，需要将三个 loss 缩放到一致的数量级才能训练，因此这里加入放缩因子α缩小 Focal_loss 的大小。（3.11）式中的 Focal_loss 可以用于难以分类的样本，当训练图像的背景占比面积较大，而手势占比面积较小时，会导致负样本 loss 占据主导。γ值一般取值为 2，β取 0.25，从而能够调节正负样本的平衡。

$$Focal\_loss=\begin{cases}-\beta\left(1-y'\right)^{\gamma}\log y' & y=1\\ -(1-\beta)y'^{\gamma}\log\left(1-y'\right) & y=0\end{cases} \tag{3.11}$$

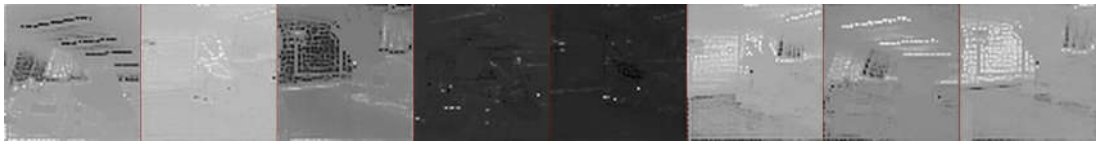
经过不断训练迭代更新胶囊分割网络的权重，直到 Loss 函数收敛，输出 U 型残差胶囊分割网络模型，再利用该模型进行手势图像的分割。

3.5.3 U 型残差胶囊分割网络中间可视化

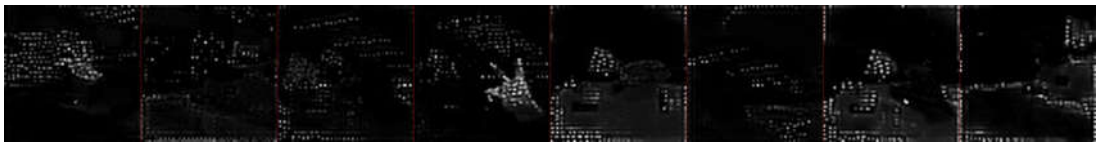
为了深入了解中间层的学习变化过程、训练效果以及方便网络的调参和修改网络结构，图 3.10 的(a)-(i)给出了网络经过 110 个 epoch 训练之后的可视化效果。

（a）网络 L1_普通卷积层

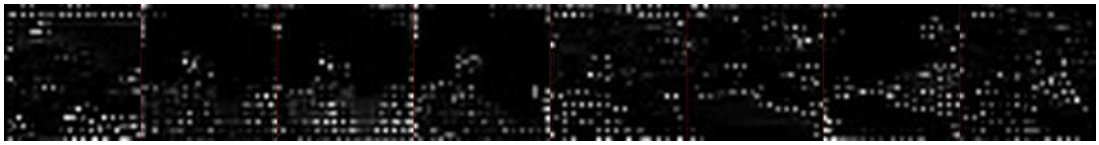
（b）网络 L2_残差胶囊块



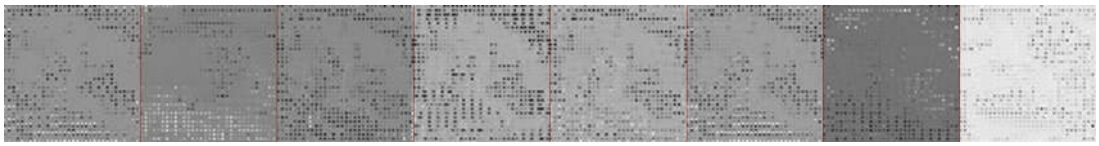
（c）网络 L3_胶囊卷积层



（d）网络 L4_残差胶囊块



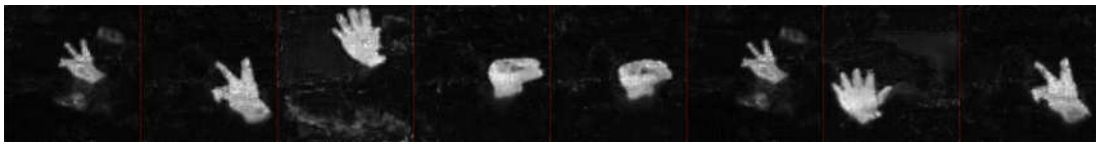
（e）网络 L6_残差胶囊块



（f）网络 L8_胶囊反卷积层

（g）网络 L11_胶囊卷积层

（h）网络 L14_胶囊卷积层



（i）网络最终取 Frobenius 范数输出层

图 3.10 中间可视化效果图

图 3.10（a）为网路结构的第一层，抽取了低级的特征；图 3.10（b）抽取了颜色显著的物体，并开始分离各种物体；图 3.10（c）开始抽取物体的边缘特征。图 3.10（d-e）网络为深层网络，学习到的内容为各个物件的细小纹理特征，表达也开始变得越来越抽象，特征描述更加细腻；图 3.10（f）作为网络的上采样结构，开始逐渐恢复图像精度以及还原要分割的手势；图 3.10（g）胶囊卷积层进行进一步还原图像精度；图 3.10（h）和（i）为网路最终输出的手势分割图像，说明网络的训练是正常的，每个像素点范围为 0~1 的概率，最终对最后一层取阈值为 0.5 得出手势二值化分割图像。

3.5.4 手势分割性能指标比较

经过使用自己制作的数据集进行训练和测试，下表给出了改进后的 u-res-cap-net 和其他分割网络的性能指标。

表 3.1 改进后的 U-res-cap-net 和其他算法的比较

Model（模型）	Auc_Roc（Roc 曲线面积值）	Auc_P-R（P-R 曲线面积值）	Specific（特异性）	Sensitivity（灵敏性）	Precision（精确率）	Jacard (杰卡德系数)	F1-Measure（F 值）	Parameters（参数量）
U-net	0.960	0.895	0.985	0.936	0.850	0.981	0.889	34,519,363
U-res-net	0.942	0.925	0.996	0.888	0.953	0.988	0.917	5,080,931
U-res-cap-net	0.960	0.940	0.995	0.925	0.949	0.990	0.936	1,100,738

公式（3.12）的 M、N 分别为正样本的数目和负样本的数目，对当前的分类算法依次增加阈值 threshold，求得各个 Score 值，再对 Score 值按照从大到小排序后得到公式（3.12）右项的 $Rank_i$ 值（最大的 Score 值对应 $Rank_n$），从而求出指标 Auc_Roc 和 Auc_P-R 的值。表中的指标先以 threshold（阈值）为 0.5 求出混淆矩阵的 TP（True Positive，真正例）、FP（False Positive，假正例）、FN（False Negative，假负例）和 TN（True Negative，真负例），再按公式（3.13）到（3.17）求出。

$$AUC = \frac{\sum_{i \in positive\_class} Rank_i - 0.5M \bullet (1+M)}{M \bullet \mathrm{N}} \tag{3.12}$$

$$Specificity = \frac{TN}{TN + FP} \tag{3.13}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{3.14}$$

$$\mathrm{Precision} = \frac{TP}{TP + FP} \tag{3.15}$$

$$J(\mathrm{A},\mathrm{B}) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{3.16}$$

$$F1 = \frac{2 \bullet \mathrm{Precision} \bullet \mathrm{Re}\, call}{\mathrm{Precision} + \mathrm{Re}\, call} \tag{3.17}$$

从表 3.1 中可以看出，指标 Auc_roc 为 ROC 曲线的下面积，本章达到了 0.960，指标 Auc_P-R 为 P-R（Precision-Recall）曲线的下面积，本章达到了 0.940，说明本模型对手势图像分割的性能非常好；指标 Specific 反应分割出来的图像中是背景的能力，Specific 值越低，则分割出来的图像越多斑点被当成手势图像；指标 Sensitivity 和召回率 Recall 一样，反应了分割出来的图像像素当中，有多少像素是属于手势图像像素的能力，Sensitivity 值越高，分割出来的手势图像越完整；指标 Jacard 作为衡量分割精度的一种，反应了分割出来的手势图像和标签图像的相似度，Jacard 值越高越近似标签图像；指标 F1 值同时衡量了 Precision 和 Recall 的性能，本章算法的 F1 值跟其他算法比较更好，同时本章模型的参数量比另外两种模

型的参数量要少，更加适用于硬件资源紧张的嵌入式设备。

在图 3.11 中给出了本章算法改进后的 u-res-cap-net 手势分割输出效果图，图中左边的图为原图，中间的图为标签图，右边的图为模型输出的二值化手势图。

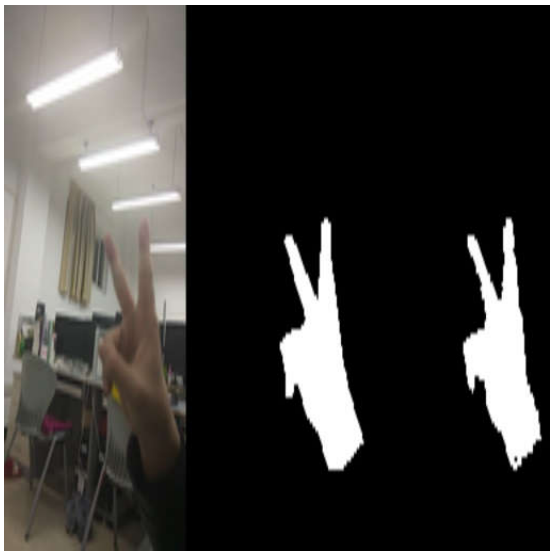


图 3.11 手势分割输出效果

为了测试算法的好坏，图 3.12 给出了评估模型的 P-R 曲线，该曲线把 Recall 作为 x 轴，该值表示在预测正确的正例中占实际所有正例数的比例；曲线把 Precision 作为 y 轴，该值表示在预测正确的正例中占预测为正的数量的比例。P-R 曲线是右上越凸越好，图中 AUC 表示曲线下的面积占正方形格子面积的比例，AUC 值为 0.9471。

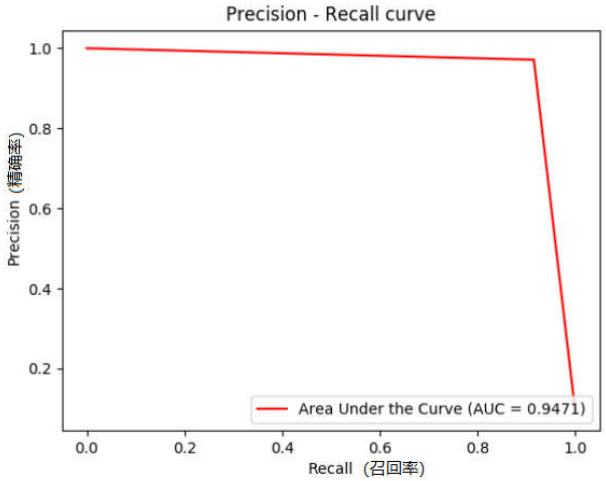


图 3.12 精确率-召回率曲线

图 3.13 给出了 ROC 曲线，该曲线把 Specificity（FPR，负责负例）作为 x 轴，该值越大则说明在预测为正的样本里实际负例的数量也就越多；曲线把 Sensitivity（TPR，负责正例）作为 y 轴，该值越大则说明预测为正的样本实际正例的数量也越多。曲线上的每一个点与一个阈值一一对应，例如当阈值为 0.4 时，可以得到一对（FPR 和 TPR）值，从而在平面上取得相应的坐标。通过曲线求出 AUC 值，该值越大，说明分割效果越好。若 AUC 值小于或等

于 0.5 则属于随机猜想，图中的 AUC 为 0.9760，该 AUC 值较高，模型已经达到的比较好效果。

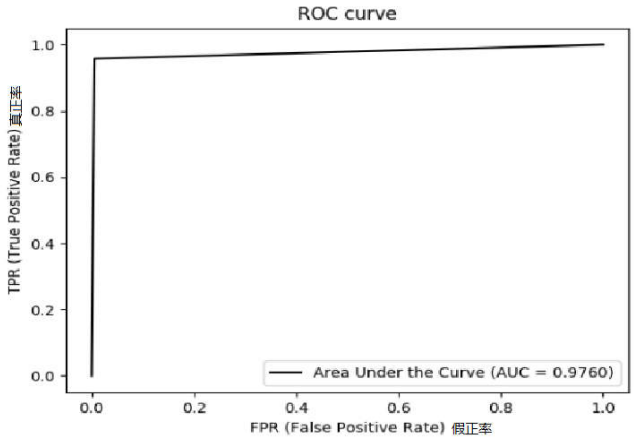


图 3.13 接受者操作特性曲线

3.6 手势定位算法

因为不能百分百完全分割正确，所以胶囊分割模型存在一定的噪声、斑点。为了确保分割出来的手势图像可以作为矩阵胶囊手势分割模型的输入，所以使用了定位算法。图 3.14 给出手势定位算法流程图，包括有模糊去噪、腐蚀去斑、计算最大轮廓包围框、裁剪原图、二值化图以及合并裁剪出来的图。其中模糊去噪采用 9×9 内核的低通滤波器平滑图像，使得每个像素替换为该像素周围像素的均值，这样可以去除分割图的噪声；进行图像腐蚀可以去出一些大点的白色斑块，使得定位更加精准。算出腐蚀剩下目标的轮廓区域，根据这些区域求得最大边框（bbox）；根据 bbox 裁剪出原图和二值化图，最终将这两图合并得到彩色手势图像。

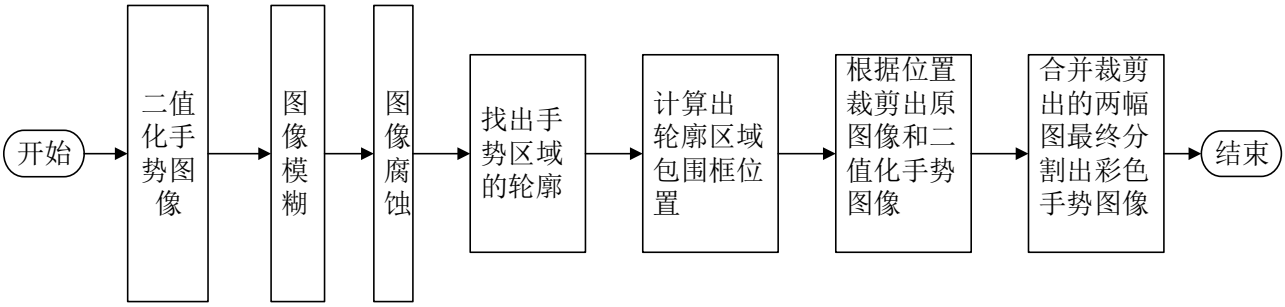


图 3.14 手势定位算法流程图

图 3.15 给出了手势定位效果图，从左到右依次为原二值化预测图、模糊图、腐蚀图、定位图、裁剪合并图。

图 3.15 手势定位效果图

3.7 基于矩阵胶囊网络的手势识别

3.7.1 矩阵胶囊网络原理

将前面定位输出的手势图像输入到训练好的矩阵胶囊网络模型中，进行手势具体分类。手势具体分类采取矩阵胶囊网络模型，矩阵胶囊网络模型由普通卷积层、主胶囊层（PrimaryCaps）、胶囊卷积层（ConvCaps）、胶囊分类层组成。卷积神经网络的每个神经元是标量输出，胶囊网络是让每个神经元向量输出，这样能够保留更多的图像特征，如方向、姿势、粗细、位置、尺寸等特征，而矩阵胶囊网络则是把每个神经元向量做成一个 n×n 大小的姿态矩阵，在做姿态变换的时候，矩阵运算能够比向量式胶囊运算节省很多计算开销，图 3.16 给出了本章设计的矩阵胶囊网络模型。

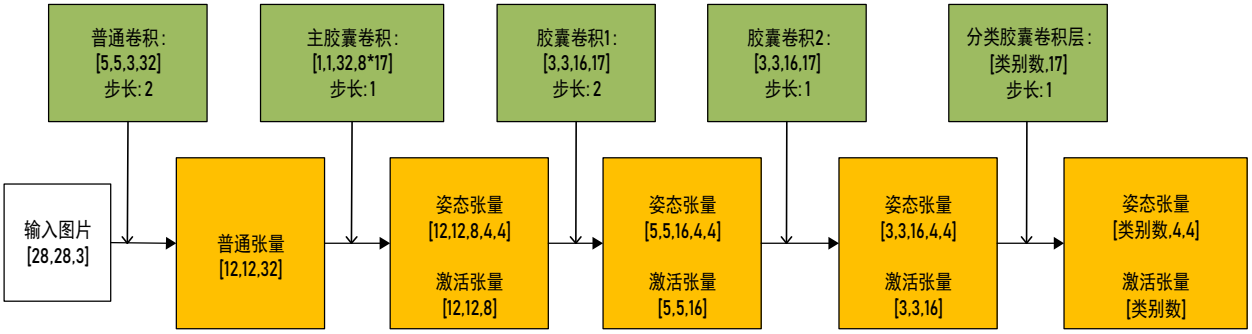


图 3.16 矩阵胶囊网络模型

矩阵胶囊网络的胶囊卷积层依次实现卷积、姿态变换以及以 EM（Expectation-Maximization）动态路由三个步骤。胶囊分类层依次实现姿态变换、EM 动态路由和平均池化，最后输出每个类别对应的姿态和激活值。

图 3.16 中的胶囊卷积层 1 和胶囊卷积层 2 前面的两个维度表示使用的卷积核大小为 3×3；第三个维度的 16 表示输出的胶囊数，该值的大小由姿态变换投票输出的数量决定；第四个维度的 17 用于拆分成 4×4 的姿态矩阵和 1×1 的激活值。胶囊卷积层的卷积是为了提取高级特征以及让张量获取正确的维度空间。姿态变换是为了让 CNN 容忍视角的一些小变动，令胶囊乘以一个变换矩阵 W 得出一个投票矩阵，从而能够应对图像即使被旋转了一些角度也能够进行识别。对所有投票矩阵进行 EM 动态路由处理，若有多少类则聚为多少类。在 GMM[47]（Gaussian Mixed Model）使用 EM 算法[48]实现了聚类过程，其中 E 步为公式（3.18）所示。该过程是将特征向量聚类为 k 个 gauss 分布。式中 x_i 为输入的投票向量，a_j 代表为第 j 类的高斯混合系数，$N(\mathrm{x}_i;\mu_j,\sigma_j^2)$ 代表数据 x_i 在第 j 类的高斯分布，分母代表 k 个混合高斯分布之和，最终求得后验概率 $p(\mathrm{j}\,|\,\mathrm{x}_i)$。

$$p(\mathrm{j}\,|\,\mathrm{x}_i)=\frac{a_j N(\mathrm{x}_i;\mu_j,\sigma_j^2)}{\sum_{j=1}^{k} a_j N(\mathrm{x}_i;\mu_j,\sigma_j^2)} \tag{3.18}$$

M 步为公式（3.19）到（3.21），由公式(3.19)算出中间值，再求出公式（3.20）第 j 类

的均值以及公式（3.21）的方差值。

$$\mathrm{r}_{ij}=\frac{p(\mathrm{j}\mid \mathrm{x}_i)}{\sum_{i=1}^{n} p(\mathrm{j}\mid \mathrm{x}_i)} \tag{3.19}$$

$$\mu_j=\sum_{i=1}^{n} r_{ij}x_{ij} \tag{3.20}$$

$$\sigma_j^2=\sum_{i=1}^{n} r_{ij}(\mathrm{x}_{ij}-\mu_j)^2 \tag{3.21}$$

由公式(3.22)求得熵 $\cos t_j$，若熵值越小则最有可能属于第 j 类，并通过公式(3.22)的 *sigmoid* 函数将值压缩到 0 到 1 之间作为激活函数，即高斯混合系数。（3.23）式中选择加入 λ 是退火策略[49]，该值作为温度值的倒数，随着训练次数的增加，温度下降让 λ 慢慢增大，使得激活函数也慢慢增大。

$$\cos t_j=\left(\beta_\mu+\sum_{l=1}^{d}\ln\sigma_j^l\right)\sum_i r_{ij} \tag{3.22}$$

$$a_j=sigmoid(\lambda(\beta_a-\cos t_j)) \tag{3.23}$$

每层胶囊层都分配有公式（3.22）和（3.23）中的参数 β_a 和 β_μ，该参数通过反向传播进行训练。公式（3.18）到（3.23）经过 3 次迭代，从而实现动态路由。

将激活向量输入到 margin_loss 函数中，该函数如公式（3.24）所示：

$$\text{margin\_}loss=\frac{1}{k}\sum_{k=1}^{k}L_k=\frac{1}{k}\sum_{k=1}^{k}\left[T_k\times\max\left(0,m^+-\|v_k\|\right)^2+\lambda\times\left(1-T_k\right)\times\max(0,\|v_k\|-m^-)^2\right] \tag{3.24}$$

式中的 k 指的是第 k 分类，margin loss 把每个分类的 loss 都加起来再取平均值。式中的 λ 是比例系数，调节两者的权重。式中的 m^+、m^- 分别取值 0.9 和 0.1，则 L_K 若要为 0，那么当第 k 分类为正样本时（即 T_K 为 1），$\|v_k\|$ 的长度必须要超过 0.9 才不会有 loss 误差，当第 k 分类为负样本时（即 T_K 为 0），$\|v_k\|$ 的长度必须小于 0.1 要且才不会有 loss 误差。把预测的结果和真实值输入 loss 函数，然后进行权值更新。

3.7.2 矩阵胶囊网络的改进

为了让最后两层卷积胶囊层能够获得手势图像丰富的高级特征，对前两层网络（普通卷积层和主胶囊卷积层）使用了多尺度卷积和恒等映射方法进行改进。仅用一种尺度通道会导致很多特征提取不完整，导致最后两层胶囊卷积层的投票作用也不明显，第一个改进是把原方法的 5×5 卷积核做成多尺度卷积，第一分支加入 2×2 的池化层和 2×2 的卷积核，第二分支加入了两个 3×3 的卷积核，第三分支保持原先的 5×5 大卷积核，第四分支使用 1×1 的卷积核，最后把不同的通道拼接从而获得不同的低级特征。由于动态路由会导致训练损失值收

敛过慢，第二个改进是把主胶囊卷积层的输入和输出特征融合，强化了信息流通，加快了收敛速度，图 3.17 是对矩阵胶囊网络前两层改进后的结构图。

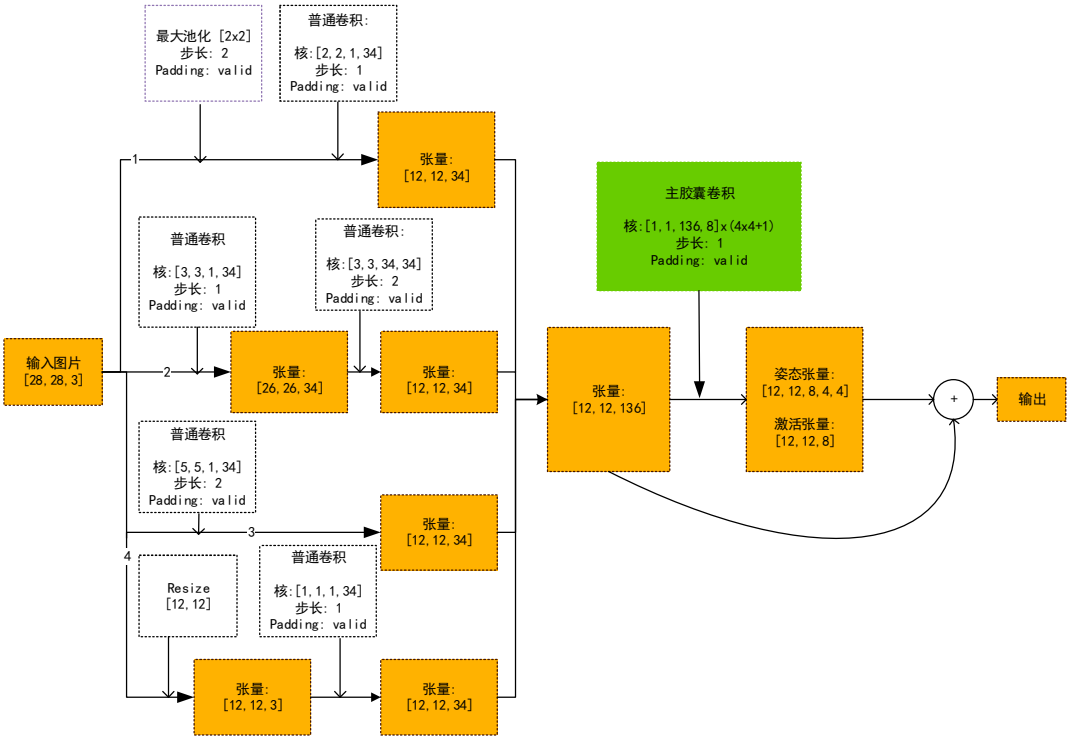


图 3.17 网络前两层改进图

3.7.3 矩阵胶囊网络的实验数据集

实验主要验证矩阵胶囊算法对手势图像识别的效果。本章的实验使用的数据集分别为 ASL 数据集[50]和 ISL 数据集[51]。ASL 数据集为美国手语手势数据集，该数据集的图片都是二维静态图，拍摄了大量志愿者手势的正面角度，包含有 26 种字母手势和 10 种数字手势，总数量为 2515 张，图 3.18 为 ASL 数据集的 36 种手势图。

图 3.18 ASL 数据集的 36 种手势

ISL 数据集为意大利手语手势数据集，数据集拍摄了 11 个志愿者总共 11008 张图片，该

数据集共 22 种手势分类，每种分类都分别从前、后、左、右和顶部这 5 个不同的角度以及分为白天或晚上两个时间段进行拍摄，ISL 数据集如图 3.19 所示。

图 3.19 ISL 数据集的不同手势和 5 种不同的拍摄角度

3.7.4 矩阵胶囊网络的实验结果与分析

图 3.20 的纵坐标为传统的 4 层卷积神经网络（classic_cnn-L4、黄色曲线）、矩阵胶囊网络（红色曲线）及本章改进后的矩阵胶囊网络（黑色曲线）使用 ASL 数据集在训练过程中的手势图像识别准确率，横坐标为训练轮数，总共训练了 150 轮。由图可见改进后的模型的手势图像识别率高于原始方法，尽管传统的 4 层卷积神经网络训练要快于胶囊网络，但是训练到后期手势图像识别准确率难以再继续提高了，说明胶囊网络的手势识别率高于传统的 4 层卷积神经网络。

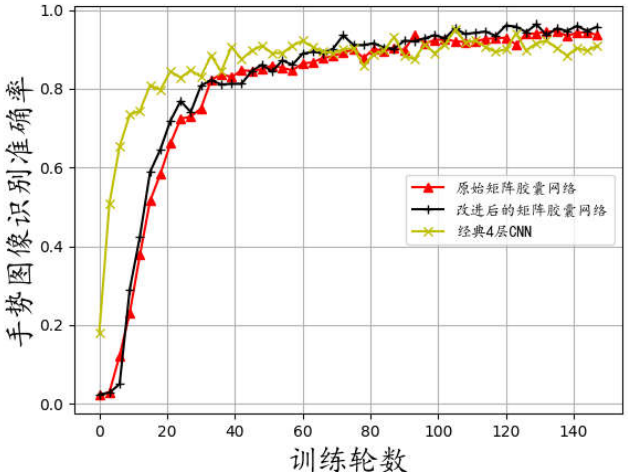


图 3.20 ASL 数据集在不同算法的手势图像识别准确率比较曲线

图 3.21 给出了矩阵胶囊网络改进前后的 loss 曲线，从图中可以看出，改进后的 loss 值也比原始算法要低，可知改进后的模型从数据中学到了更多的内容。

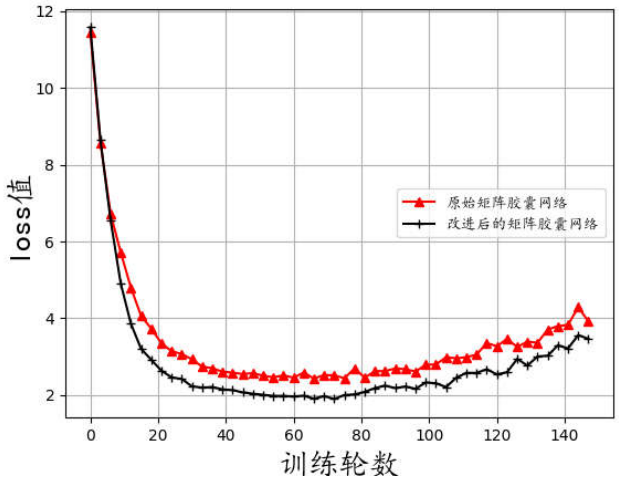


图 3.21 ASL 数据集训练总损失值曲线

图 3.22 给出了 ISL 数据集的手势图像识别准确率训练曲线，由于 ISL 数据集拍摄的角度和图片数量更多，而且矩阵胶囊网络更善于识别多角度的手势类别，因此比传统的 4 层卷积神经网络的识别率更高。

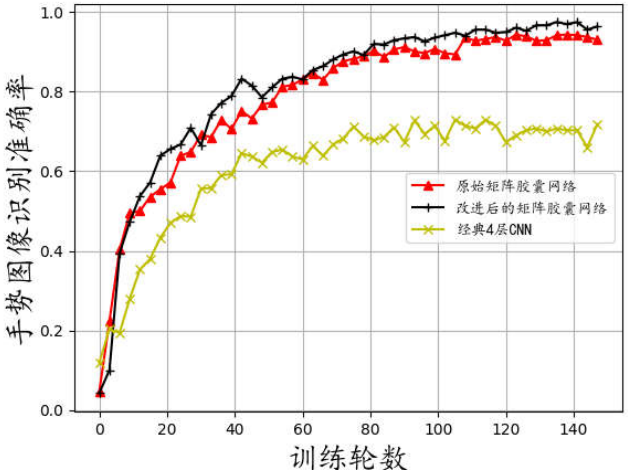


图 3.22 ISL 数据集在不同算法的手势图像识别准确率比较曲线

图 3.23 为 ISL 数据集经过 150 个 epoch 训练得到的总损失值曲线图，改进后的 loss 值更低。

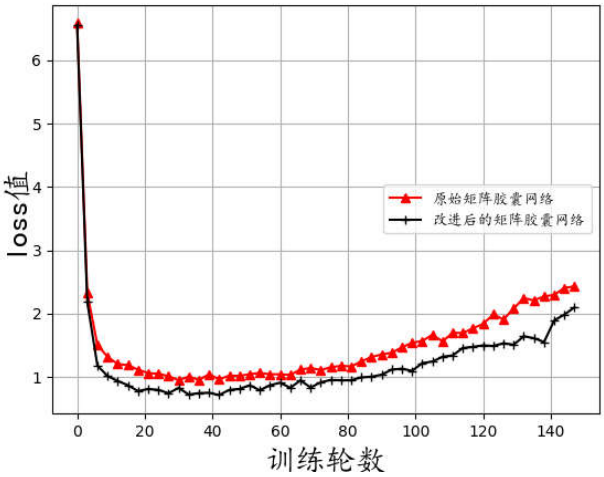


图 3.23 ISL 数据集训练总损失值曲线

为了更准确分析每类图像正确分类的情况，实验结果得到图 3.24 中关于 ASL 数据集的 36 个手势图像识别准确率的混淆矩阵。对角线位置颜色较深，对应 Y 轴各类手势识别准确率，其他方块为被误分为其他类别的概率。由于数字手势 0 和字母手势 o、数字手势 2 和字母手势 v 以及数字手势 6 和字母手势 w 都比较相似，容易被误分到其他类别，因此手势图像识别准确率比较低，但其它大多数手势的识别率都比较高。从表中的结果可以看出，使用改进的矩阵胶囊网络的方法进行手势动作的识别可以达到一个较满意的结果。

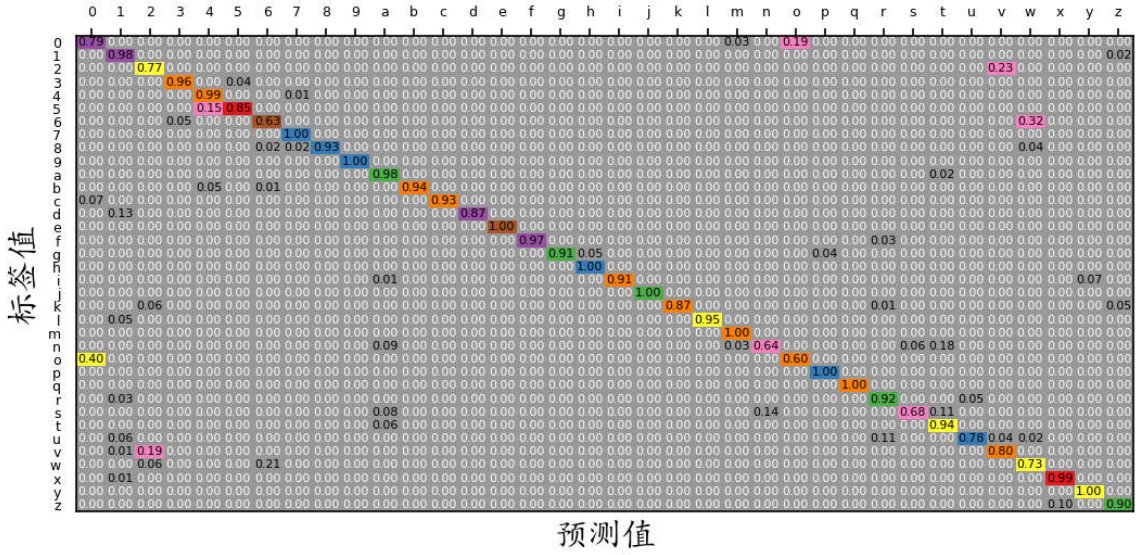


图 3.24 ASL 数据集各类手势图像识别准确率混淆矩阵

表 3.2 的测试结果为分别使用各自的测试数据集在各个模型测试取得的平均识别率。本章最终算法分别使用 ASL 手势数据集和 ISL 手势数据集的识别率比原算法分别提高了 3.57% 和 3.33%。改进后的 Capsule-EM 能有效提高手势识别率，且比同等层数的传统 CNN 的参数量要少。在学习多视角的 ISL 手势数据集，Capsule-EM 模型明显比传统 CNN 模型要更占优势。

表 3.2 不同模型的测试集准确率比较

Model	ASL 数据集 accuracy/%	ISL 数据集 accuracy/%	模型参数量
传统 CNN-L4	90.59	74.24	1,657,270
论文[34] Capsule-EM	90.99	93.59	1,599,386
本章 Capsule-EM	94.56	96.92	1,624,774

使用所有本章上述所有算法，给出了图 3.25 本章的最终手势识别的效果图，从左到右依次为原图、分割输出图、定位图、预测结果。

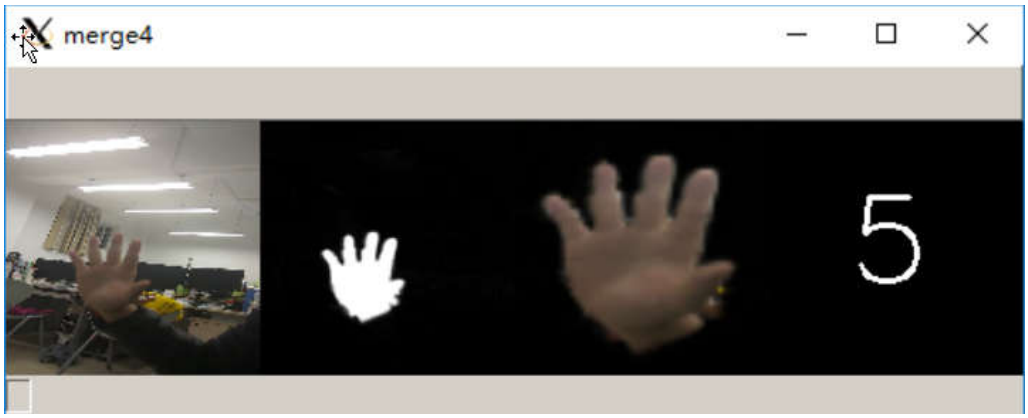


图 3.25 手势识别最终效果图

3.8 本章小结

本章提出了一种基于改进胶囊网络与算法的手势图像分类方法。主要由算法及相应的软件部分构成，主要包括视频的截帧、图像增强、手势图像的分割、手势定位的计算和手势的具体分类。首先使用了 U 型残差胶囊网络实现手势图像分割，并通过改进算法使得网络能够正常训练，再跟其他模型对比各项图像分割性能指标以及分析网络特征层的变化。然后改进了矩阵胶囊网络模型，研究了同层多尺度卷积核以获得更加丰富的特征以及使用残差网络的恒等映射方法加强信息流通和加快收敛速度，并使用了混淆矩阵进行分析，最后通过两个数据集的实验证明，本章所改进的模型均比原始模型的手势识别率提高了 3~4%，并且 loss 值更低。

第 4 章 基于 Tiny Yolo v3、Deep-sort 和 DenseNet 多模型与算法综合的手势识别

Tiny Yolo v3[52]是 Yolo（You only look once）系列的一种端到端的微型多目标检测器，在此之前还有 R-CNN（Region-CNN）系列和 SSD（Single Shot MultiBox Detector）[53]等检测器。本章使用 Tiny Yolo v3 做为手的检测器，该检测器是 Yolo 系列检测器的一种精简版，能够保证在复杂场景下进行快速的检测，经过实验测试可以达到每秒 7~9 帧。本章使用 Tiny Yolo v3 构架，并在此基础之上采用基于 coco 数据集的二次迁移学习策略，即利用已经训练好的模型再重新训练达到检测出手的目的，再用 Deep-sort（Simple Online and Realtime Tracking with a Deep Association Metric）进行手势跟踪，实现手势重识别，追踪特定的手势目标，最后利用 DenseNet 模型实现具体的手势分类。

4.1 手势识别系统整体设计

手势识别系统整体设计如图 4.1 所示，结构使用了 Tiny Yolo v3 进行多种手势对象检测，再把检测成功的手使用有卡尔曼跟踪预测的功能 Deep-sort 跟踪算法进行手势 ID 标注，如果需要进行精确的手势分类，则使用 DenseNet 对跟踪裁剪出来的手势图片进行识别。

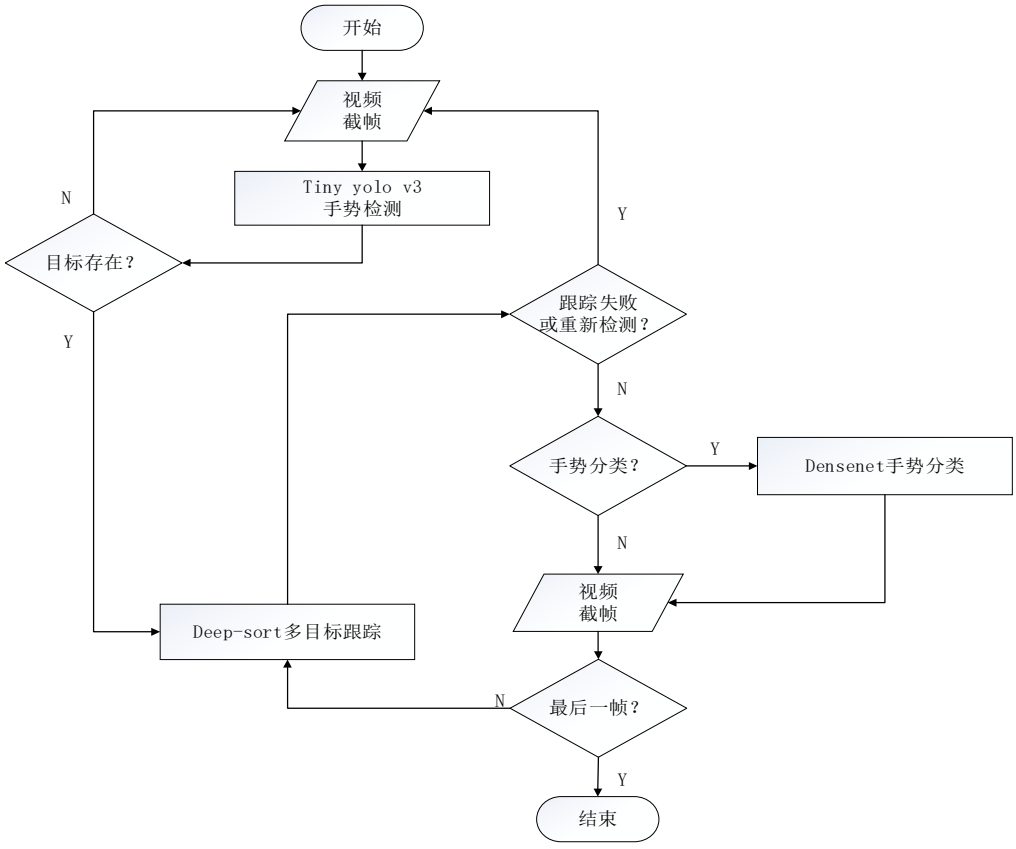


图 4.1 手势识别系统整体设计流程图

4.2 基于 Tiny Yolo v3 模型的手势检测

4.2.1 Tiny Yolo v3 的网络结构

目前 Yolo 算法已经从 v1 版本升级到 v3 版本，而 Tiny Yolo v3 是牺牲了精度但是得到更快的检测速度的模型。图 4.2 的 Tiny Yolo v3 模型通过输入一张 416×416 大小的图片为例，整个结构的特征图长宽分别经历了 5 次缩小和 4 次缩小，每次缩小 2 倍，最终在末端得到缩小了 32 倍的 y1 和缩小了 16 倍的 y2，因此要求输入图片是 32 的整数倍。图中的 DBL 表示 Darknet Conv、BatchNorm 和 Leaky Relu。特征块 y1 和 y2 的大小不同，实现了多尺度预测的目的。

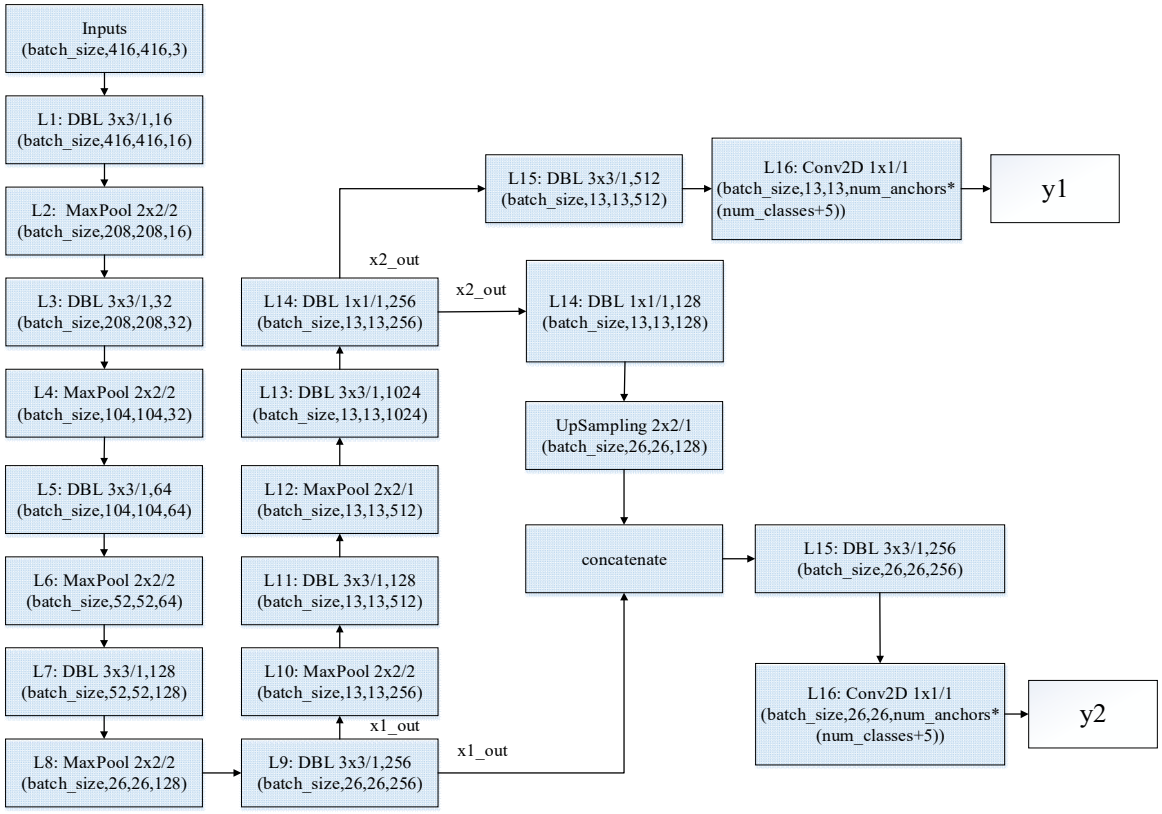


图 4.2 Tiny Yolo v3 的网络结构

4.2.2 Tiny Yolo v3 数据集的制作和图像增强

数据集利用 Labelme 开源标注软件对手势图像进行标注，图 4.3 为数据的标注，每种检测的手势类型各标注 200 个，共 10 种手势。标注完后得到生成的 xml 标注文件，利用标注文件的信息用 Kmean 聚类算法对所有的框框的长和宽聚类成 6 个先验框（Priors anchor），用于训练 Tiny Yolo v3 结构。

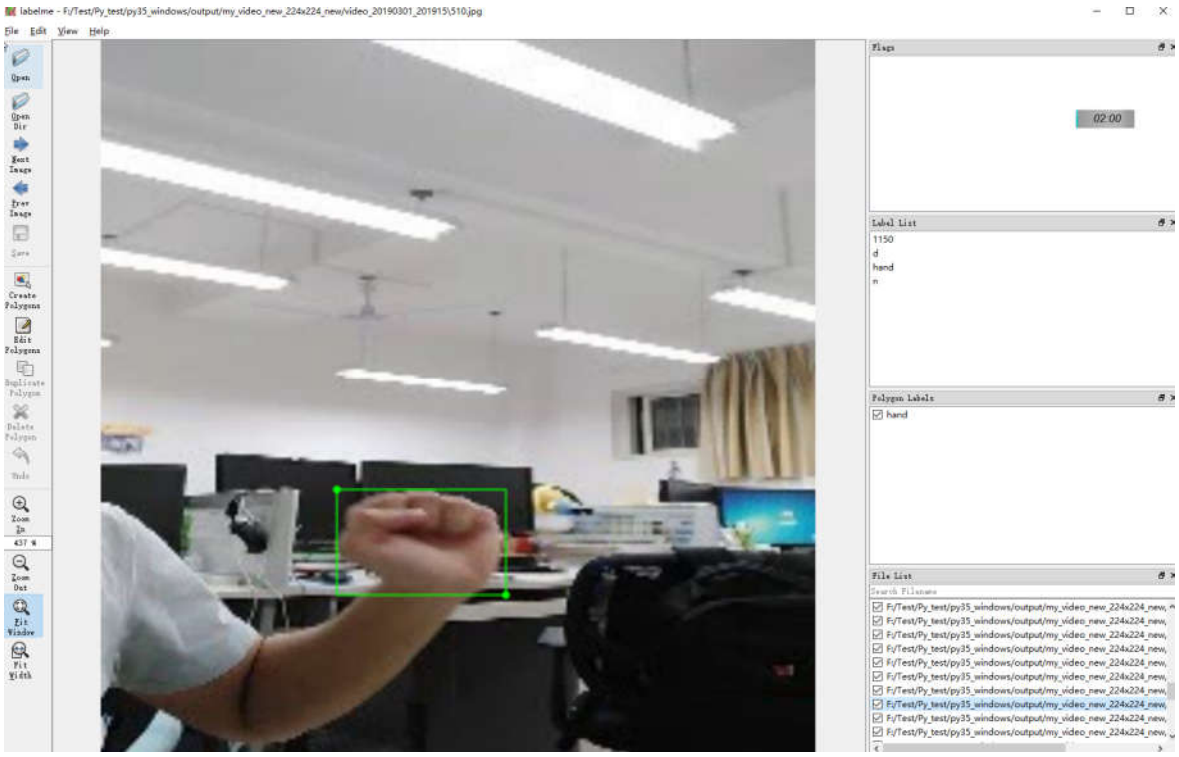


图 4.3 Labelme 开源标注软件的标注

加载手势图片之后要进行数据扩充操作，大量的数据能够有效提升检测识别率。数据扩充的手段如下：

（1）图片旋转：训练过程中对图片进行随机角度的旋转，让同一张的图片朝向每轮训练都不一样。

（2）图片翻转变换：随机进行按水平或者垂直方向去翻转图像。

（3）图片平移变换：对图像进行平移随机大小的幅度。

（4）图片尺度变换：对图片采取随机放大或者缩小的手段。

（5）图片噪声干扰：为了提升模型的抗噪能力，对所有的训练集进行随机添加椒盐噪声、泊松噪声以及高斯噪声。

（6）图片放缩因子：把所有的图片都除以 255，使得图片的每个像素点幅值大小都在 0~1 之间。

4.2.3 Tiny Yolo v3 的算法原理

算法在进行目标检测时通过把图片划分共 $S\times S$ 个单元格子，具体如图 4.4 所示，每个格子都会被单独检测。一个格子会有长度为 num_anchors×(num_classes+5)的神经元负责预测（对应图 1 的 *y1* 和 *y2*），其中，5 指的是预测框的 x、y、w、h 和 $Pre(Object)$，num_anchors 指的是每个网格子生成候选框的个数，num_classes 为能检测的目标类别数，输出 num_classes 个概率，$Pre(Object)$ 为 0~1 之间的概率，表示物体是否存在的概率。

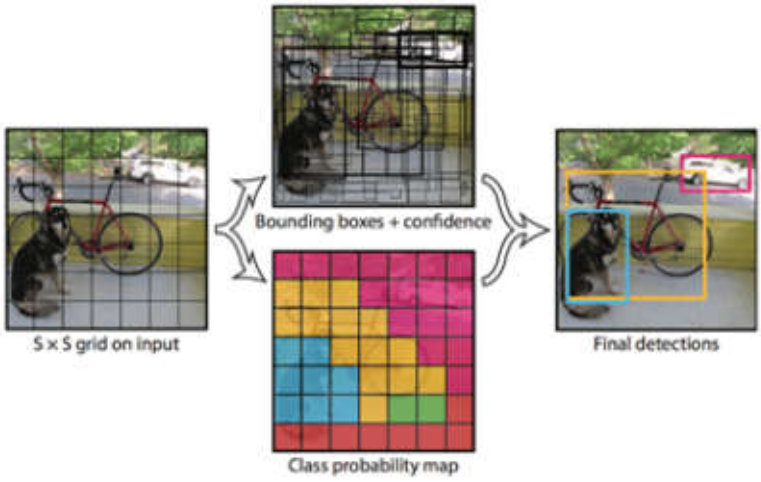


图 4.4 Tiny Yolo v3 的目标检测

Tiny Yolo v3 的预测输出经过以下四条式子，得出坐标位置和长宽。

$$x_i^p = \widehat{x_i} + \widehat{w_k}\delta_{x_{ijk}} \tag{4.1}$$

$$y_i^p = \widehat{y_i} + \widehat{h_k}\delta_{y_{ijk}} \tag{4.2}$$

$$w_k^p = \widehat{w_k}\exp(\delta_{w_{ijk}}) \tag{4.3}$$

$$h_k^p = \widehat{h_k}\exp(\delta_{h_{ijk}}) \tag{4.4}$$

根据上面四条式子计算每个预测框和前面制作数据集得到的 6 个 Priors anchor（先验框）的 IOU 值，IOU 为公式（4.5）。

$$IOU = \frac{area(\mathrm{A}) \cap area(\mathrm{B})}{area(\mathrm{A}) \cup area(\mathrm{B})} \tag{4.5}$$

利用求得的 IOU 值计算置信度 $Conf(\mathrm{Object})$。

$$Conf(\mathrm{Object}) = \Pr(\mathrm{Object}) \bullet \mathrm{IOU}_{\mathrm{Pred}}^{Truth} \tag{4.6}$$

图 4.4 中一张图片对 y1 的预测框的数量为 num_anchors×(num_classes+5)×13×13，对 y2 的预测框的数量为 num_anchors×(num_classes+5)×26×26。由于预测框的数量较多，会有大量置信度低的预测框以及重复较多的预测框，因此先根据公式（4.5）和（4.6）滤除置信度 $Conf(\mathrm{Object})$ 低的，把剩下 $Conf(\mathrm{Object})$ 较高的进行非极大抑制（NMS）筛选最合适的作为最终的预测框。NMS 用于检测高度重叠的框并将其丢弃，例如两个预测框都预测一只手，如果这两个框的重叠度比较高，则可能是预测到同一个物体，则应该去除掉其中一个框才合理。NMS 过程为:选定一个合适的值，如 0.35 作为 IOU 阈值，将所有窗口由前面求得的分数 $Conf(\mathrm{Object})$ 从高到低排序，算出分数最高的框和剩下所有的框的 IOU 值，如果 IOU 值大于阈值 0.35 就将该框删除。接着继续从剩余的框选择分数最高的，重复上述过程，一直到所有的窗口都已经被处理过为止。

通过公式（4.7）的均方和误差 loss 函数进行训练 Tiny Yolo v3 模型。

$$\begin{aligned} Loss = {} & \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathrm{E}_{ij}^{obj} \left[\left(x_i - \hat{x}_i \right)^2 + \left(y_i - \hat{y}_i \right)^2 \right] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathrm{E}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ & + \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathrm{E}_{ij}^{obj} \left(c_i - \hat{c}_i \right)^2 \\ & + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathrm{E}_{ij}^{noobj} \left(c_i - \hat{c}_i \right)^2 \\ & + \sum_{i=0}^{s^2} \sum_{c \in classes} E_i^{obj} \left(p_i(\mathrm{c}) - \hat{p}(\mathrm{c}) \right)^2 \end{aligned} \tag{4.7}$$

该 Loss 分别由坐标 xy 误差、宽高 wh 误差、目标存在 confidence 误差（惩罚 c_i 不为 1）、目标不存在 confidence 误差（惩罚 c_i 不为 0）和类别概率误差共 5 个均方误差组成。由于式子中训练的几个误差数量级不一致，因此式子中的 λ_{coord} 和 λ_{noobj} 在训练中分别设置为放大倍数 5 和缩小倍数 0.5 调整训练。E_{ij}^{obj} 和 E_{ij}^{noobj} 分别表示训练标签的目标存在（值为 1）和目标不存在（值为 0），s^2 表示划分格子的格子数，i 为处理第 i 个格子，B 表示生成的候选框的个数，j 为处理第 j 个候选框。

4.2.4 Tiny Yolo v3 的训练

由于模型需要大量的数据集和比较长的时间进行训练，本章的 Tiny Yolo v3 的训练采用二次迁移学习策略，使用经过 coco 数据集训练过能检测出 80 个目标种类的模型再次重新训练 10 种手势类别。最后利用深度学习框架 Tensorflow 内置的 Tensorboard 观察训练状况，图 4.5 为经过 200 个 epoch 的训练过程，通过曲线观察表明模型的训练已经收敛。

图 4.5 训练校验集 Loss 曲线

4.2.5 Tiny Yolo v3 的测试

经过在实验室环境的视频流测试，Tiny Yolo v3 能把训练的 10 种手势类型测试出来，

目标检测速度达 7~9 帧/s，测试集准确率达 87.02%，召回率达 85.1%。图 4.6 显示的是其中一种手势类型检测。

图 4.6 Tiny Yolo v3 的测试

4.3 基于 Deep-sort 的手势跟踪

4.3.1 sort 多目标跟踪原理

sort（simple online and realtime tracking）多目标跟踪[54]原理如图 4.7 的程序流程图，其核心方法是卡尔曼滤波和匈牙利匹配，由于算法在线更新、实时性好，是一个几乎达到 state-of-art 的在线追踪算法。图 4.7 中需要使用前面 Tiny Yolo v3 的检测框，对图像中所有的手势进行检测，并把检测框输入到卡尔曼滤波器对匀速运动的手势轨迹做出预测输出。把预测输出的追踪框和检测框使用匈牙利算法（Hungarian Algorithm）实现一一匹配，对有检测框匹配的跟踪器进行卡尔曼状态更新，并对没有跟踪器匹配的检测框作为初始值重新初始化卡尔曼跟踪器，最后筛选出最佳的预测结果进行输出。图中参数 time_since_update 表示没有进行卡尔曼状态更新的次数，如果某些跟踪器长期没有和检测框匹配，这种情况下如果累计超过 30 次则该跟踪器会被清除，新的生成的跟踪器 time_since_update 重新初始化为 0；图中的 hit_streak 用于选出卡尔曼状态更新次数累计超过 3 次的跟踪器，保证了预测的精确性；图中的 frame_count 用于选择 3 帧以内的跟踪结果，结果比较准确。

图 4.7 中用以 Tiny Yolo v3 的检测框作为 sort 跟踪的输入，以卡尔曼滤波实现目标跟踪。卡尔曼在计算机视觉领域用处极广，卡尔曼跟踪如公式（4.8）到（4.12）的 5 条公式，公式按预测、实测和修正的顺序进行递推，实现一个常量均速的线性观测模型。其中公式（4.8）到（4.9）用于预测，公式（4.10）到（4.12）用于状态更新。最后的公式（4.13）作为观测模型。

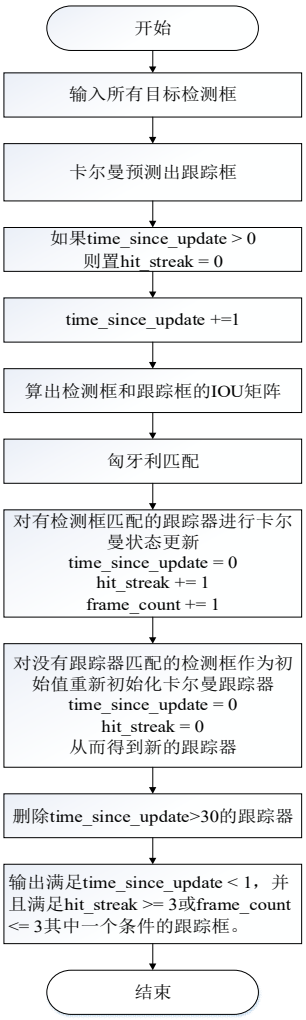


图 4.7 sort 多目标跟踪程序流程图

式（4.8）为状态预测模型，使用了前一状态的最佳结果 $X(k-1|k-1)$、系统参数矩阵 A 进行预测得到系统结果，式中的 $U(\mathrm{k})$ 为控制量，$W(\mathrm{k})$ 为过程噪声，B 为输入控制矩阵，若不需要则设置为 0。

$$X(k|k-1)=A\,X(k-1|k-1)+BU(\mathrm{k})+W(\mathrm{k}) \tag{4.8}$$

式（4.9）协方差预测方程，该式子使用了前一状态的协方差、系统参数矩阵 A、转置矩阵 A^T 和不确定性矩阵 Q 求出当前时刻的协方差。

$$P(k|k-1)=A\,P(k-1|k-1)\,A^T+Q \tag{4.9}$$

式（4.10）的 $K_g(k)$ 表示求卡尔曼增益（Kalman Gain），其中 H 表示观测矩阵，R 表示误差协方差矩阵。

$$K_g(k)=P(k|k-1)H^T\left(HP(k|k-1)H^T+R\right)^{-1} \tag{4.10}$$

式（4.11）利用了式（4.9）求出的协方差以及式（4.10）的卡尔曼增益更新误差协方差值。

$$R = (I - K_g(k)\ H)\ P(k|k-1) \tag{4.11}$$

式（4.12）利用了式（4.8）预测结果以及式（4.10）的卡尔曼增益做加权平均，从而更新当前的状态估计。

$$X(k|k) = X(k|k-1) + K_g(k)\ \left(Z(k) - H\ X(k|k-1)\right) \tag{4.12}$$

式（13）中的$Z(k)$是测量模型，H是测量矩阵，V是测量噪声。

$$Z(k)\ = HX(k) + V \tag{4.13}$$

上述公式的状态X输入为向量$(x_{center}, y_{center}, a, h, \dot{x}_{center}, \dot{y}_{center}, \dot{a}, \dot{h})$，其中$(x_{center}, y_{center})$表示物体运动中心点，$(a,h)$分别表示目标的宽高比值、高度，其余四个为在图像坐标中相应的速度信息。公式中的参数A为转移矩阵，如式（4.14）所示。

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

为了让上述工作流程更加直观，图 4.8 给出了 kalman 滤波器工作原理图，该流程的核心思想是：用当前时刻的测量数据和上一刻的预测量与误差值，从而求得到当前时刻的最优值，再去预测下一刻的量。

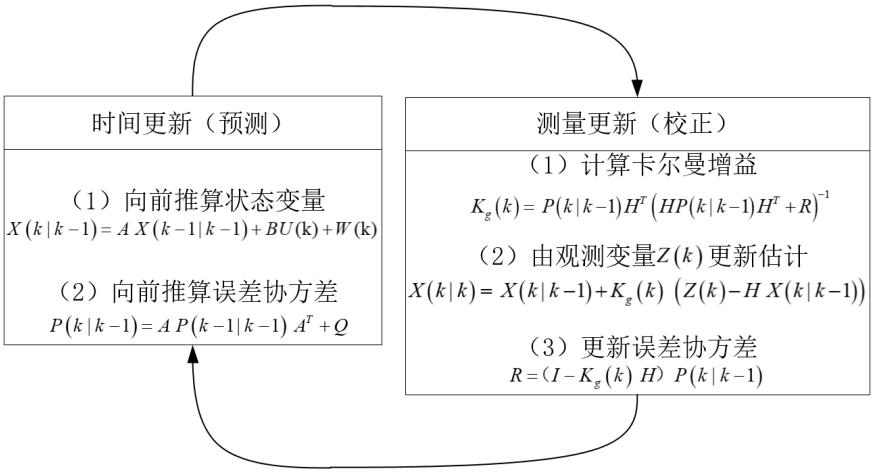


图 4.8 kalman 滤波器工作原理图

在图 4.7 用卡尔曼滤波得出追踪框后，接着算出检测框和跟踪框的 IOU 矩阵，从而能够使用匈牙利匹配算法进行指派。匈牙利匹配算法如图 4.9 所示，该算法时间复杂度为 O(n3)，作用是解决指派问题或分配问题。

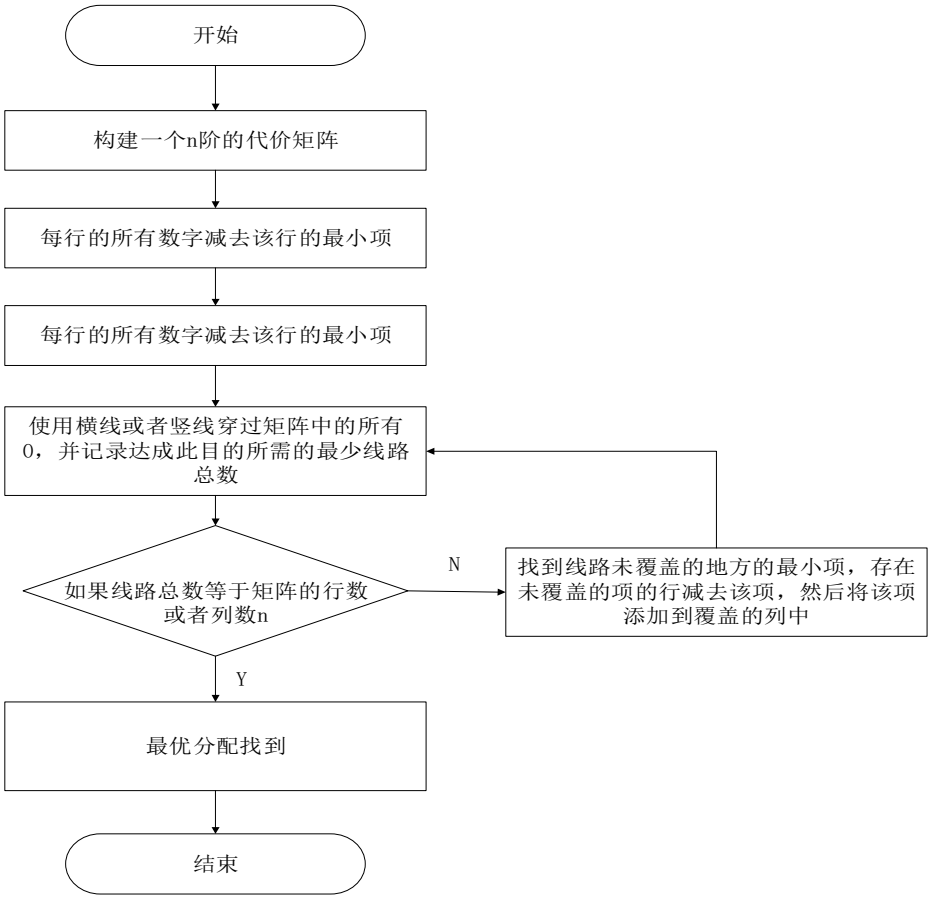


图 4.9 匈牙利匹配算法路程

图 4.9 的流程通过 IOU 代价矩阵，对所有追踪框都指派了与之对应最合理的检测框。图中构造了 n×n 的代价矩阵，目的是能够找出匹配数量最大的方案，算法过程主要分为五个步骤：

（1）建立代价矩阵 $Cost_{xy}$。

（2）把 $Cost_{xy}$ 矩阵的每一行全部的数字都减去该行当中的最小那项，使得 $Cost_{xy}$ 矩阵的每行都会有 0 元素出现。

（3）接着把 $Cost_{xy}$ 矩阵的每一列全部的数字都减去该列中的最小那项，使得矩阵的每行都会有 0 元素出现。

（4）以数量最少的直线去对 $Cost_{xy}$ 矩阵的零元素进行覆盖，若直线条的数量等于 $Cost_{xy}$ 矩阵的行数或者列数，则找到了最优的分配方案，算法流程结束。若直线条的数量少于 $Cost_{xy}$ 矩阵的行数或者列数，则进行第下一步。

（5）基于上一步找出线路没有被覆盖地方的最小那一项，对应的行减去该项，再把该项加到已经被覆盖的列当中，再重复步骤（4），直到实现指派任务为止。

4.3.2 Deep-sort 原理

Deep-sort 是基于 sort 的基础上进行改进，结合了深度学习的方法，基于深度学习的目标跟踪还有 Goturn[55]、SiameseFC[56]和 CFNet[57]等算法，这些算法比较容易和其他检测网络结合。本章加入 Deep-sort 算法对检测的手势进行跟踪，从而起到标记手势 ID 和记录手势轨迹，起到手势重识别的目的。

Deep-sort 加入了已经训练好的深度学习模型，可以有效地改善有东西遮挡的情况下的物体跟踪效果，同时也大大减少了跟踪目标 ID 跳变的情况。图 4.10 给出 Deep-sort 的算法流程，算法先用 Yolo 检测取得所有目标检测框，进行 NMS 处理，减少冗余的检测对象，将框内的图片裁剪出来进行预处理，做成特定大小的图片，输入已经训练好的 WRN 模型提取 128 维度的特征向量。算法加入了图中的级联匹配，如果用级联匹配方法无法实现检测框和追踪框的匹配，则用原始的 sort 算法进行 IOU 指派。

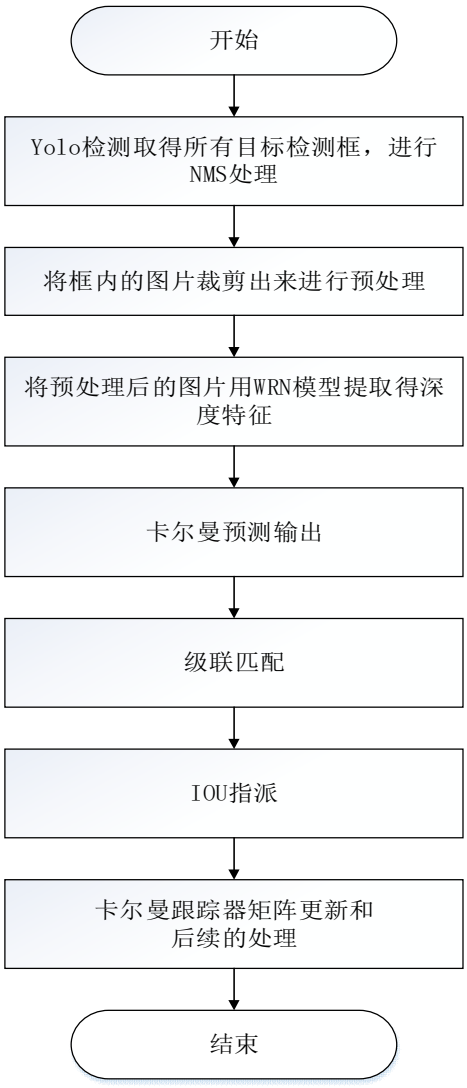


图 4.10 Deep-sort 整体流程流程图

在 Deep-sort 中，用 Wide ResNet（WRN）构建深度学习模型，其宽度残差块如图 4.11 所示，结构加上了一个宽度因子 k，实现拓宽卷积核的个数。WRN 让输出的特征宽度放大

多倍，使得这种矮胖的模型比深度模型更好训练，也大大提高了计算效率，在和深度模型相同参数量的情况下还提高了训练精度。

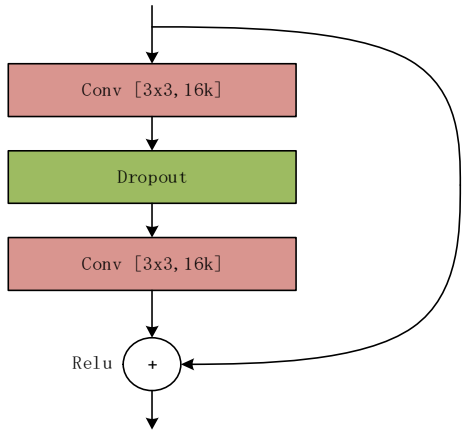


图 4.11 宽度残差块

本章的 WRN 模型具体参数如下表 4.1 所示：

表 4.1WRN 模型参数值

操作类型	卷积核大小	步长	输出维度
卷积层 1	3 × 3	1	32 × 64 × 64
卷积层 2	3 × 3	1	32 × 64 × 64
宽度残差块 4	3 × 3	1	32 × 64 × 32
宽度残差块 5	3 × 3	1	32 × 64 × 32
宽度残差块 6	3 × 3	2	64 × 32 × 16
宽度残差块 7	3 × 3	1	64 × 32 × 16
宽度残差块 8	3 × 3	2	128 × 16 × 8
宽度残差块 9	3 × 3	1	128 × 16 × 8
全连接层+dropout	-----	-----	128
BN	-----	-----	128

网络最后输出 128 维度的特征向量，训练加入了 L2 正则化，并使用 Dropout 和 Batch norm 来防止过拟合和提高性能。

4.3.3 级联匹配原理

由于单纯使用 kalman 滤波去计算运动相似度会存在缺陷，比如目标被遮挡的时间太长，卡尔曼在不断进行预测后会存在概率弥散的问题。因此 Deep-sort 加入了级联匹配，级联匹配分别使用基于最小余弦距离的表观特征匹配以及基于马氏距离的运动匹配。具体的算法流程如图 4.12 所示。

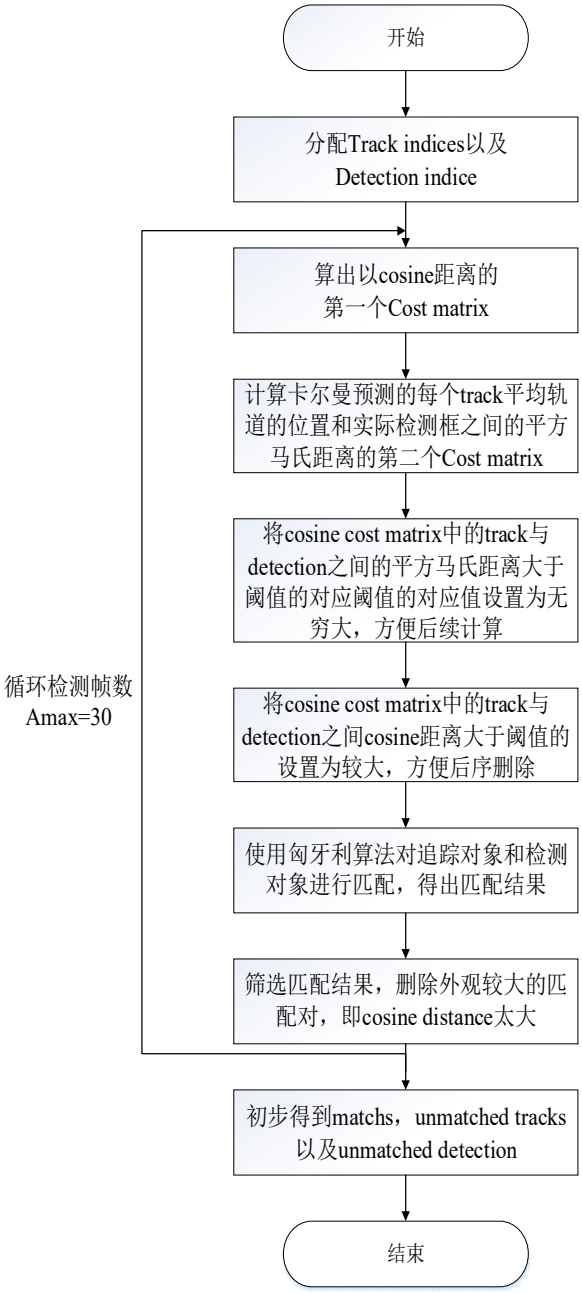


图 4.12 级联匹配算法流程

图中余弦距离为公式（4.15），余弦相似度的角度值越小则值越接近 1，则表明两者越相似。该公式衡量检测和预测的 ROI 区域内图像相似程度（表观特征相似度），即比较两者在 WRN 网络提取特征输出的 128 维度特征向量的距离。

$$\cos(\theta)=\frac{x_1y_1+x_2y_2+....+x_ny_n}{\sqrt{x_1^2+x_2^2+.....x_n^2}\sqrt{y_1^2+y_2^2+....+y_n^2}} \tag{4.15}$$

同理，而当 n 个维度的的样本点，也能用像公式（4.16）的余弦角概念进行相似度的衡量，如公式（4.16）所示：

$$\cos(\theta)=\frac{\vec{x}\bullet\vec{y}}{\left|\vec{x}\right|\left|\vec{y}\right|}=r_j^{\ T}r_k^{(\mathrm{i})} \tag{4.16}$$

要取的 cosine distance 值越小越相似，因此在外观度量上，最终的距离公式（4.17）所示：

$$d^{(2)}(\mathrm{i},\mathrm{j})=\min\{1-r_j^{\ T}r_k^{(\mathrm{i})}\mid r_k^{(\mathrm{i})}\in R_i\} \tag{4.17}$$

马氏距离（mahalanobis distance）表示的是数据的协方差距离，如式（4.18）的 M 公式所示，该公式用于衡量目标检测框和卡尔曼预测结果之间的距离，从而得到两者的运动匹配的程度。

$$M(\mathrm{x})=\sqrt{(\mathrm{x}-\mu)'\ \Sigma^{-1}(\mathrm{x}-\mu)} \tag{4.18}$$

取马氏平方距离来进行位置的度量，如公式（4.19）所示：

$$d^{(1)}(\mathrm{i},\mathrm{j})=(\mathrm{d}_j-\mathrm{y}_i)^T S_i^{-1}(\mathrm{d}_j-\mathrm{y}_i) \tag{4.19}$$

最后以加权的方式把这两个距离来相加：

$$\mathrm{c}_{i,j}=\lambda d^{(1)}(\mathrm{i},\mathrm{j})+(1-\lambda)d^{(2)}(\mathrm{i},\mathrm{j}) \tag{4.20}$$

4.4 基于 DenseNet 的手势识别

4.4.1 DenseNet 原理

为了提高手势分类效果，本章使用 DenseNet[58]进行手势具体分类。DenseNet 结构如图 4.13 所示，每个 Dense Block 是带密集连接的块，达到特征重用的目的，每个 Block 之间有转变层（Transition Layer），用于改变特征图尺度和压缩层数。

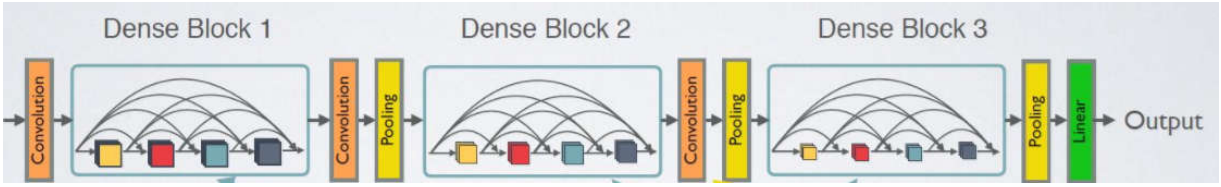
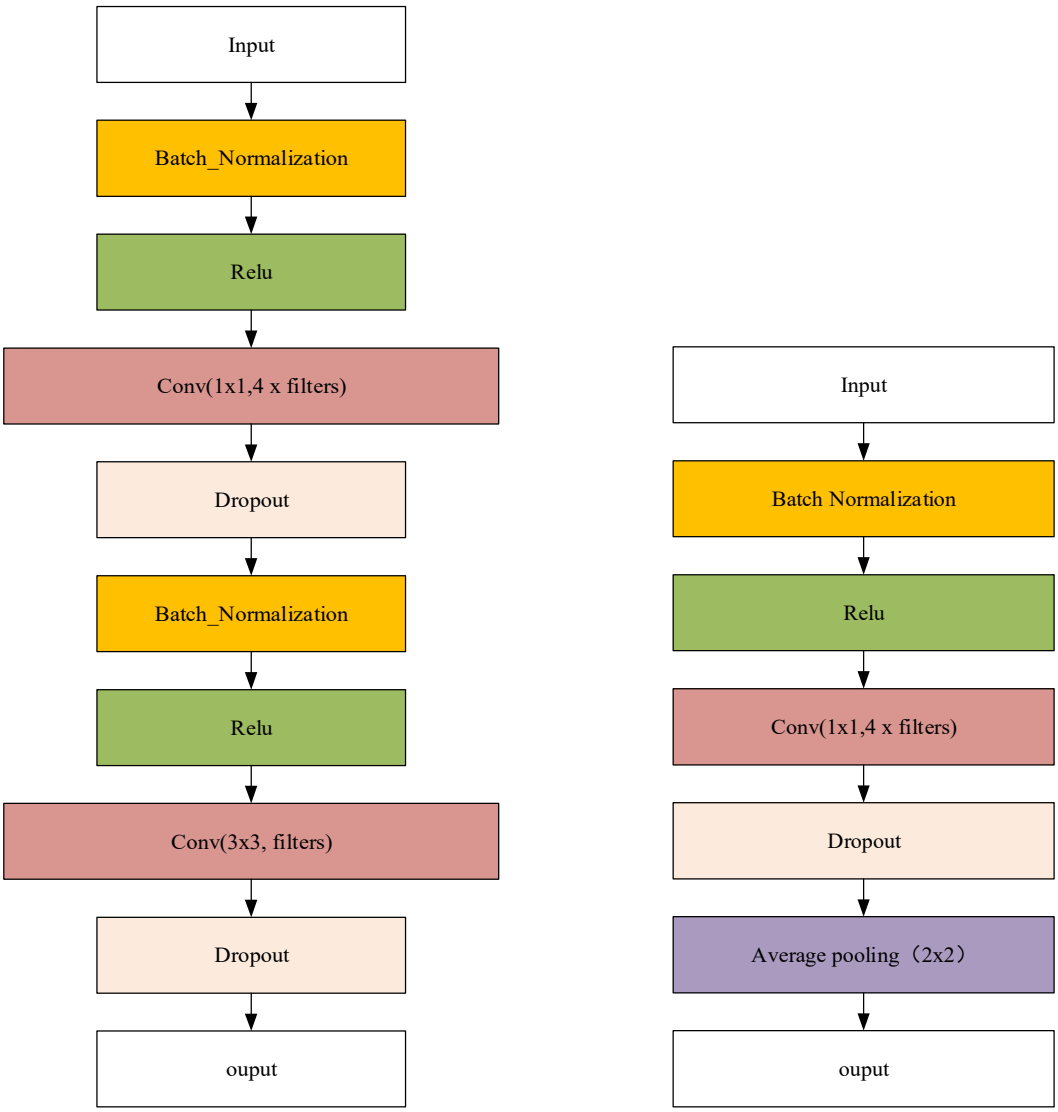


图 4.13 DenseNet 结构图

每个 Dense Block 内部的瓶颈层（Bottleneck Layer）稠密连接，瓶颈层结构图 4.14（a）所示，结构由批标准化（Batch Normalization）、激活函数 Relu、1×1 卷积层、3×3 卷积层和 Dropout 组成。转变层（Transition Layer）如图 4.14（b）所示，主要由 Batch Normalization 层、激活函数 Relu、1×1 卷积层和平均池化层（Average pooling）组成。



（a）瓶颈层　　（b）转变层

图 4.14 瓶颈层和转变层结构

4.4.2 DenseNet 的搭建和训练结果

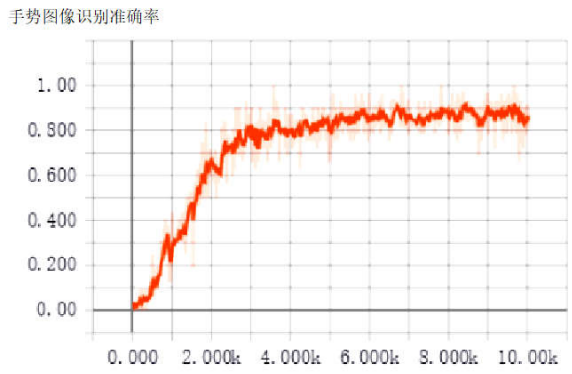
表 4.2 给出了本章搭建的 DenseNet 的网络结构的参数设置，表中的卷积层共有 100 层，网络最终实现对 10 种手势的具体分类。

表 4.2 DenseNet 的网络结构的参数设置

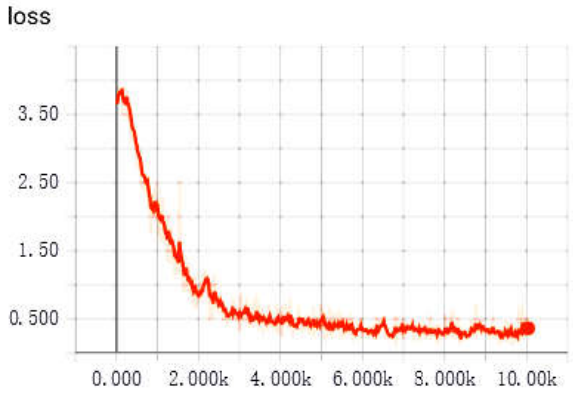
操作类型	输出维度	卷积层数
Input	(32, 28, 28, 3)	- - - - -
Conv_layer	(32, 14, 14, 48)	1
Dense_block_1	(32, 14,14,192)	6
Transition_layer_1	(32, 7, 7, 24)	1
Dense_block_2	(32, 7, 7, 312)	12
Transition_layer_2	(32, 3, 3, 24)	1
Dense_block_3	(32, 3, 3, 1176)	48
Transition_layer_3	(32, 1, 1, 24)	1

Dense_block_4_final	(32, 1, 1, 792)	32
Batch_Normalization + Relu	(32, 1, 1, 792)	- - - - -
Global_Average_Pooling	(32, 792)	- - - - -
Flatten	(32, 792)	- - - - -
Linear	(32, 10)	- - - - -

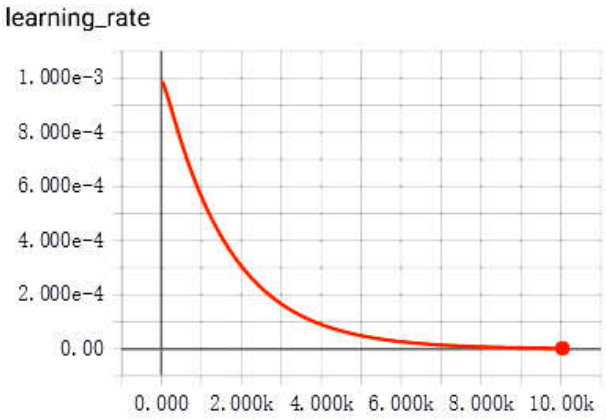
图 4.15（a）、（b）和（c）分别给出了 Tensorboard 在训练过程显示的手势图像识别准确率曲线、loss 曲线和学习率变化曲线，经过 10,000 步训练，DenseNet 已经收敛，学习率从 1e-3 跌到 1e-5，手势识别准确率最终为 87.5%。



（a）手势图像识别准确率曲线



（b）loss 曲线

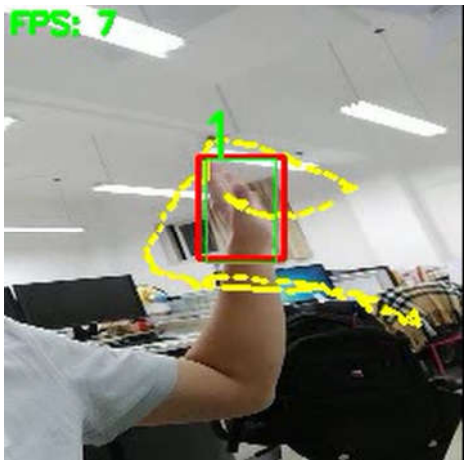


（c）学习率变化

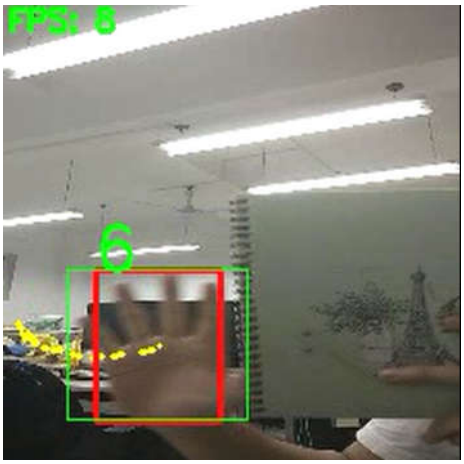
图 4.15 训练过程的手势图像识别准确率、loss 和学习率曲线

4.5 手势识别系统测试结果

图 4.16（a-d）给出了手势检测跟踪效果图，黄色的线为 Deep-sort 跟踪的中心点轨迹，绿色框为 Deep-sort 预测框，红色框为 Tiny Yolo v3 的手势检测框，算法流程速度为 7~8 帧每秒（FPS）。图中的数字 1 和数字 6 分别是 Deep-sort 分配给左手和右手的手势 ID（identity），给不同的手分配不同的 ID，从而能够追踪特定的手，这种手势重识别手段能在日常应用中有效防止其他人的手势干扰，从而实现了特定手势跟踪的人机手势控制。

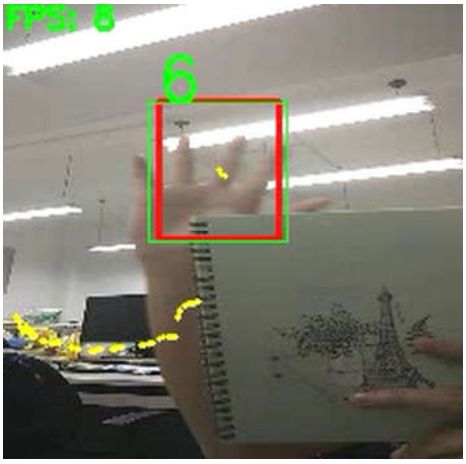


（a）左手手势检测跟踪效果图



（b）右手手势遮挡前

（c）右手手势被遮挡



（d）右手重新出现后再跟踪

图 4.16 手势检测、跟踪和识别

4.6 本章小结

本章首先给出了基于 Tiny Yolo v3、Deep-sort 和 DenseNet 多模型与算法综合的手势识别手势识别的整体流程图；其次介绍 Tiny Yolo v3 的基本原理，并进行训练实现了 10 种手势类别的检测，检测速度达到每秒 8~9 帧；第三介绍了 Deep-sort 原理和算法流程图，其主要方法为使用宽度残差网（WRN）提取检测目标的深度特征（外观模型）从而减少被遮挡情况下 id 被切换的次数、并使用卡尔曼预测、求出余弦距离代价矩阵（cost matrix）和平方马氏距离 cost matrix、进行匈牙利匹配及 IOU 指派、矩阵更新后续处理，从而实现了多个手势的跟踪；在实现了对手势图像的检测和跟踪的基础上，基于 DenseNet 结构，对 DenseNet 进行训练，能够实现在复杂背景下的手势轨迹识别和手型分类，整个算法流程速度为每秒 7~8 帧。

第 5 章　总结与展望

5.1 总结

手势的识别在许多领域受到广泛的应用，涉及到计算机视觉领域的图像分割、目标跟踪、目标检测和图像识别等，由于不同算法都有自己的局限性，是否各项性能都能保证达到较高的水平是算法研究的难点。计算机视觉的各种算法经过不断地发展和较深入研究，各个领域每年都有大量十分优秀的算法和思想出现，但计算机视觉在手势识别的应用领域，在受到如手势形变、手势拍摄的光照影响、背景相似、手势快速运动和运动模糊等各种复杂环境的影响，仍然存在很多问题需要解决。

本文的研究工作主要为下面几个方面：

（1）阐述手势识别在国内外研究发展历程、研究现状和研究的意义，并简述了本文的总体框架设计。

（2）对卷积神经网络的基础理论做了阐述，包括卷积神经网络的原理、卷积层和反卷积层的特征输出、目标函数的优化、以及残差技术原理，为后续手势图像识别算法的研究和实现奠定了理论基础。

（3）提出了基于改进胶囊网络与算法的手势图像分割与识别方法，即通过手势分割、手势定位和手势识别的方式实现了手势识别。首先对深度学习 Tensorflow 框架的实验平台做了介绍，分析了该框架的特点和本文训练环境的配置；其次，给出了整个视频流截帧、图像增强、手势图像分割、手势定位和手势图像识别的整体流程图；第三，在阐述了原始胶囊网络原理后，提出了一种改进的胶囊网络算法，再结合 U-net 网络和残差技术，实现了 U 型残差胶囊分割网络，完成了对手势图像的分割；第四，阐述了矩阵胶囊网络原理，在此基础上进行结构优化，完成了对不同类别的手势图像的识别；最后介绍了公开数据集的特点和本文制作数据集的流程，通过数据集的训练和测试，得出了手势图像分割的性能指标、P-R 曲线、ROC 曲线、网络中间层特征可视化图、手势图像识别的混淆矩阵、手势图像识别 loss 曲线和识别准确率曲线。通过实验结果表明，提出的改进模型达到了较好的识别效果。

（4）研究了基于 Tiny Yolo v3、Deep-sort 和 DenseNet 多模型与算法综合的方法实现手势识别。具体研究分析了 Tiny Yolo v3 检测算法、Deep-sort 目标跟踪算法和 DenseNet 算法，通过制作数据集进行模型的训练，首先训练好 Tiny Yolo v3、Deep-sort 和 DenseNet 三个模型，然后将几种模型和算法进行有机整合，实现了一种在复杂背景下对手势的检测、追踪和姿态的识别方法，算法流程速度达到每秒 7~8 帧。

5.2 展望

本文是基于计算机视觉的手势识别技术研究，主要通过了两种方法去实现。本文的第一种手势识别技术方法：基于改进胶囊网络与算法的手势图像分割与识别方法，通过分割网络去掉背景干扰以及使用胶囊网络实现了较高的手势识别率和较小的模型量，但是由于胶囊层比较复杂，模型的训练耗费的时间和识别速度是传统卷积神经网络的十几倍，分割、定位和识别整个算法流程去识别每个手势动作仅为 1~2 帧每秒，因此该改进算法需要在提高识别速度上做进一步的研究。本文第二种手势识别技术方法是：基于 Tiny Yolo v3、Deep-sort 和 DenseNet 的手势识别方法，速度可以达到每秒 7~8 帧，但是用的模型较多，参数量比较大，需要至少 2 张 GPU 显卡加速，硬件开销比较大，手势图像识别准确率也不如第一种方法，因此可以在压缩模型和手势图像识别准确率上做进一步改进。下面给出了今后的研究重点：

（1）胶囊卷积层的动态路由人为设计因素太过复杂导致训练时间过长，因此胶囊卷积层是否有其他更加合适的聚类算法，仍值得今后进一步研究。

（2）算法中结合强化学习的手段提高模型性能，减少人为设计神经网络，让计算机自主学习，自动搜索找出最佳的网络堆叠方案。

（3）算法中借鉴对抗网络的辅助，实现被手势被遮挡情况下的手势的还原和识别，从而提高算法的鲁棒性。

（4）进行算法上的改进或借鉴其他优良的处理手段，从而加快模型训练速度和收敛速度、并提高模型性能指标和减少模型参数量。

（5）将技术应用到如手机移动端或手势云台控制等一系列嵌入式产品中，实现在日常生活中的手势识别应用。

参考文献

[1]. Russell S J, Norvig P. Artificial intelligence: a modern approach[M]. Malaysia; Pearson Education Limited, 2016.

[2]. Yildirim G, Hallac İ R, Aydin G, et al. Running genetic algorithms on Hadoop for solving high dimensional optimization problems[C]//2015 9th International Conference on Application of Information and Communication Technologies (AICT). IEEE, 2015: 12-16.

[3]. Voulodimos A, Doulamis N, Doulamis A, et al. Deep learning for computer vision: A brief review[J]. Computational intelligence and neuroscience, 2018, 2018.

[4]. YANG M, TAO J. Intelligence methods of multi-modal information fusion in human-computer interaction[J]. SCIENTIA SINICA Informationis, 2018, 48(4): 433-448.

[5]. Zhang G, Zhang D. Research on vision-based multi-user gesture recognition Human-Computer Interaction[C]//2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing. IEEE, 2008: 1455-1458.

[6]. Pu Q, Gupta S, Gollakota S, et al. Whole-home gesture recognition using wireless signals[C]//Proceedings of the 19th annual international conference on Mobile computing & networking. ACM, 2013: 27-38.

[7]. Prasad K D, Bhattacharya A, Murty S V S. An ambient light sensing module for wireless sensor networks for planetary exploration[J]. Planetary and Space Science, 2012, 70(1): 10-19.

[8]. YUAN M, YAO H, LIU J. Dynamic gesture segmentation combining three-frame difference method and skin-color elliptic boundary model[J]. Opto-Electronic Engineering, 2016 (6): 10.

[9]. Adamovich S V, Merians A S, Boian R, et al. A virtual reality—based exercise system for hand rehabilitation post-stroke[J]. Presence: Teleoperators & Virtual Environments, 2005, 14(2): 161-174.

[10]. 付倩, 沈俊辰, 张茜颖,等. 面向手语自动翻译的基于 Kinect 的手势识别[J]. 北京师范大学学报(自然科学版), 2013(6):586-592.

[11]. Takahashi T, Kishino F. Hand gesture coding based on experiments using a hand gesture interface device[J]. Acm Sigchi Bulletin, 1991, 23(2): 67-74.

[12]. Yoruk E , Konukoglu E , Sankur B , et al. Shape-based hand recognition[J]. IEEE Transactions on Image Processing, 2006, 15(7):1803.1815.

[13]. Roy; S . Hand recognition system[J]. Journal of the Acoustical Society of America, 2006, 119(3):1310.

[14]. Microsoft Kinect [EB].http:// www.kinect.com/, 2013.03.05.

[15]. Hongo H, Ohya M, Yasumoto M, et al. Focus of attention for face and hand gesture recognition using multiple cameras[C]//Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580). IEEE, 2000: 156-161.

[16].Devineau G, Moutarde F, Xi W, et al. Deep learning for hand gesture recognition on skeletal data[C]//2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). IEEE, 2018: 106-113.

[17].张良国,吴江琴, 高文,等. 基于 Hausdorff 距离的手势识别[J]. 中国图象图形学报, 2002, 7(11):1144.1150.

[18].李文生, 姚琼, 邓春健. 粒子群优化神经网络在动态手势识别中的应用[J]. 计算机工程与科学, 2011, 33(5):74.79.

[19].刘杨俊武. 实时动态手势识别技术的研究与应用[D].南京邮电大学,2017.

[20].刘耀杰, 舒畅, 傅志中. 基于骨架运算的动态手势定位及识别算法[J]. 电子科技, 2014, 27(3):7-11.

[21].杜新怡. 基于 Kinect 的手势识别与三指灵巧手交互研究[D].兰州理工大学,2018.

[22].吴晴. 基于改进的 CNN 和 SVM 手势识别算法研究[D].江西农业大学,2018.

[23].常亮, 邓小明, 周明全,等. 图像理解中的卷积神经网络[J]. 自动化学报, 2016, 42(9):1300-1312.

[24].倪志伟. BP 网络中激活函数的深入研究[J]. 安徽大学学报(自科版), 1997(3):48-51.

[25].陈耀丹, 王连明. 基于卷积神经网络的人脸识别方法[J]. 东北师大学报：自然科学版, 2016, 48(2):70-76.

[26].He K, Zhang X, Ren S, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1026-1034.

[27].Fisher D H. Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)[C]// Fourteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc. 1997.

[28].LeCun Y, Boser B E, Denker J S, et al. Handwritten digit recognition with a back-propagation network[C]//Advances in neural information processing systems. 1990: 396-404.

[29].Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation[C]//Proceedings of the IEEE international conference on computer vision. 2015: 1520-1528.

[30].Shi J, Malik J. Normalized cuts and image segmentation[J]. Departmental Papers (CIS), 2000: 107.

[31].周祥全, 张津. 深层网络中的梯度消失现象[J]. 科技展望, 2017(27).

[32].Shah A, Kadam E, Shah H, et al. Deep residual networks with exponential linear unit[J]. arXiv preprint arXiv:1604.04112, 2016.

[33].Sabour S, Frosst N, Hinton G E. Dynamic routing between capsules[C]//Advances in neural information processing systems. 2017: 3856-3866.

[34].Geoffrey Hinton, Sara Sabour, Nicholas Frosst. Matrix capsules with em routing[J].International Conference on Learning Representations. 2018.

[35].Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation[C]//International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015: 234-241.

[36].Abadi M, Barham P, Chen J, et al. Tensorflow: A system for large-scale machine learning[C]//12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). 2016: 265-283.

[37].李抵非, 田地, 胡雄伟. 基于分布式内存计算的深度学习方法[J]. 吉林大学学报(工学版), 2015, 45(3):921-925.

[38].Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C]//Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014: 675-678.

[39].Miao Y, Gowayyed M, Metze F. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding[C]//2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). IEEE, 2015: 167-174.

[40].Gers F A, Schmidhuber E. LSTM recurrent networks learn simple context-free and context-sensitive languages[J]. IEEE Transactions on Neural Networks, 2001, 12(6): 1333-1340.

[41].孙吉贵, 刘杰, 赵连宇. 聚类算法研究[J]. 软件学报, 2008, 19(1):48-61.

[42].Kodym O, Španěl M, Herout A. Segmentation of Head and Neck Organs at Risk Using CNN with Batch Dice Loss[C]//German Conference on Pattern Recognition. Springer, Cham, 2018: 105-114.

[43].Creswell A, Arulkumaran K, Bharath A A. On denoising autoencoders trained to minimise binary cross-entropy[J]. arXiv preprint arXiv:1708.08487, 2017.

[44].Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.

[45].Russell B C, Torralba A, Murphy K P, et al. LabelMe: a database and web-based tool for image annotation[J]. International journal of computer vision, 2008, 77(1-3): 157-173.

[46].曾文锋, 李树山, 王江安. 基于仿射变换模型的图像配准中的平移、旋转和缩放[J]. 红外与激光工程, 2001, 30(1):18-20.

[47].任莎莎,郎文辉.基于 K-GMM 算法的 SAR 海冰图像分类[J].地理与地理信息科学,2018,34(05):42.48.

[48].王民,张鑫,负卫国,卫铭斐,王静.基于核模糊 C-均值和 EM 混合聚类算法的遥感图像分割[J].液晶与显示,2017,32(12):999-1005.

[49].高尚. 模拟退火算法中的退火策略研究[J]. 航空计算技术, 2002, 32(4):20-22.

[50].Barczak A L C, Reyes N H , Abastillas M , et al. A New 2D Static Hand Gesture Colour Image Dataset for ASL Gestures[J]. Massey University, 2011.

[51].Pietrandrea P. Iconicity and arbitrariness in Italian sign language[J]. Sign Language Studies, 2002, 2(3): 296-321.

[52].Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.

[53].Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.

[54].Bewley A, Ge Z, Ott L, et al. Simple online and realtime tracking[C]//2016 IEEE International

Conference on Image Processing (ICIP). IEEE, 2016: 3464-3468.

[55]. Held D, Thrun S, Savarese S. Learning to track at 100 fps with deep regression networks[C]//European Conference on Computer Vision. Springer, Cham, 2016: 749-765.

[56]. Bertinetto L, Valmadre J, Henriques J F, et al. Fully-convolutional siamese networks for object tracking[C]//European conference on computer vision. Springer, Cham, 2016: 850-865.

[57]. Valmadre J, Bertinetto L, Henriques J, et al. End-to-end representation learning for correlation filter based tracking[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017: 2805-2813.

[58]. Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.

攻读硕士学位期间的科研成果

一、攻读硕士学位期间发表的论文：

[1]**莫伟珑，**罗晓曙，钟叶秀，蒋文杰.Image recognition using convolutional neural network combined with ensemble learning algorithm.第四届智能计算与信号处理国际学术会议（ICSP 2019），已录用（第一作者）

二、攻读硕士学位期间申请的专利:

[1]**莫伟珑**，罗晓曙，赵书林.一种基于改进胶囊网络与算法的手势图像分割与识别方法. 发明专利：201910130815.4，已受理（第一作者）

致 谢

岁月如梭，弹指即去，转眼三年的硕士生涯即将结束，在论文即将完成之际，我要感谢硕士期间给予我巨大帮助的老师、同学、家人和朋友们，在此向你们表示最真挚的感谢。

我特别感谢我的导师罗晓曙教授，他是一位平易近人、严谨求实、认真负责的老师，他在项目开发中的指导和意见以及在图像处理理论分析和算法设计中给予的指导，给了我很大的帮助和启发。在整个研究生学习期间和后续论文、专利的完成都离不开罗老师悉心和中肯的指导，不管是在严谨的科研学术作风，还是求实的治学态度，都在无时无刻的影响着我，他是学业上的导师，更是人生的指路人因此，在这里我由衷地感谢罗老师那辛勤的付出。

在这里对广西科技重大专项基金项目的资助表示感谢（合同编号：桂科 AA18118004），为我的实验提供了良好的实验条件。

感谢三年来陪伴我成长的朋友、我们共同学习、共同理解、共同进步，我相信这是硕士期间最难忘的时光，同大家朝夕相处的时光让我收获了友谊与快乐，学会了宽容与感恩。

感谢实验室的师兄师姐、同门和师弟师妹们，和大家在一起进行学习成果的交流是我读研生涯中一件幸运的事情，大家一起相互团结，互相学习，营造了实验室温馨和谐的学术氛围，使我在实验室的学术学习和生活充满乐趣，祝愿大家一切顺利，前程似锦。

最后非常感谢我的家人，感谢你们对我无私的关爱和支持，为我完成学业提供巨大的动力，祝福你们身体健康，万事如意，我将载着满满的爱不断前进，积极生活，不负期望。

论文独创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下进行的研究工作及取得的成果。除文中已经注明引用的内容外，本论文不含其他个人或其他机构已经发表或撰写过的研究成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

研究生签名：莫伟珑 日期：2019.6.14

论文使用授权声明

本人完全了解广西师范大学有关保留、使用学位论文的规定。本人授权广西师范大学拥有学位论文的部分使用权，即：学校有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文；学校有权向国家有关部门或机构送交学位论文的复印件和电子版。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权广西师范大学学位办办理。

本学位论文属于：

□保密，在　　年解密后适用授权。

☑不保密。

论文作者签名：莫伟珑 日期：2019.6.14

指导教师签名：罗晓曙 日期：2019.6.14

作者联系电话：_____ 电子邮箱：_____