摘要

指纹识别技术是当前十分流行并且飞速发展的身份认证技术，然而近些年假指纹攻击已经成为了使用指纹识别系统的严重问题。假指纹攻击指的是使用人工制造的假指纹进行指纹识别并达到与真指纹相同的效果来进行破解。因此针对这种假指纹攻击，假指纹检测技术应运而生。其中光学指纹，即使用光学指纹采集仪采集的指纹图像的防伪检测又是难点。

本文提出了一个基于卷积神经网络的指纹防伪算法，网络中的核心部分为三个Inception模块。本文所使用的训练数据来源于LivDet 2019，经过划分得训练集163840张，验证集16384张，测试集480张图像，其中真指纹与四种不同材料假指纹的图像比例为4：1：1：1：1。编程语言为Python，采用了Keras和TensorFlow深度学习框架，并且使用了Cuda框架和Cudnn库对训练进行了GPU加速。

经过30次完整迭代，76800次权值更新训练得到的网络权值在测试集上进行了测试，结果为：总准确率96.875%，拒真率1.666%，认假率4.583%，损失为0.2455。在训练集上的准确率为99.91%，损失为0.0031，在验证集上准确率为99.98%，损失为0.0011。网络最优权值出现在第19次迭代结束。表明本文所提出的算法具备良好的准确率，同时拒真率与认假率也同时保持在一个较小的水平。

**关键词：**假指纹检测，光学指纹，卷积神经网络，Inception

Abstract

Fingerprint recognition technology is a very popular and rapidly developing identity authentication technology. However, in recent years, fake fingerprint attacks have become a serious problem in the use of fingerprint recognition systems. Fake fingerprint attack refers to the use of artificially manufactured fake fingerprints for fingerprint recognition and the same effect as live fingerprints for cracking. Therefore, for this fake fingerprint attack, fake fingerprint detection technology came into being. Among them, the optical fingerprint that is the anti-counterfeiting detection of the fingerprint image collected by the optical fingerprint collecting instrument is difficult.

This thesis proposes a fingerprint anti-counterfeiting algorithm based on convolutional neural network. The core part of the network is three Inception modules. The training data used in this thesis is from LivDet 2019. The training set is 163840, the validation set is 16384, and the test set is 480 images. The ratio of live fingerprint to four different material fake fingerprints is 4:1:1:1:1. The programming language is Python, using the Keras and TensorFlow deep learning framework, and using the Cuda framework and the Cudnn library for GPU acceleration.

After 30 complete iterations, the network weights obtained from 76800 weights updating training were tested on the test set. The results showed that the total accuracy rate was 96.875%, the false rejection rate(frr) was 1.666%, the false acceptance rate(far) was 4.583%, and the loss was 0.2455. The accuracy rate on the training set is 99.91%, the loss is 0.0031, the accuracy on the verification set is 99.98%, and the loss is 0.0011. The network optimal weight appears at the end of the 19th iteration. The experimental results are compared with the results of the LivDet 2019 competition algorithm. The results show that the proposed algorithm has good accuracy, while the false rejection rate(frr) and the false acceptance rate(far) are kept at a small level.

**Keywords：**fake fingerprint detection, optical fingerprint, convolutional neural network, Inception

**目录**

[摘要 II](#_Toc11237769)

[Abstract I](#_Toc11237770)

[第一章 绪论 1](#_Toc11237771)

[1.1 研究背景 1](#_Toc11237772)

[1.2 国内外研究发展现状 1](#_Toc11237773)

[1.2.1 国内研究现状 1](#_Toc11237774)

[1.2.2 国外研究现状 4](#_Toc11237775)

[1.3 项目的研究任务 6](#_Toc11237776)

[1.3.1 研究目标 6](#_Toc11237777)

[1.3.2 研究的基本内容 6](#_Toc11237778)

[1.4 研究意义 8](#_Toc11237779)

[1.5 本文的主要工作 8](#_Toc11237780)

[1.6 本文的组织结构 9](#_Toc11237781)

[1.7 本章小结 9](#_Toc11237782)

[第二章 工具与技术路线 10](#_Toc11237783)

[2.1 开发语言 10](#_Toc11237784)

[2.1.1 Python 10](#_Toc11237785)

[2.1.2 Anaconda 10](#_Toc11237786)

[2.2 开发框架 10](#_Toc11237787)

[2.2.1 TensorFlow 11](#_Toc11237788)

[2.2.2 Keras 11](#_Toc11237789)

[2.3 开发环境 11](#_Toc11237790)

[2.3.1 电脑环境 11](#_Toc11237791)

[2.3.2 Python环境 11](#_Toc11237792)

[2.3.3 GPU加速 12](#_Toc11237793)

[2.3.4 训练可视化 12](#_Toc11237794)

[2.4 本章小结 12](#_Toc11237795)

[第三章 卷积神经网络简介 13](#_Toc11237796)

[3.1 概念简介 13](#_Toc11237797)

[3.1.1 机器学习 13](#_Toc11237798)

[3.1.2 深度学习 13](#_Toc11237799)

[3.1.3 卷积神经网络 14](#_Toc11237800)

[3.2 基本部件 14](#_Toc11237801)

[3.2.1 卷积层 14](#_Toc11237802)

[3.2.2 池化层 16](#_Toc11237803)

[3.2.3 激活层 17](#_Toc11237804)

[3.2.4 全连接层 19](#_Toc11237805)

[3.2.5 目标函数 19](#_Toc11237806)

[3.2.6 批标准化层 19](#_Toc11237807)

[3.3 经典网络 20](#_Toc11237808)

[3.3.1 LeNet 20](#_Toc11237809)

[3.3.2 AlexNet 20](#_Toc11237810)

[3.3.3 VGGNet 21](#_Toc11237811)

[3.3.4 GoogLeNet 22](#_Toc11237812)

[3.3.5 ResNet 22](#_Toc11237813)

[3.4 本章小结 23](#_Toc11237814)

[第四章 指纹防伪算法的卷积神经网络设计 24](#_Toc11237815)

[4.1 数据集划分 24](#_Toc11237816)

[4.1.1 训练集 25](#_Toc11237817)

[4.1.2 验证集 25](#_Toc11237818)

[4.1.3 测试集 25](#_Toc11237819)

[4.2 Inception模块 26](#_Toc11237820)

[4.2.1 Inception概念 26](#_Toc11237821)

[4.2.2 Inception结构 26](#_Toc11237822)

[4.2.3 瓶颈层 27](#_Toc11237823)

[4.3 卷积神经网络结构 27](#_Toc11237824)

[4.3.1 网络结构 27](#_Toc11237825)

[4.3.2 尺寸计算 31](#_Toc11237826)

[4.3.3 损失函数 32](#_Toc11237827)

[4.3.4 训练过程 32](#_Toc11237828)

[4.4 本章小结 33](#_Toc11237829)

[第五章 指纹防伪算法的卷积神经网络训练优化 34](#_Toc11237830)

[5.1 CPU性能分析 34](#_Toc11237831)

[5.2 配置GPU加速 35](#_Toc11237832)

[5.2.1 显卡驱动更新 35](#_Toc11237833)

[5.2.2 Cuda与Cudnn 36](#_Toc11237834)

[5.3 训练性能比较 36](#_Toc11237835)

[5.4 本章小结 37](#_Toc11237836)

[第六章 实验结果分析 38](#_Toc11237837)

[6.1 实验结果 38](#_Toc11237838)

[6.2 实验结果分析 40](#_Toc11237839)

[6.3 本章小结 40](#_Toc11237840)

[第七章 总结 41](#_Toc11237841)

[7.1 完成的工作 41](#_Toc11237842)

[7.2 存在的问题及下一步工作 41](#_Toc11237843)

[7.2.1 存在的问题 41](#_Toc11237844)

[7.2.2 下一步工作 41](#_Toc11237845)

[参考文献 43](#_Toc11237846)

[致谢 47](#_Toc11237847)

[附录 48](#_Toc11237848)

[附录1 毕业设计文献综述 48](#_Toc11237849)

[附件2 毕业设计开题报告 48](#_Toc11237850)

[附件3 毕业设计外文翻译（中文译文与外文原文） 48](#_Toc11237851)

图目录

图3‑1 水平梯度卷积核 15

图3‑2 卷积操作示意图 15

图3‑3 经典卷积核示意图 16

图3‑4 平均池化与最大池化示意图 17

图3‑5 Sigmoid激活函数图象 18

图3‑6 Tanh激活函数图象 18

图3‑7 Relu激活函数图象 19

图3‑8 LeNet网络结构图 20

图3‑9 AlexNet网络结构图 21

图3‑10 VGGNet网络结构图 22

图3‑11 残差模块结构图 23

图4‑1 真指纹图像 24

图4‑2 假指纹图像 25

图4‑3 Inception模块示意图 26

图4‑4 Inception\_1模块示意图 28

图4‑5 Inception\_2模块示意图 29

图4‑6 Inception\_3模块示意图 29

图4‑7 完整网络结构示意图 30

图4‑8 数据变化流程图 31

图4‑9 训练过程截图 32

图5‑1 CPU训练截图 34

图5‑2 CPU训练时CPU使用率 34

图5‑3 CPU训练时GPU使用率 35

图5‑4 NVIDIA显卡驱动版本 36

图5‑5 Cuda框架与Cudnn加速库 36

图5‑6 GPU训练时CPU使用率 36

图5‑7 GPU训练时GPU使用率 37

图5‑8 GPU训练截图 37

图6‑1 准确率折线图 39

图6‑2 损失值折线图 39

图6‑3 准确率与损失值折线图 39

表目录

表6‑1训练中部分迭代结果 38

表6‑2算法在测试集上表现 38

# 第一章 绪论

## **研究背景**

指纹识别系统由于其可靠的性能和可用性而被广泛用于用户认证，与其他生物识别系统相比。此外，由于低成本采集设备的发展，这些系统已被广泛用于法医和商业应用，例如刑事调查，电子商务和访问控制。虽然人们对于指纹识别系统的研究兴趣一直在增加，但一些研究人员的研究结果显示这些系统可能容易受到某些威胁。有人提出了指纹系统可能受到的八种潜在攻击方式，并提出了一些保护它们免受这些攻击的一般指导原则。虽然从安全角度来看，所有这些攻击都是严重的，但根据一些研究人员的说法，其中使用假指纹的攻击是最严重的。

因为根据用户的指纹而制造出的假指纹有机会让指纹识别仪器判定其为真，从而进入指纹识别系统，达到破解目的。因此，对于假指纹防伪技术的研究是有意义且迫不及待的。

## **国内外研究发展现状**

### 1.2.1 国内研究现状

1. 方等人在论文中提出了一种基于小波分析改进之后的假指纹检测算法。他们在原始的小波分析处理后，使用去噪图像与残留噪声图像以计算局部二值直方图，并从中提取特征指纹。他们认为这种方法所得特征与传统小波分析相比能够保留更多的纹理信息。之后使用SVM-RFE方法进行特征选择并使用支持向量机进行分类。该方法在第二届与第三届假指纹检测竞赛的数据库上进行测试。分类准确率分别为88.75%和89.37%，第二届优胜者为74.41%，第三届优胜者为86.63%[1]。
2. 陈等人对基于方向场置信度的分割算法进行改进，他们结合了区域跟踪与腐蚀膨胀算法。通过结合一阶对称复数滤波，验证基于传统Poincare法所得奇异点的准确性，在此基础上提出了基于“主中心点”脊线跟踪的指纹分类方法，该种方法可以用于确定指纹纹型[2]。
3. 张等人认为传统的基于细节特征点的指纹识别算法大多适用于采集面积较大的指纹，而对于智能手机端的小采集面积指纹图像的支持并不好，他们提出了一种基于深度学习的小面积指纹匹配方法。该算法的主要步骤为：首先提取指纹图像的细节特征点；其次搜索和标定感兴趣区域（ROI）；然后构建改进后的基于残差结构的轻量级深度神经网络；最后制定手机端的注册匹配策略实现小面积的指纹匹配。实验结果表明,提出方法在公开库FVCDB1与自建数据库上的等错率(EER)分别仅为0.50%与0.58%，能够更好地满足智能手机端的需求[3]。
4. 李等人认为指纹检测以其令人满意的性能在许多安全应用领域中得到了应用。近年来，基于二值化统计图像特征（BSIF）的活度指纹纹理检测技术得到了广泛的应用。然而，他们发现指纹边缘的像素比中间的像素所含有的信息更少。因此，他们提出了一种经过优化的BSIF的真假指纹检测方法。根据高斯分布，与边缘的像素相比，中间的像素会更多地被采样。改进BSIF的实验结果证明，改进BSIF方法比传统方法具有更高的准确率。这可能意味着中间的像素比边缘的像素更能提供生动性检测的信息[4]。
5. 吴等人在论文中提出，干湿手指导致指纹质量差，对指纹识别和匹配有影响。基于山脊、山谷、细节或毛孔特征的识别方法受皮肤状况的影响。提出了一种利用脊线特征对不同分辨率图像进行干指纹检测的新方法。干燥的指纹有模糊的毛孔和不连续和支离破碎的脊。因此，他们采用的检测特征是脊连续性、脊破碎和脊谷比。在不同的图像分辨率下可以清晰地观察到这些特征，因此他们提出的方法可以在500-1200 dpi下工作。其实验采用NASIC数据库（1200dpi）和FVC2002 DB1（500dpi），SVM分类精度分别为99.00%和99.09%[5]。
6. 金等人在分析指纹图像曲率图的基础上，提出了一种新的指纹奇异点检测算法。利用曲率图和方向场定义和计算指纹图像参考方向。由于每个指纹图像都有一个且仅在FIRD上，因此在进行指纹匹配时，FIRD可以用作参考帧方向。在FVC2002中对DB3的训练数据库B（training database）进行了一系列的指纹图像公共域采集实验，对他们所提出的方法进行了评价。使用FIRD参考帧的处理方法经过实现的验证，被证明了具有较高的准确度与性能[6]。
7. 贾等人提出在众多的指纹活力检测方法中，局部二值模式（LBP）被认为是最有效的检测方法之一。但原有的LBP算子具有空间支撑面积小的局限性。因此，一种基于MSLBP的欺骗指纹检测方法被他们所提出。一般来说，MSLBP有两种不同的实现方式。实现了两种类型的MSLBP。每个MSLBP都与一组过滤器组合在一起。这样，LBP圆中的每个样本都可以从大面积而不是单个像素收集强度信息。实验结果表明，这两种类型的MSLBP在伪指纹检测中都是有效和优越的[7]。
8. 邵等人提出纳米荧光粉在潜在指纹成像中有着巨大的应用前景。采用微乳液法合成了YVO4:Bi3+、Eu3+等粒径分布～22\_nm的荧光粉。由于荧光发射强、粒径小，采用荧光粉作为指纹荧光标签，显示出清晰、清晰的指纹乳头状脊线，具有高灵敏度、高对比度、无/低背景干扰等特点，证明YVO4:0.1bi3+、0.05eu3+是一种很有前景的潜在指纹荧光L[8]。
9. 葛等人提出了一种基于指尖检测的手势识别方法。首先，通过microsoft公司生产的Kinect设备获取人手的深度图像，结合阈值分割和颜色空间完成手部分割。最后，设置了重心距离的指尖检测算法，设置了掌心到指尖的距离和俯角，完成了指尖标记点的识别。实验结果经100个实验验证，识别率为97%[9]。
10. 陈等人提出了一种指纹分割算法和块特征评价准则。分割使用三个块特征：块簇度、块平均信息和块方差。针对每一块的分类问题，训练了一个最佳线性分类器，并采用了最小误分类样本数准则。形态学已被应用于后处理，以减少分类错误的数量。FVC2002数据库进行的测试表面，只有2.45%的块被错误分类，而后处理进一步降低了这一比例。实验表明，该分割方法能够很好地去除噪声背景下的伪指纹特征[10]。
11. 高等人报道了一种用核壳结构的CdTe@SiO2量子点（QDS）作为荧光标记的简易粉末刷涂技术。以NH2NH2·H2O为调节剂和稳定剂，采用简单的溶液法制备了核壳结构的CdTe@SiO2量子点，并探讨了其在改善潜在指纹检测中的应用。得到的CdTe@SiO2量子点呈球形，具有明确的核壳结构，根据pH调节器和稳定剂的类型封装不同数量的量子点。可实际应用于潜在指纹检测。与传统荧光粉、银粉等相比，CdTe@SiO2 QD粉对多种物体表面潜在指纹的检测效果有了很大提高。CdTe@SiO2量子点的合成方法简单、廉价、易于大规模生产，在指纹检测的实际应用中具有许多优势[11]。
12. 唐等人在论文中提出了一种改进的基于连续方向场的指纹图像奇异性检测方法，以提高奇异性的定位精度和可靠性。首先，通过计算庞加莱指数来检测可能含有奇点的块。然后，在块图像中检测到奇异点。实验表明这种基于连续方向场的方法具有很好的扩展性[12]。

### 1.2.2 国外研究现状

1)约塞普帕克等人发现大多数现有的伪指纹检测方法要么需要额外的传感器，要么需要复杂的计算。他们在论文中提出了一种结合六个简单统计矩特征进行检测的伪指纹检测方法。六个统计矩是指偏差、方差、偏度、峰度、超锐度和超平面度。另外还使用了平均亮度、标准偏差和差分特性[23]。

2)鲍勃霍姆等人发现通过学习真假指纹图像建立的基于静态特征的伪指纹检测系统难以准确地分类新的或者未经训练的假指纹类型。为了解决这个问题，现有的假指纹检测仪应该逐步训练新类型的假指纹。为了检测一种新型的假指纹，一位专门针对这种新型假指纹进行培训的新专家被集成到了欺骗检测器中。每个专家都由多个支持向量机（SVM）组成，这些支持向量机由一个增量学习算法（learn++.nc）应用，其中每个SVM都采用三种纹理特征中的一种进行欺骗检测[24]。

3)尹申英等人在论文中提出，随着生物识别服务环境向开放系统的扩展，侵犯隐私和非法占用信息的威胁都在增加。然而，虽然证书或智能卡可以很容易地取消和重新发行，但是如果发现丢失，没有办法恢复个人的独特生物特征信息后，将会产生安全漏洞。随着人们认识到这一威胁因素可能会干扰大规模的公务员业务的临近实施，但迄今为止还没有客观的绩效评估方法的报道。因此，他们提出了一种评价当前备受关注的伪造指纹检测技术客观性能的方法[25]。

4)拉姆普拉卡什等人提出指纹的识别已广泛应用于各种应用领域。然而利用指纹的人工复制可以达到欺骗传感器的效果。在这种情况下，指纹活度检测确保了真实合法指纹的实际存在，而不是虚假的自制合成样本。一种基于静态软件的指纹活度检测方法在他们的论文中被提出。这些局部质量特征构成了13维特征向量。该系统在2009年LivDet竞赛的公开数据集上进行了测试，并取得了优越的成绩[26]。

5)巴纳卡等人介绍了一种用于指纹生物识别系统的图像处理技术，用于检测被认证人的活度。该系统基于小波分析，在这项工作中，活体指纹图像和假指纹图像之间的区别已经被利用。指纹图像的轮廓分析给出了相同的结果。基于小波变换的指纹图像分解方法是利用达贝契小波分解到四个层次，并对近似图像进行轮廓分析。对50幅活体和50幅硅伪指纹图像进行了测试，绘制了接收端工作特性曲线，得到了0.078和0.147的等误差率。最后，将这两种方法级联起来进行活动性检测。成功率为100%[27]。

6)安德鲁M.麦克唐纳等人提出氧化锌是一种荧光粉末，用于检测非多孔表面上的指纹。将氧化锌粉末应用于玻璃、聚乙烯、铝箔等无孔表面上的新鲜和老化指纹。在长波紫外光照射下，氧化锌能清晰地显示潜在指纹的荧光[28]。

7)穆罕默德等人说明自动指纹识别系统（AFISS）的成功很大程度上取决于指纹上参考点的准确检测。利用方向场和边缘检测算法确定奇异点的第一近似位置。然后，为了找到这些点的精确位置，引入了一种新的二次滤波器。通过组合这些项目，将获得参考点。这可以应用于所有类别的指纹，尤其是有问题的拱形指纹[29]。

8)罗伯特等人提出了一组28种基于统计、梯度和光谱密度的特征，用于使用低分辨率扫描进行潜在指纹检测。在这项工作中，一个彩色白光（CWL）传感器被用于图像采集。实验评价是基于扫描1680个潜在指纹在三个不同的表面。结果表明，该方法在具有均匀反射特性的平面非吸收表面上具有很好的性能，最佳检测率为2.51%。另一方面，困难来自具有非均匀/可预测反射特性的表面[30]。

9)鲍迪塞拉的工作提出了一种新的方法，以确保指纹扫描仪不出现伪造的指纹。气味传感器（电子鼻）用于对气味信号进行采样，特殊算法允许将手指皮肤气味与其他材料（如乳胶、硅胶或明胶）区别开来，这些材料通常用于伪造指纹[31]。

10)埃林斯加德指出自动指纹识别系统（AFIS）的成功已经导致越来越多的事件发生，人们为了逃避身份识别而改变自己的指纹。尤其是在边境口岸，指纹要与观察名单进行比较。文章讨论了自动检测更改指纹的方法。该方法基于对指纹图像两种不同的局部特征的分析。第一个分析识别像素方向的不规则性，这些不规则性与奇异点具有相似的特征。第二种分析比较了覆盖局部但比第一种分析面积大的细节点方向。在每个分析中创建一个全局密度图，以确定分析差异的分布。实验结果表明，该方法的性能完全可以与目前最先进的方法相媲美[32]。

## 项目的研究任务

### 1.3.1 研究目标

指纹识别因其有效性与可靠性成为目前主流的生物识别技术与身份验证技术。然而指纹识别技术并不是绝对安全的，它也有可能受到各类攻击，其中以假指纹攻击的危害与后果最为严重。所以本研究致力于解决光学指纹的防伪问题。目标是能够设计与实现快速、可靠、准确的假指纹检测算法。进一步提高指纹识别技术的安全性。

### 1.3.2 研究的基本内容

之前出现的假指纹检测的方法可以被分类为硬件与软件方法。使用硬件的方法主要是通过额外的硬件采集更多的人体生物信息(假指纹所不具备)来实现假指纹检测。而硬件手段往往要承担更高的成本，所以软件方法才是目前研究的主流。以往的假指纹算法大部分是比较传统的算法或者使用支持向量机(SVM)的算法。本研究的目标是通过目前流行的卷积神经网络(CNN)这一方法用于光学指纹防伪检测算法。

(1)数据集的预处理

对于卷积神经网络的训练而言，需要训练集、验证集、测试集三个数据集来完成全部过程。其中训练集、验证集用于训练过程，在训练集上进行训练，并在每一次迭代结束通过验证集验证准确率。而测试集用于最后的准确率测试。测试集与验证集的区别在于，验证集对于训练结果不是无偏的，而测试集是无偏的，也就是说训练结果不受到测试集的影响，所以测试集的准确率才是更好的评价标准。所以在开始训练前需要将数据集划分成适合的训练集、验证集、测试集。

(2)卷积神经网络的选取

基于不同的方法，搭建的卷积神经网络是不同的，在项目中需要选取较为合适，与性能好的方法、模块进行卷积神经网络的搭建。

(3)卷积神经网络的结构设计

卷积神经网络的整体结构设计，会对训练结果产生极大的影响，有时甚至是最关键的因素，所以网络的结构设计极为重要。本研究需要设计合适的网络结构，包括卷积神经网络部件的选取，目标函数设定，具体的数值大小，来使得训练得到的结果对于假指纹检测问题具有较好的判断效果。

(4)训练调参

对于神经网络的训练，调参几乎是必不可少的步骤，通过调参来使得自己的网络训练在研究问题上具有更好的表现。在这个过程中，需要注意不要让测试集的数据混入其中，因为我们需要保证测试集的数据无偏性，如果有测试数据混入，可能会导致结果的假优，这是因为在训练过程中网络会向着能够更好处理验证集情况的方向靠近。

(5)结果分析

对于训练好的卷积神经网络的结果，进行结果分析，并与LivDet2019竞赛上的结果进行比较、分析，用以评判该算法的准确性是否满足要求。

## 研究意义

指纹识别系统由于其可靠的性能和可用性脱颖而出，被广泛应用于用户认证系统。随着对指纹识别系统研究的增加与积累，人们发现了指纹识别系统可能会受到某些威胁，而其中又以假指纹攻击的危害性最为严重。而用于检测假指纹的方法手段，主要有硬件和软件两种。硬件方面，通过使用额外的硬件来获取人体活性指标，例如：体温，脉搏，血流量，心跳等等，然而这样的方法需要增加额外的硬件，往往会使成本大量增加，系统体积也会变得庞大。而与之对应的软件方法检测假指纹则不需要额外的硬件，是目前假指纹研究的重点。

而基于软件的方法又大致可以分为动态与静态两种，动态方法需要通过一系列的指纹图像来进行判断。而静态方法经过训练之后可以通过单一图像来判断。显然这在某种程度上降低了可靠性，但是同时也带来了更强的鲁棒性。目前与未来的研究热点应该是静态方法的研究。

尽管指纹可以伪造，但由于制作材料与真实皮肤间存在的差异，会使得采集到的指纹图像有一定差别，其具体表现在从指纹图像中提取到的特征上。例如，由于在制作过程中人造材料大分子聚集，假指纹表面粗糙度大于真指纹。假指纹检测技术就是基于真实指纹和假指纹的特定类型的可区分特征来判断指纹图像是否来自于活体指纹的技术。当前，利用活体指纹的动态行为（如脊线形变、汗液）或者静态特征（例如纹理特性、脊线频率、皮肤弹性）来检测指纹图像的活性信息的方法，由于不需要额外的硬件设备、适应性更强，受到广泛关注，具有重大的实用价值和推广意义。

## **本文的主要工作**

本文的工作主要是以下5部分：1) 介绍了假指纹检测的研究背景、发展现状、研究意义; 2)对于卷积神经网络作了一个简介，包括常见的开源框架与经典的一些网络; 3)基于卷积神经网络设计了自己的光学假指纹检测算法，用于处理假指纹检测问题; 4)由于使用CPU训练速度并不能令人满意，配置了GPU加速来提高训练速度; 5)分析了本文所得算法的性能好坏。

## **本文的组织结构**

本文总共含有7章，对于光学假指纹检测问题进行了研究，并设计与实现了基于卷积神经网络的算法，本文对该算法与实验结果进行了详细阐述。为方便阅读，特此说明全文的组织结构如下：

第一章绪论：介绍了光学假指纹检测问题的研究背景，发展现状，研究任务，研究意义，本文的主要工作与组织结构。

第二章方法与技术：列出了本文所提出的算法所使用的方法与技术路线，包括开发语言、框架、环境，并对其中主体部分的工具做了简单介绍，使得读者可以更好地了解实验环境。

第三章卷积神经网络简介：简单介绍了卷积神经网络，包括它的概念，涉及到的主流框架，基本构成，及这些部件各自的作用，与一些经典的卷积神经网络。

第四章卷积神经网络搭建：介绍了该卷积神经网络的结构，包括前期的数据集划分工作以及重点模块Inception，以及Inception模块的一部分介绍。

第五章GPU加速：介绍了在训练中，配置GPU加速的过程，这一环节可以极大提升网络的训练速度，具体提升效果视实际CPU与GPU包括其余硬件参数而定，实际上与训练的数据集可能也有一定的关系。

第六章实验结果分析：对本文提到的卷积神经网络，分析本文所提算法的性能好坏，讨论该算法的可行性。

第七章总结：对于本文中所提及的完成工作、存在的问题及下一步工作的方向做一个总结。

参考文献、致谢以及附录。

## **本章小结**

本章主要介绍了假指纹检测研究问题的背景，发展现状，本项目的研究任务与意义。并在章末部分给出了全文的主要工作与组织结构。

# 第二章 工具与技术路线

## **开发语言**

大多数人在研究机器学习问题时，会选用Python作为主语言，事实上由于Python代码的易于阅读，包含的大量第三方库与开源机器学习框架，以及很多用于处理密集计算的C/C++/Java底层库。使得Python成为最适合机器学习的语言。所以本项目由Python编写，在Python中使用了Anaconda工具进行库管理。

### 2.1.1 Python

Python是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。Python在Web和Internet开发、科学计算和统计、软件开发、后端开发、机器学习等领域具有较好的应用。Python还具有很多优点，比如设计的时候Python就遵循简单主义，所以Python的使用非常简单，可以让使用者将自己的精力集中在解决问题的思路上，而不是代码语言本身。而Python的简单也意味着学习成本低，初学者可以很顺利的掌握Python编程基础，特别是有其他语言编程基础的人，也就是易上手。丰富的库，这一点也是吸引机器学习相关人员使用Python作为主语言的重要原因之一，除了强大的标准库，Python的各种第三方库也都具备相当高的质量。

### 2.1.2 Anaconda

Anaconda是一个Python的开源包、环境管理器，也可以看作是一个Python发行版本。使用Anaconda进行包管理，十分方便，并且配合PyCharm使用，可以可视化地管理Python包及其依赖，为Python开发带来极大的方便。并且使用Anaconda相当于创建了一个虚拟的环境与系统环境隔离开来，也可以自由切换。

## **开发框架**

在Github上有许多开源的深度学习框架比如：TensorFlow、Keras、Scikit-Learn、Caffe、PyTorch、Theano、CNTK等，它们各有优劣，选择适合自己的框架有利于研究、开发。本项目中机器学习开源框架选择了Keras，基于TensorFlow的后端实现。

### 2.2.1 TensorFlow

TensorFlow说是目前最为流行的开源深度学习框架可以说是毫无问题。截至2019/5/14，在github上TensorFlow项目获得了8587个Watch，127501个Star，74601个Fork，1987个contributors。TensorFlow是一个开源的软件库，多维数据数组(tensor也叫张量)可以在其中的节点间灵活的流动(flow)，这也是其名字的来源。TensorFlow是谷歌开发的，目的是方便机器学习和深度神经网络的研究，经过时间的检验，该框架具有足够的通用性，可以应用到其他领域中。

### 2.2.2 Keras

Keras是Python中的高层神经网络的API，以TensorFlow、Theano、CNTK作为后端，可以任意选用。使用Keras可以更快的上手深度学习，并且Keras对CPU和GPU都有良好的支持，之间的切换也十分简洁，并不繁琐。目前Keras对于Python的适配版本是2.7-3.6。

## **开发环境**

### 2.3.1 电脑环境

电脑：华硕K550JX4200

操作系统：Windows8.1 中文版

CPU：Intel Core i5-4200H

GPU: Nvidia GTX950M

内存：8GB

### 2.3.2 Python环境

Python IDE：PyCharm 2018.2.4x64

Python：3.6

Keras：2.2.4

TensorFlow：1.13.1

### 2.3.3 GPU加速

Cuda：10.0.130

Cudnn：7.3.1

### 2.3.4 训练可视化

TensorBoard：1.13.1

## **本章小结**

本章介绍了本文中实现算法使用的开发语言、开发框架与开发环境及其版本。本章内容会让读者对本文描述的算法的具体实现有更清楚的认识，并且如果读者希望尝试复现本算法，我相信本章会非常有帮助。

# 第三章 卷积神经网络简介

## 概念简介

### 3.1.1 机器学习

机器学习(Machine Learning)是人工智能(Artificial Intelligence)的核心部分，如果将其看作一门学科，那么它不是单一化的，它是一门交叉学科，其中将会涉及到统计学、概率论、算法复杂度理论、凸分析等等学科。然而，在目前人们对于机器学习并没有一个完全统一的定义，有的只是不同人在不同时期给出的定义。比如说在1996年，Langley给出的定义，机器学习是一门人工智能科学，它所涉及的主要内容是人工智能，尤其在提升算法性能方向，通过经验学习的方式。在1997年，Tom Mitchell给出了他的定义，机器学习是指计算机算法能够自动提升性能通过经验学习。在2004年Alpaydin给出了自己的理解：机器学习是计算机程序通过样例数据和过往经验优化提升自己的表现。而就我自己的理解来说，机器学习的名字就足够表示其内涵，那就是让机器去学习，这也是机器学习算法区别于传统算法的地方，对于传统算法，其有效性往往来自于人类的赋予，而机器学习其有效性来自于机器（往往是各种计算机）的自我学习，人们只需要提供给它训练所需的数据，它就能得到自己理解下的算法结果，所以它是属于人工智能学科下的一门子学科[13-14]。

### 3.1.2 深度学习

深度学习(Deep Learning)是机器学习(Machine Learning)的一部分，在深度学习中的计算可以由数据的流向图表示出来，而对于这种数据流向图，深度信息是非常重要的一个指标或者说数据。在2019年3月27日，随着国际计算机学会(Association for Computing Machinery，下称ACM)的宣布，三位深度学习的创造者Yoshua Bengio, Yann LeCun, Geoffrey Hinton获得了2019年的图灵奖，这一计算机研究领域的至高奖项。Hinton的主要贡献来自于1986年的反向传播理论以及2012年对于卷积神经网络的改进。Bengio的贡献主要来自于1990年的概率模型与神经网络的结合以及2000年的高维词向量表征自然语言。Yann LeCun的贡献则是卷积神经网络以及改进了反向传播理论。目前深度学习在计算机视觉、自然语言处理、生物信息学、自动控制等领域蓬勃发展并都展现出良好的效果与实力[20-22]。

### 3.1.3 卷积神经网络

卷积神经网络(Convolutional Neural Networks，也缩写为CNN)是深度学习的一部分，也可以理解为包含卷积操作的深度神经网络。早在二十世纪八九十年代就出现了最早的卷积神经网络，他们是时间延迟网络和LeNet。如上一小节中提到的，Yann LeCun因其在卷积神经网络中的贡献获得了2019年的图灵奖。卷积层可以说是卷积神经网络的核心所在，它的原理其实说起来也并不复杂，是进行加权求和提取特征信息的过程。在卷积神经网络中最吸引人的是他的特征参数并不需要人为设置，这在以前基本依靠专家设定，这是非常需要技巧的构造性工作，也十分困难。事实证明，在这一点上人类做的并不如计算机那么出色，卷积神经网络训练出来的特征参数往往具有更好的效果，而人们在训练前只需要设定好超参数的数值即可，这将带来好几个数量级的工作简便[15-18]。

## 基本部件

### 3.2.1 卷积层

单从名字上我们也可以感受到，卷积层是卷积神经网络的核心部分，毕竟在名字中就包含了这样的字眼。实际上在CNN中的卷积操作并不复杂，虽然在一些其他领域也有着一些称之为卷积的操作，并且他们与CNN中的卷积操作并不完全一样，这里我只介绍一下在CNN这一领域中的卷积操作，或许它会与你曾经接触过的卷积略有不同，这是因为有着一些历史遗留原因，反正最终在CNN中我们的卷积操作是这样的。我们以2维数据(往往是图片)来举例，实际上无论是1维，还是3维，甚至更多的N维都是可以进行卷积的，仅仅为了便于理解选择2维数据来描述。

卷积操作实际上是一种加权求和的过程，比如说我们有一个3×3的卷积核如图3-1所示。

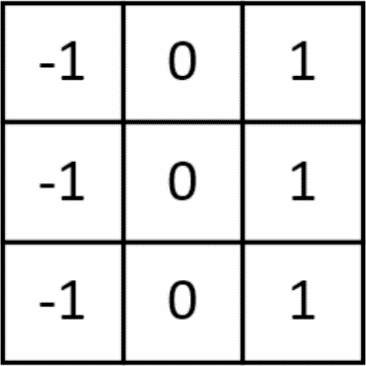


图3‑1 水平梯度卷积核

然后我们就可以使用上图的卷积核对我们的图像数据进行加权求和操作，只要将对应像素点与卷积核中数字乘起来在做求和即可，就像图3-2中所示一样，非常简单，我们对原图中每一块这样的3×3的九宫格都做这样的图片，最终原图5×5的图像，我们会得到一张3×3的新图像。

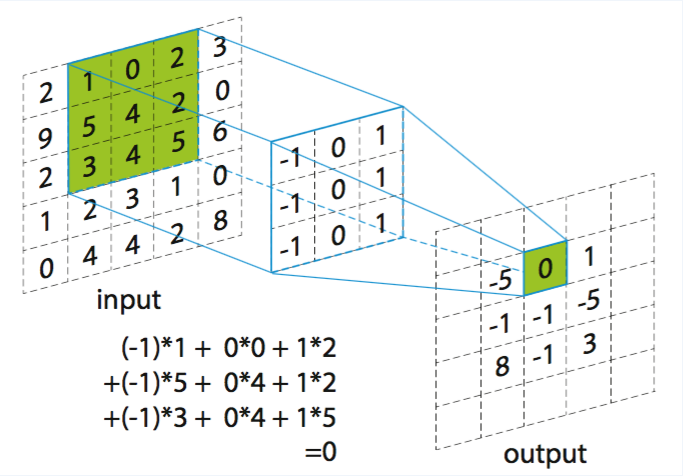


图3‑2 卷积操作示意图

在CNN中的有些时候，或者说很多时候，我们并不希望图片大小在卷积操作之后发生改变，这时候我们可以事先在原图的周围填充上一些数据。在这个例子中，如果我们先在原图周围填上一圈的0数据(之所以是0，大家都在这样做，事实上效果也确实不错)，这样就会变成从7×7变为5×5的图像。

而经过一个卷积处理的图像，通常会提取出一些特征信息，如果只是比较少的卷积次数，那么得到的就会是比较浅层的特征；但是经过较多的卷积，甚至可以得到深层次的特征。以下给出了4种非常经典的卷积核。如图3-3所示。

|  |  |
| --- | --- |
| 1. 锐化卷积核 | 1. 平滑均值卷积核 |
| 1. 垂直梯度卷积核 | 1. Soble边缘水平卷积核 |

图3‑3 经典卷积核示意图

### 3.2.2 池化层

池化层与卷积层有相似的地方，池化层中也存在池化核，但是池化核并不存在具体的参数，取而代之的是池化核的类型，比如说最大池化与平均池化。最大池化就是以池化核对应位置中的最大数值直接作为池化后的数据；平均池化则是对这些对应数字求和取平均，来作为处理后的数据。如图3-4所示，是在一个4×4的原图上分别做池化核大小为2×2的平均池化与最大池化后得到的结果。在早些时候，人们认为平均池化的效果比较好，但是目前大家往往选用最大池化。

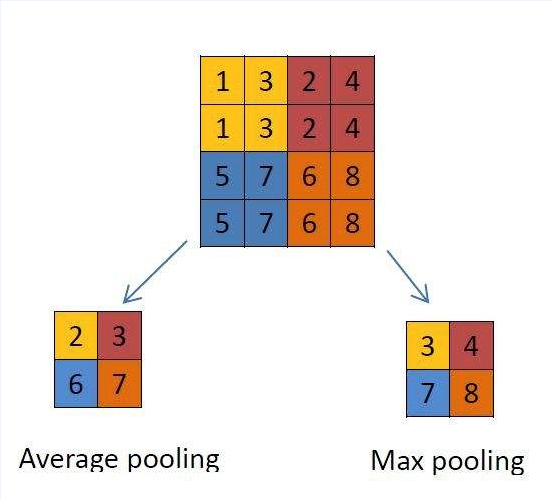


图3‑4 平均池化与最大池化示意图

### 3.2.3 激活层

所谓的激活层，其实只对原数据进行一个激活函数的映射，这是在原图上进行的，也就是说其并不会改变图片的大小。这一点与卷积层，池化层有所不同，这两者都有可能改变图片大小。可以这样理解，如果在原图中的像素值是x，经过激活层之后其值会变为f(x)，当然这些所谓图片的像素值在网络中其实代表的含义可能已经是提取出来的特征。之所以要引入激活层的概念，是想要引入非线性变换，如果不引入激活层，那么在网络中的数据变换都是线性变换的，这样的网络表现往往并不好。激活函数是多种多样的，也可以设计属于自己的激活函数，这里介绍几种经典的激活函数，如图3-5至3-7所示。

1. Sigmoid函数，如式3-1所示。

（3-1）

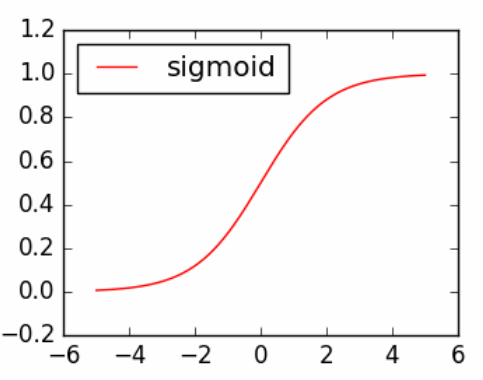


图3‑5 Sigmoid激活函数图象

1. Tanh函数，如式3-2所示。

（3-2）

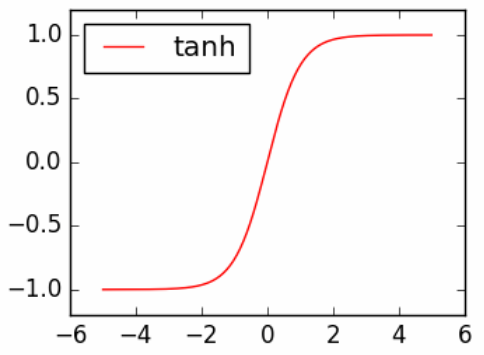


图3‑6 Tanh激活函数图象

1. Relu函数，如式3-3所示

（3-3）

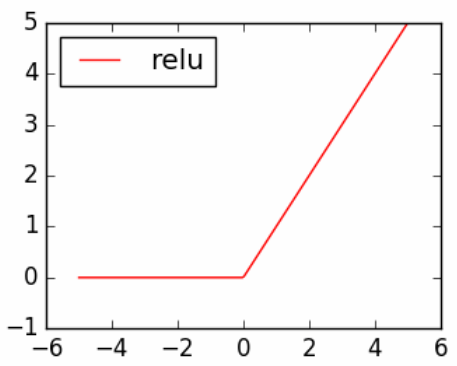


图3‑7 Relu激活函数图象

### 3.2.4 全连接层

全连接层其实就是说将前一层的所有结点与后一层的所有结点全部连接起来的一层，它的具体实现是通过矩阵乘法来实现的。它的目的就是将之前通过卷积层、池化层、激活层处理得到的深层特征与最终结果的标记数据形成一个映射。由于其所有结点相互之间都连接起来，往往也会带来大量需要训练的参数。

### 3.2.5 目标函数

目标函数的目的是为了评价，算法的准确性而设置的。也就是说对于训练数据而言，存在一个真实数据，又存在一个我们训练好的算法计算的预测数据。这两者并不一定都是相同的，当然直觉上我们觉得两者越接近越好，所以我们使用损失函数(也叫代价函数)来描述两者的接近程度。但是使用损失函数存在一个问题，那就是容易导致过拟合，所以在损失函数的基础上增加正则化来规避或减弱这种不好的情况。下面介绍几种经典的损失函数，如式3-4至3-7所示。

1. 0-1损失函数

（3-4）

1. 合页损失函数

（3-5）

1. 平方损失函数

（3-6）

1. 指数损失函数

（3-7）

### 3.2.6 批标准化层

批标准化操作的结果是使得输入信号经过变化变为均值为0，方差为1的输出信号。它主要解决的问题是有些神经网络难以训练，并且它在网络过拟合的情况下也有不错的表现。根据其提出者，一般这层会加在激活层之前，但是后续的研究者有提出，将批标准化层加在激活层之后可能会有更好的表现。

## 经典网络

### 3.3.1 LeNet

LeNet是Yann LeCun在1998年提出，最初用于解决手写数字识别的问题。LetNet可以说是现有卷积神经网络的基础，很多现在广泛使用的网络部件，在这一网络中就已经出现，比如说卷积层、池化层、全连接层，并且这种卷积层之后紧接池化层，并通过多个这样的结构提取特征，直至今天还是非常重要的思想[33]。LeNet-5的结构如图标3-8所示。

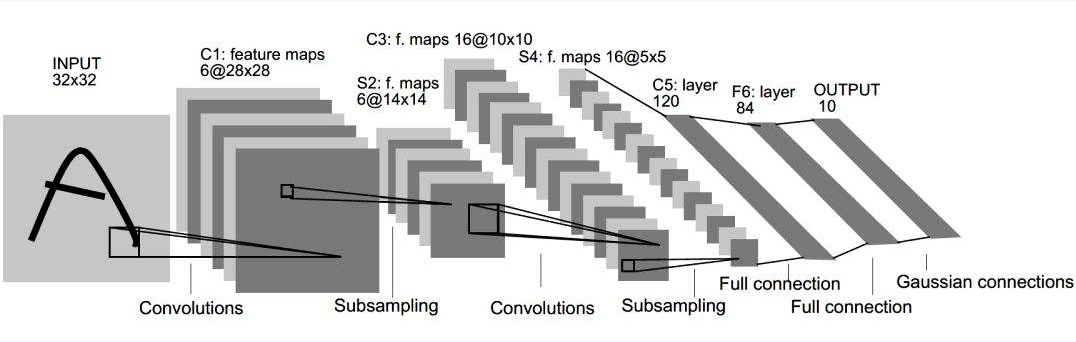


图3‑8 LeNet网络结构图

通过结构图我们可以看到，LetNet的输入是32×32×1的图像；首先是一个6个卷积核为5×5的卷积层，数据变为28×28×6；接着是池化核为2×2的池化层，数据变为14×14×6；再是16个卷积核为5×5的卷积层，数据变为10×10×16；然后是池化核为2×2的池化层，数据变为5×5×16；最后是两层全连接层以及softmax分类器。

### 3.3.2 AlexNet

如果说LetNet是卷积神经网络的开山之作，那么是AlexNet才让大众的注意力真正转移到卷积神经网络上面。在2012年的ImageNet竞赛中，AlexNet以绝对的优势一举夺冠，让大家了解了深度学习与卷积神经网络的魅力与潜力。从此，越来越多的人员加入到这一领域的研究，这一领域也一直火热至今[34]。AlexNet的结构如图3-9所示。

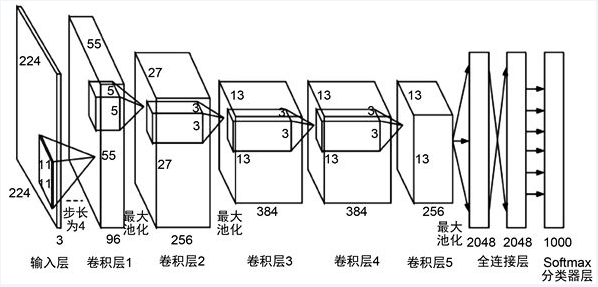


图3‑9 AlexNet网络结构图

AlexNet的主要结构与LeNet并没有区别，但是它是一个更大的网络，它总共有5层可以训练的卷积层。AlexNet的输入数据也非常庞大，为224×224×3的图像，并且第一层的卷积核大至11×11，这表示了它有非常多的参数需要被训练，也会影响到训练速度。并且在AlexNet中已经出现了ReLu激活函数与Dropout丢失层。

### 3.3.3 VGGNet

VGGNet由牛津大学VGG(Visual Geometry Group)团队提出，该网络在2014年的ImageNet竞赛上有良好的表现，获得了定位的第一名与分类的第二名[35]。VGGNet的结构如图3-10所示。VGG最显著的特点是它相比前者更深了，它总共有16个可训练的层，比之前的网络都要多，但是它却并不是一个更大的网络。实际上，它比AlexNet要小得多，这是因为它采用了更小的卷积核，整个网络内只使用了1×1和3×3的卷积核，所以它的所需训练参数并不会很夸张[19]。

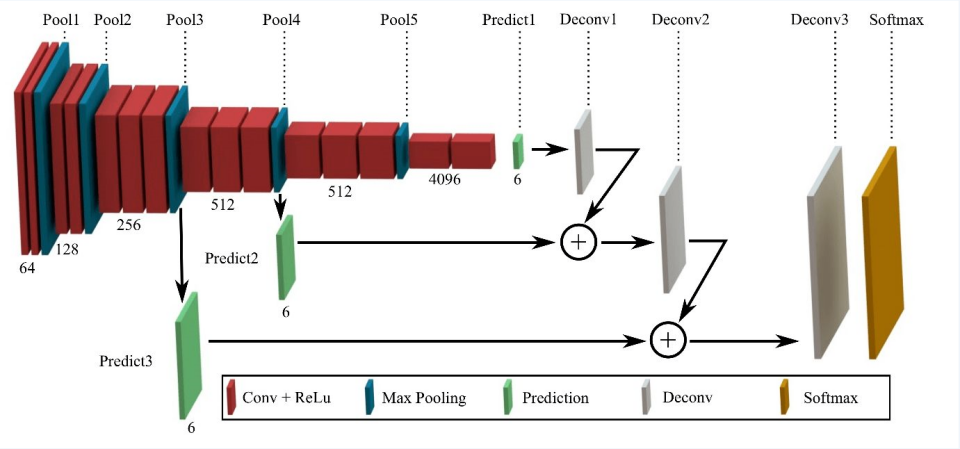


图3‑10 VGGNet网络结构图

### 3.3.4 GoogLeNet

单从名字上我们也不难看出，GoogLeNet是由Google公司所推出的，它在2014年的ImageNet竞赛中的分类任务中击败VGGNet，获得第一。与VGGNet相同的是，GoogLeNet也采用了更深的网络结构，而与VGGNet不同的是，GoogLeNet还同时拓宽了自己的网络结构[36]。引入了他们称之为Inception的结构。有关这一点会在4.2小节中作更详细的介绍。关于这个网络的命名有个有意思的地方，他们特意将自己的Google变为了GoogLe，这就使得名字后半部分变为LeNet，我认为这可能是一种故意的致敬。并且他们为自己创新的模块取名Inception，同时也是电影《盗梦空间》的英文名《Inception》，也表明了对于网络可以越走越深的信心。

### 3.3.5 ResNet

ResNet是2015年何恺明推出的网络，该网络在ISLVRC和COCO竞赛上获得冠军。ResNet主要解决的问题是随着网络层数的不断增加，准确率却没有如预期那般提升的问题，ResNet的解决方案是引入残差模块来解决[37]。残差模块的结构如图3-11所示。

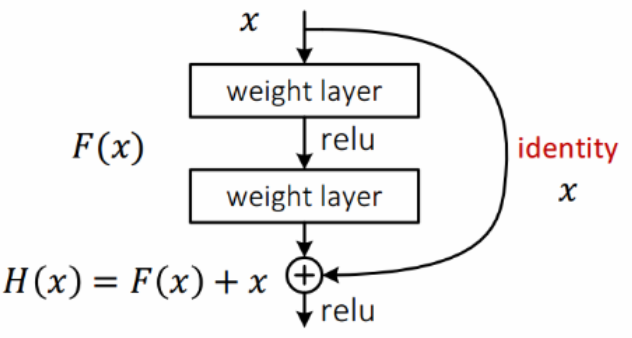


图3‑11 残差模块结构图

数据一方面经过正常的流程，另一方面通过一条捷径与后面直接相加。比如原数据为x，经过变化变为F(x)，根据以往的网络，F(x)就是后续输入，但是在残差模块中H(x)=F(x)+x才是后续输入。这条捷径也被称为跳跃连接。

## 本章小结

本章主要是对卷积神经网络做一个简单介绍，3.1节介绍了它的概念，3.2节介绍了网络中的一些基本部件，3.3节则是介绍了5种经典的卷积神经网络。通过本章，可以对卷积神经网络有一个基础的了解。

# 第四章 指纹防伪算法的卷积神经网络设计

## 数据集划分

本文使用的数据集来自于活体检测竞赛(Liveness Detection Competitions，简写为LivDet)，LivDet是一个包括了指纹、虹膜的生物活体信息检测竞赛。本文中使用的指纹数据包括了真指纹图像，如图4-1所示，分别由四种材料制备的假指纹图像，这四种材料分别是共聚酯(Ecoflex)、明胶(Gelatine)、乳胶(Latex)、木浆胶(WoodGlue)如图4-2所示，图像像素为112×112的256级灰度图。

|  |  |
| --- | --- |
| G:\livdet2019\train\Live\Normal\002\0\002_0_0_CROP_0.png   1. 真指纹1 | G:\livdet2019\train\Live\Normal\002\0\002_0_0_CROP_1.png   1. 真指纹2 |

图4‑1 真指纹图像

|  |  |
| --- | --- |
| G:\livdet2019\train\Fake\Ecoflex_00-50\002\0\002_0_0_CROP_0.png   1. Ecoflex材料假指纹 | G:\livdet2019\train\Fake\Gelatine\002\0\002_0_0_CROP_0.png   1. Gelatin材料假指纹 |
| G:\livdet2019\train\Fake\Latex\002\0\002_0_0_CROP_0.png   1. Latex材料假指纹 | G:\livdet2019\train\Fake\WoodGlue\002\0\002_0_0_CROP_0.png   1. Woodglue材料假指纹 |

图4‑2 假指纹图像

### 4.1.1 训练集

本文所提出的卷积神经网络采用了163840张前文提到的图像作为训练集，其中81920张为真指纹图像，20480张共聚酯假指纹图像，20480张明胶假指纹图像，20480张乳胶假指纹图像，20480张木浆胶假指纹图像，即他们的比例为4：1：1：1：1。

### 4.1.2 验证集

本文采用的验证集使用了16384张图片，他们的分布与训练集保持一致，只是等比例缩小了10倍。即8192张为真指纹图像，2048张共聚酯假指纹图像，2048张明胶假指纹图像，2048张乳胶假指纹图像，2048张木浆胶假指纹图像。

### 4.1.3 测试集

本文采用的测试集使用了480张图片，他们的分布与训练集、验证集保持一致。即240张为真指纹图像，60张共聚酯假指纹图像，60张明胶假指纹图像，60张乳胶假指纹图像，60张木浆胶假指纹图像。

训练集、验证集、测试集的图片数据都是相互独立的，即不存在一张图片，同时属于两个或三个集合中。这也是为了保证测试数据的无偏性，毕竟在网络训练的调参中，网络会逐渐适应验证集的情况，在验证集上做出更好的表现，所以在验证集上的表现并不具有无偏性，最后需要用一个测试集来测试算法的性能。

## Inception模块

### 4.2.1 Inception概念

Inception的概念是由Google中的一个团队在他们的GoogLeNet中提出的。它的主要目的是为了解决人们不确定该使用多大的卷积核的困境。在设置卷积核大小的时候，我们可能会想是用1×1，3×3，5×5又或者更大的。而GoogLeNet给出的解决手段就是，全部用上，我们增加网络的宽度，同时在一层内使用1×1，3×3，5×5甚至3×3的最大池化。这样我们的卷积层就具备了更好的自适应性。并且我认为，有些情况下可能一个卷积层内就是又需要提取3×3卷积核的信息，又需要提取5×5卷积核的信息，Inception模块其实是通过拓宽网络的宽度，使得网络在一层中得以提取更多的特征信息。Inception和GoogLeNet也第一次让人们意识到网络不仅在变得更深时有更好的表现，事实上变得更宽也可以。

### 4.2.2 Inception结构

一个Inception模块的结构示意图如图4-3所示。

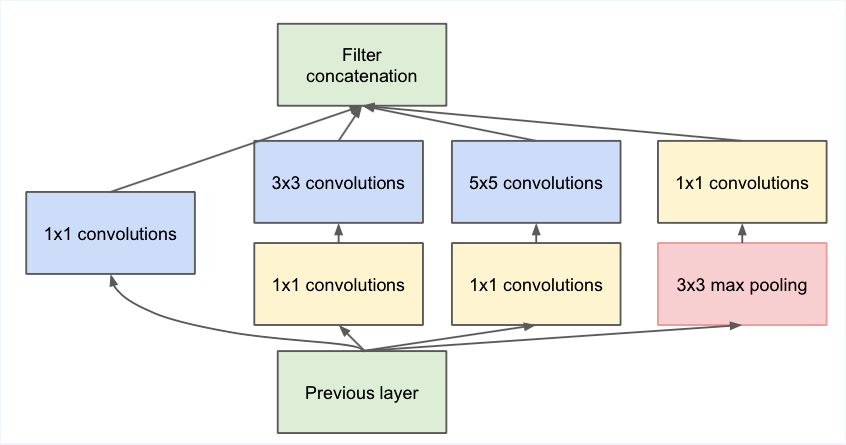


图4‑3 Inception模块示意图

我们可以见到在这个结构图中有四条通道，它不仅包含了1×1的卷积核，还包括了3×3，5×5的卷积核，以及一个3×3的最大池化。

### 4.2.3 瓶颈层

我们可以在前一小节的结构图中看到，不止在1×1的通道中，在3×3以及5×5卷积核的通道中也包含了1×1的卷积核。这些“多余”的1×1的卷积核就是瓶颈层。我们先来考虑这样的情况，假设数据经过一系列的处理变为了28×28×192，接下来需要进入包含32个5×5卷积核的卷积层，那么仅仅在这一步中就需要28×28×192×5×5×32=120422400次的乘法计算，这非常消耗计算资源。所以在GoogLeNet中引入了瓶颈层，他们在进行5×5的卷积操作前加入一层包含16个1×1的卷积核的卷积层，先将数据量减小下来。之所以这样的结构要叫瓶颈层，是因为在这之前与之后的数据量都要大于瓶颈层，这一层就像一个玻璃瓶的颈部一样。这样子总计算量就变为28×28×192×1×1×16+28×28×16×5×5×32=12443648，这样几乎减少了10倍的计算量。并且实践证明只要瓶颈层的大小选择合适，并不会明显影响网络训练好之后的性能。

## 卷积神经网络结构

### 4.3.1 网络结构

本文所搭建的卷积神经网络包含了3个Inception模块，首先先介绍一下这三个Inception模块。Inception\_1的结构如图4-4所示。

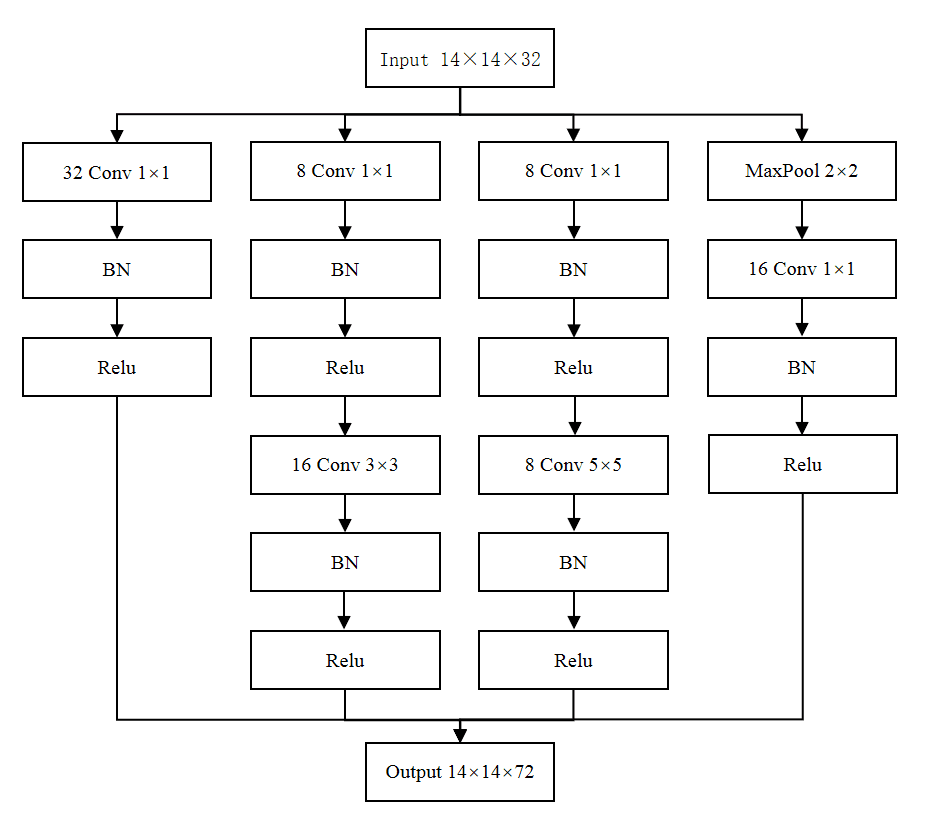


图4‑4 Inception\_1模块示意图

通过这个结构图，我们可以看到Inception\_1的数据输入为14×14×32，表示经过之前的处理，提取出了14×14的特征值，通道数为32。他们来到Inception模块，将分别经过四条路径去进行1×1，3×3，5×5的卷积操作和2×2的最大池化，最后再合并起来，最终输出数据格式为14×14×72。其中第二第三条路径上的第一步8个1×1的卷积核的部分就是瓶颈层，目的是减小网络参数，加快训练速度。

另外两个Inception模块，即Inception\_2、Inception\_3具有相似的结构，但是由于数据格式的不同在具体的参数上略有不同，这里也给出他们的详细结构图。而在后文中的全部网络结构图中，将会分别以简便的形式表示这些Inception模块，这是为了使得网络图的可读性更好，它的具体内容就是如这部分所示的图中那样。如图4-5和4-6所示为Inception\_2和Inception\_3的结构图。

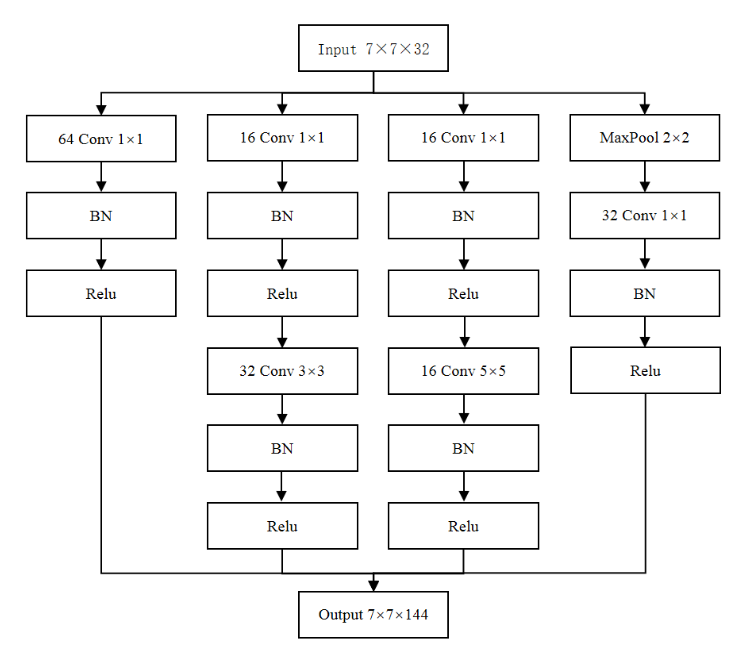


图4‑5 Inception\_2模块示意图

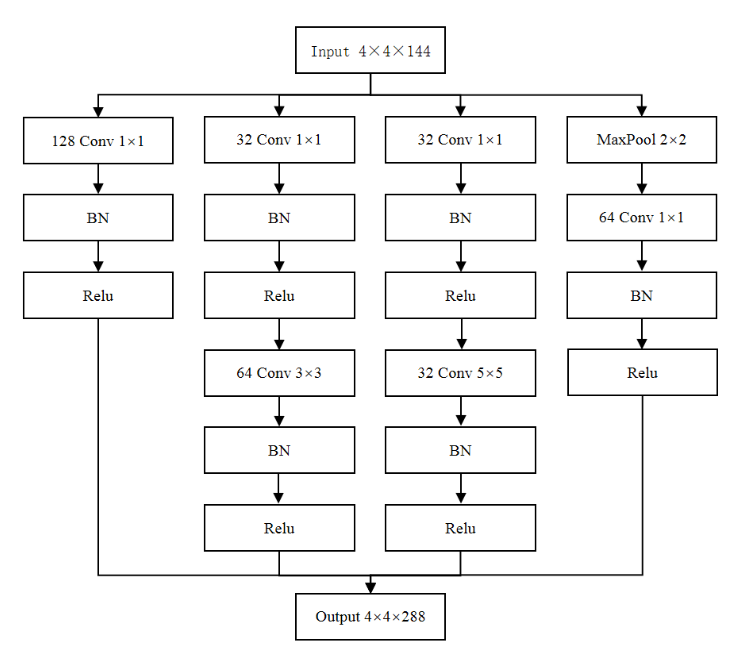


图4‑6 Inception\_3模块示意图

以上部分是本文提出的卷积神经网络的Inception模块部分的结构图。下面将给出完整的网络结构。如图4-7所示。

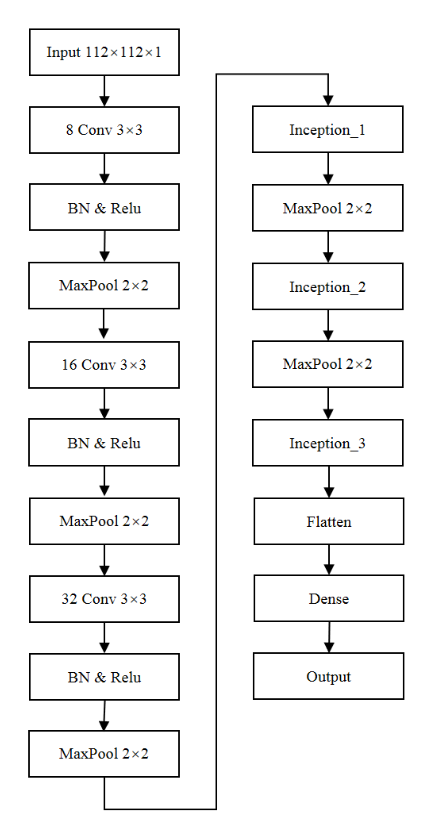


图4‑7 完整网络结构示意图

如图所示为一个10层的卷积神经网络，以可训练层计算，即卷积层加上全连接层。如果算上池化层则为16层，但是习惯上我们只计算可训练层。这个网络的开始通过三块卷积、池化的结构不断提取特征，然后进入三块Inception模块，最后拉直送入全连接层进行真假指纹的分类预测。

### 4.3.2 尺寸计算

在卷积神经网络中，图片(数据)的尺寸计算非常重要，必须要保证数据可以在网络中正常的流通，下面给出了数据在网络中的变化情况。如图4-8所示。

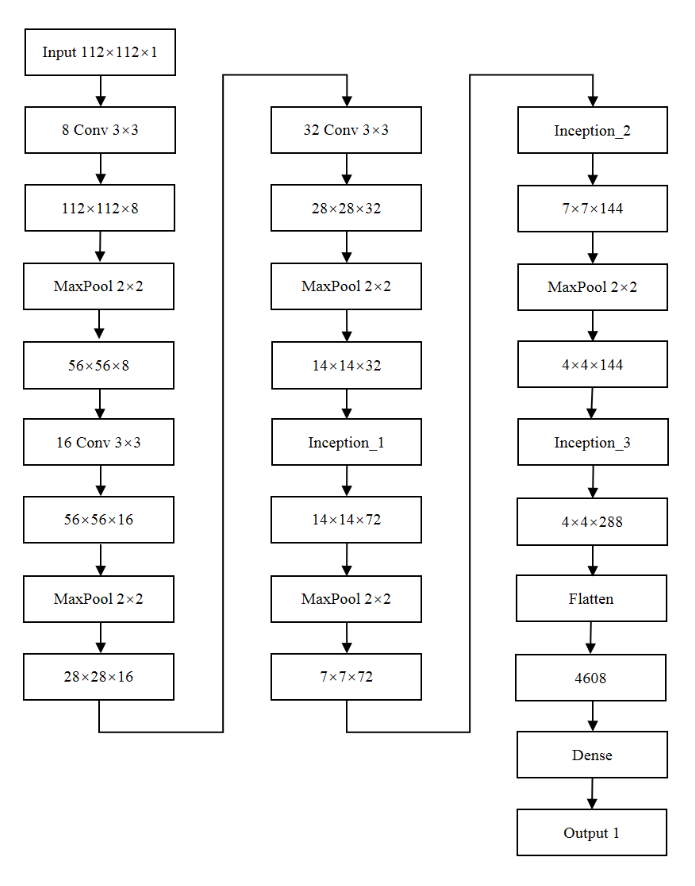


图4‑8 数据变化流程图

从图中我们可以看到，卷积层通常并不改变数据大小，但是有可能会改变通道数，即数据的第三维，而池化层可以改变数据的大小。粗糙一点理解的话，卷积层的功能是提取出当前数据下的深一层的特征信息，而池化层是保留下更重要的特征信息同时往往减小数据的尺寸以保证网络的训练速度在可接受的范围内。从图中我们可以看到对于一张112×112像素的灰度原图，经过一次卷积、池化之后变为了56×56×8；再经过一次卷积、池化之后变为28×28×16；之后再一次卷积、池化变为14×14×32。数据的大小不断变小，但是通道数逐渐增多，这样的结构有利于更好地提取特征信息。然后经过三个Inception模块之后数据格式变为了4×4×288，将这些数据拉平，送入全连接层就可以用于后续分类预测了。也就是说我们从112×112的图像中提取出了4608个高维特征，并使用这些特征进行真假指纹的分类。

### 4.3.3 损失函数

在这个网络中采用的损失函数为二元交叉熵，这个损失函数通常应用于二分类问题，显然指纹防伪算法就是一个二分类问题。二元交叉熵损失函数如式4-1所示。

（4-1）

其中y表示真实值向量，f(x)表示预测值向量，yi的取值为0(表示假指纹)或1(表示真指纹)，f(xi)的取值在0～1之间。只有当f(x)=y时，即两个向量完全相同时该损失函数才会为0，其余时候都是一个正数，且差距越大，损失函数越大。

### 4.3.4 训练过程

如图4-9所示，为训练过程中的截图。

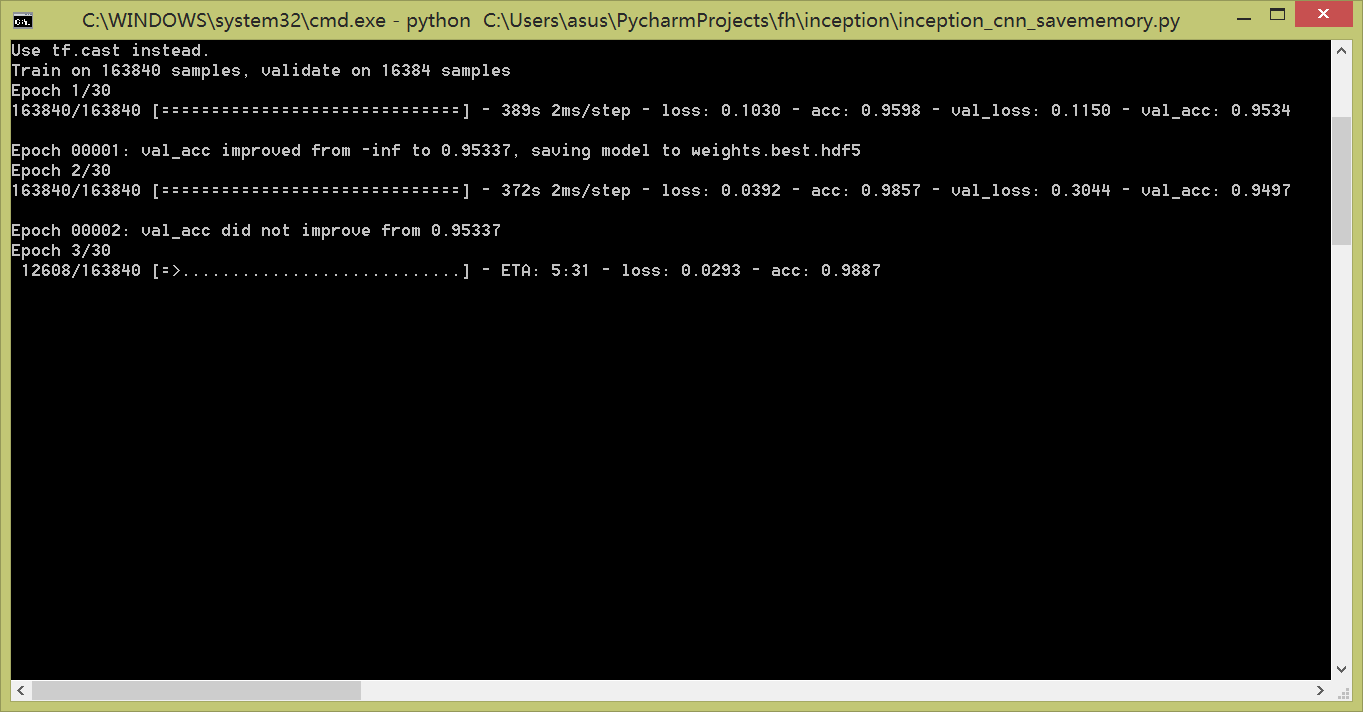


图4‑9 训练过程截图

这个卷积神经网络需要进行30次总训练集的迭代，才能得到最终的网络权值，并且以其中在验证集上表现最好的一次作为结果。

## 本章小结

本章主要讲述了该卷积神经网络的搭建过程，4.1节是划分数据集的过程，4.2节具体介绍了Inception的相关内容，4.3节则是本文所提出网络的结构。本章内容是本文算法的重点。

# 第五章 指纹防伪算法的卷积神经网络训练优化

## CPU性能分析

在第一版的程序中网络的训练运行在CPU上，如图5-1。

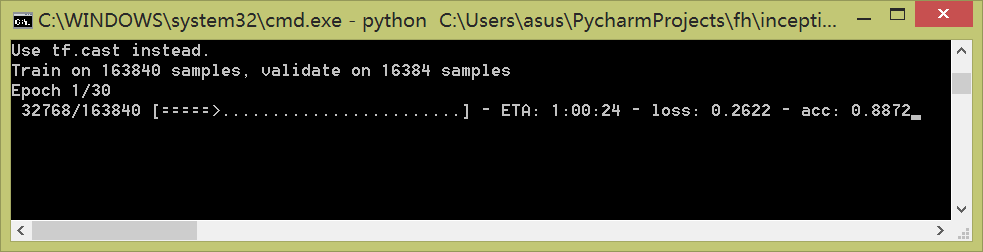


图5‑1 CPU训练截图

当训练至第一次迭代的20%时，第一次迭代的预估剩余时间仍然有1小时之久。以此计算的话，全部的训练需要1×4/5×30=24小时，整整一天的时间，这样的时间成本过高了。并且我们此时查看当时的CPU与GPU运行情况，如图5-2和图5-3所示。可以看到CPU利用率为92.5%，而GPU则为0%。

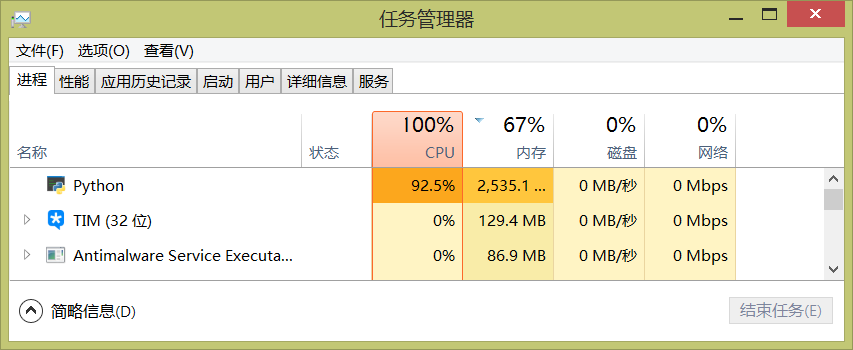


图5‑2 CPU训练时CPU使用率

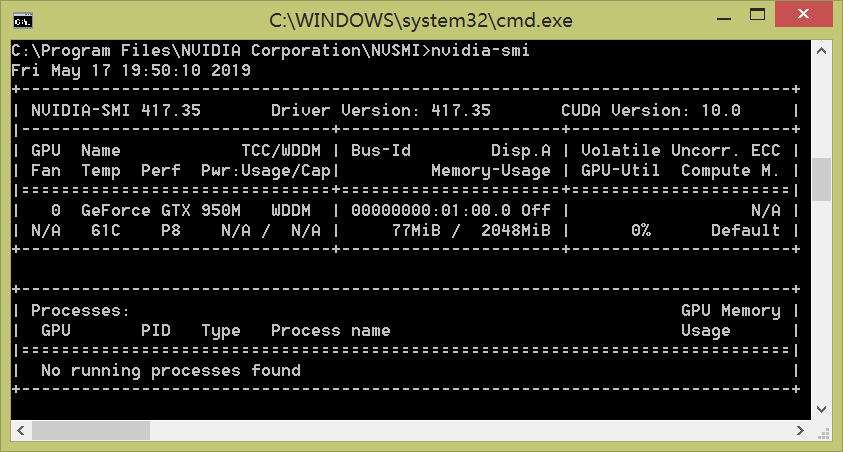


图5‑3 CPU训练时GPU使用率

## 配置GPU加速

### 5.2.1 显卡驱动更新

由于神经网络在GPU上的训练需要Cuda框架与Cudnn加速库两样工具的支持。而他们的版本与TensorFlow和Keras由有所牵连，必须搭配好。他们对于较新的显卡驱动版本的支持比较好，所以我们需要更新自己的显卡驱动，如图5-4所示，驱动更新至417.35。



图5‑4 NVIDIA显卡驱动版本

### 5.2.2 Cuda与Cudnn

在自己的Python工程环境中配置Cuda与Cudnn库，由于本项目中采用了Anaconda管理工具，所以这一步比较轻松，并且有可视化界面。如图5-5所示。

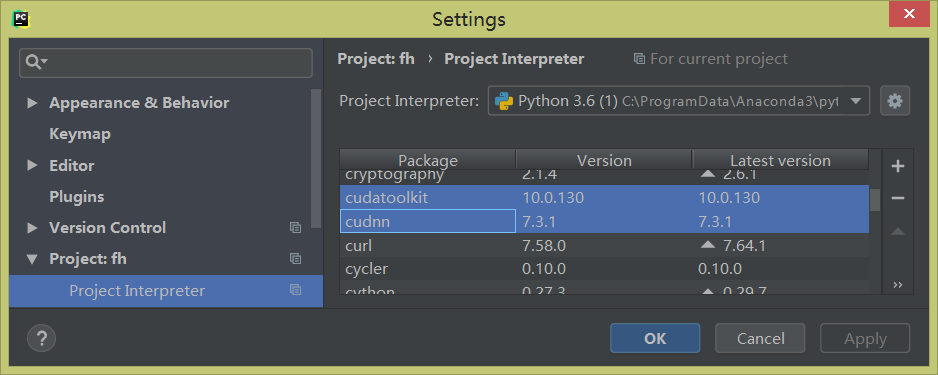


图5‑5 Cuda框架与Cudnn加速库

## 训练性能比较

在配置好了GPU加速之后，训练时的CPU与GPU使用情况如图5-6与图5-7所示。CPU利用率仅为46.2%，而GPU利用率达到71%。

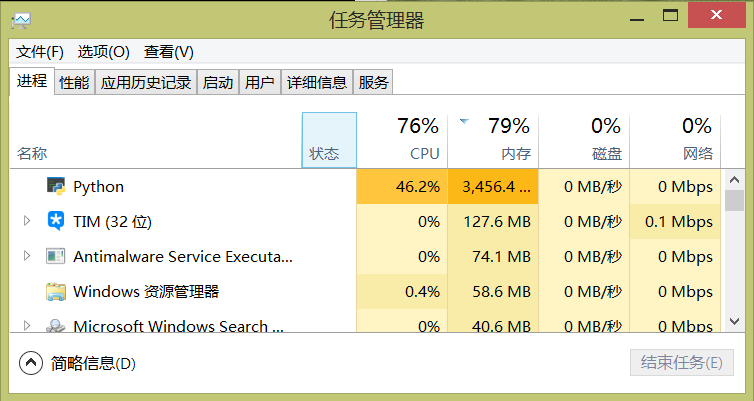


图5‑6 GPU训练时CPU使用率

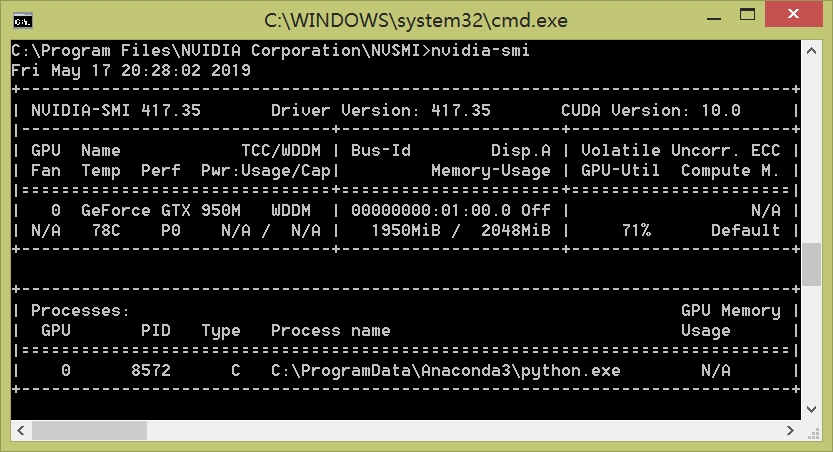


图5‑7 GPU训练时GPU使用率

而训练速度则如图5-8所示。

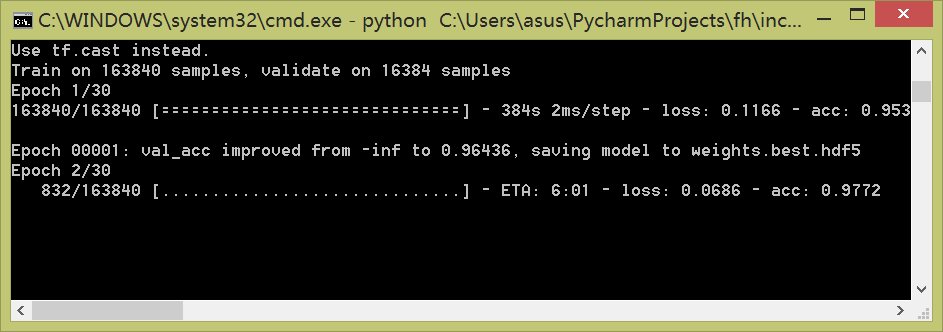


图5‑8 GPU训练截图

第一次迭代的完整训练时间为384秒，依次为计算的话，总训练时间为384/60/60×30=3.2小时，训练速度提升至7.5倍，带来了明显的性能上的提升。

## 本章小结

本章内容介绍了在使用CPU训练速度不理想的情况下，配置GPU训练的过程，并就前后的训练时的速度，CPU与GPU的利用率做了对比。

# 第六章 实验结果分析

## 实验结果

经过了182.5分钟之后，网络结束了30次完整的迭代，76800次权值更新，其中第19次迭代后形成的权值在验证集上的表现最好，准确率为99.98%，损失为0.0011，此时在训练集上的准确率为99.91%，损失为0.0031。如表6-1所示。

表6‑1训练中部分迭代结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 迭代次数 | 耗时[秒] | train\_loss | train\_acc[%] | val\_loss | val\_acc[%] |
| 17/30 | 365 | 0.0031 | 99.89 | 0.7267 | 90.12 |
| 18/30 | 365 | 0.0035 | 99.89 | 0.0024 | 99.93 |
| 19/30 | 365 | 0.0031 | 99.91 | 0.0011 | 99.98 |
| 20/30 | 365 | 0.0029 | 99.90 | 0.0014 | 99.98 |
| 21/30 | 364 | 0.0025 | 99.92 | 0.1870 | 96.59 |

此时在测试集上的准确率为96.875%，损失为0.2455，如表6-2所示。

表6‑2算法在测试集上表现

|  |  |  |  |
| --- | --- | --- | --- |
| 损失值 | 拒真率[%] | 认假率[%] | 总准确率[%] |
| 0.2455 | 1.666 | 4.593 | 96.875 |

在训练过程中在训练集与验证集上的准确率与损失函数值变化情况分别如图6-1所示和图6-2所示。图6-3为同时绘出准确率与损失函数的折线图。

|  |  |
| --- | --- |
| 图6‑1 准确率折线图 | 图6‑2 损失值折线图 |

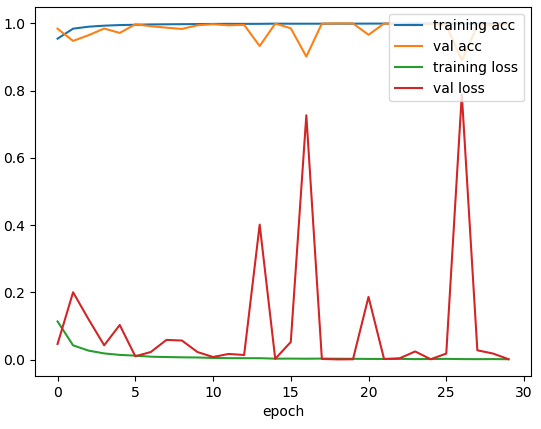


图6‑3 准确率与损失值折线图

其中两条蓝线是训练集的情况，两条黄线是验证集情况。可以看到在训练集上准确率稳步提升，损失值也稳定减小，而在验证集上则出现了较明显的波动。这应该是因为网络是根据训练集的数据进行的训练，而验证集只是为了验证网络的效果并帮助我们保留最优权值，而且在准确率图中的y轴范围是0.90至1.00，所以其实波动并不是那么的夸张。从图中反映出，似乎第1次迭代结束的时候，权值在训练集与验证集上的表现就已经达到了不错的水平。我想这是因为训练集的数据有163840张图片，而batch\_size设置为64，也就是说哪怕是第一次迭代的结束权值也已经调整了2560次。

## 实验结果分析

该训练好的算法在测试集上的拒真率为1.666%(4/240)，认假率为4.583%(11/240)，总准确率为96.875%(465/480)。本文算法无论拒真率还是认假率均保持了较高的水准，准确率也相当不错，可以认为本算法具备较为优良的性能。但是本文中测试集的准确率是通过自己划分的测试集数据计算所得，由于测试集与训练集属于同源数据，尽管并没有交集，也容易导致准确率相对较高。

## 本章小结

本章说明了实验的结果，算法的准确率，说明了本文算法的性能好坏。

# 第七章 总结

## **完成的工作**

1、学习卷积神经网络的相关知识，了解卷积神经网络的发展历史与一些经典网络，学习卷积神经网络的一些基本部件，了解如何搭建一个网络。

2、对数据集进行划分，划分出合适的训练集、验证集、测试集，用于后续的网络训练。

3、搭建卷积神经网络，语言采用了Python，这一在机器学习领域最流行的语言，框架使用Keras，以TensorFlow为后端。在网络中加入了合适的Inception模块以提升网络的性能。

4、出于训练速度的考虑，配置了GPU加速训练，主要就是使用了Cuda框架与Cudnn加速库。说明了算法的性能好坏。

## **存在的问题及下一步工作**

### 7.2.1 存在的问题

1、数据同源问题，由于数据来源唯一，后续训练集、验证集、测试集均为自己划分而得，导致算法的泛化能力较差。

2、网络结构相对来说还比较简单，受限于硬件条件的限制，出于训练速度的考量，没能够搭建一个更复杂的卷积神经网络。

3、关于训练过程中出现的一些情况，没有一个准确的理解，导致出现一些意想之外的情况，不能很快的定位到问题点。

### 7.2.2 下一步工作

1、收集更多的指纹数据，以增强算法的健壮性，可以考虑通过指纹采集仪采集数据自建指纹库。

2、继续修改网络结构，尝试更复杂的模块与结构，有可能会有更好的结果。由于目前已经是一个10层的卷积神经网络，可以算是一个深度神经网络，可能会遇到在迭代后期随着代数增加，性能反而下降的情况，符合使用ResNet中提出的残差模块的情况。所以一方面考虑继续增加网络层数，另一方面考虑加入残差模块结构。当然这需要更好的硬件支持，仅仅依靠笔记本电脑应该是不够的。

3、加强自己对卷积神经网络、机器学习的理解，这里面可以挖掘的东西还很多，只有理解的更深入，才可能做出更好的结果。

# 参考文献

[1]方珊珊, 金亦挺, 朱叶骏, et al. 一种改进的基于小波分析的假指纹检测方法[J]. 浙江工业大学学报, 2018, v.46；No.194(04):32-36.

[2] 陈锋, 覃征, 张永良. 基于“主中心点”脊线跟踪的指纹分类算法[J]. 计算机工程与设计, 2007, 28(8):1752-1756.

[3] 张永良, 周冰, 詹小四,等. 基于深度学习的小面积指纹匹配方法[J]. 计算机应用, 2017, 37(11):3212-3218.

[4]Li Q , Chan P P K . Fingerprint liveness detection based on binarized statistical image feature with sampling from Gaussian distribution[C]// International Conference on Wavelet Analysis & Pattern Recognition. IEEE, 2014.

[5]Wu C J , Chiu C T . [IEEE 2017 IEEE International Workshop on Signal Processing Systems (SiPS) - Lorient (2017.10.3-2017.10.5)] 2017 IEEE International Workshop on Signal Processing Systems (SiPS) - Dry fingerprint detection for multiple image resolutions using ridge features[J]. 2017:1-5.

[6]Qi J , Xie M . A robust algorithm for fingerprint singular point detection and image reference direction determination based on the analysis of curvature map[C]// IEEE Conference on Cybernetics & Intelligent Systems. IEEE, 2008.

[7]Jia X , Yang X , Cao K , et al. Multi-scale local binary pattern with filters for spoof fingerprint detection[J]. Information Sciences, 2014, 268(2014):91-102.

[8]Jingyu S , Jinghui Y , Xiaoguang L , et al. Novel fluorescent label based on YVO4: Bi3+, Eu3+ for latent fingerprint detection[J]. Dyes and Pigments, 2018:S0143720818316966-.

[9]Ru GY., Wei ZG., Hong LQ. (2018) Gesture Recognition Algorithm Based on Fingerprint Detection. In: Xhafa F., Patnaik S., Zomaya A. (eds) Advances in Intelligent Systems and Interactive Applications. IISA 2017. Advances in Intelligent Systems and Computing, vol 686. Springer, Cham

[10] Chen X , Tian J , Cheng J , et al. Segmentation of Fingerprint Images Using Linear Classifier[J]. EURASIP Journal on Advances in Signal Processing, 2004, 2004(4):480-494.

[11] Gao F , Han J , Lv C , et al. Application of core–shell-structured CdTe@SiO2quantum dots synthesized via a facile solution method for improving latent fingerprint detection[J]. Journal of Nanoparticle Research, 2012, 14(10):1191 (11 pp.)---1191 (11 pp.).

[12]Tang T , Wu X , Xiang M . An Improved Fingerprint Singular Point Detection Algorithm Based on Continuous Orientation Field[C]// International Symposium on Computer Science & Computational Technology. IEEE Computer Society, 2008.

[13] 王珏, 石纯一. 机器学习研究[J]. 广西师范大学学报(自然科学版), 2003, 21(2):1-15.

[14] 周志华, 王珏. 机器学习及其应用2009[M]. 清华大学出版社, 2009.

[15] 周飞燕, 金林鹏, 董军. 卷积神经网络研究综述[J]. 计算机学报, 2017, 40(6):1229-1251.

[16] 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.

[17] 江璐, 赵彤, 吴敏. 基于深度卷积神经网络的指纹纹型分类算法[J]. 中国科学院大学学报, 2016, 33(6):808-814.

[18] 龙敏, 龙啸海, 马莉. 基于深度卷积神经网络的指纹活体检测算法研究[J]. 信息网络安全, 2018, No.210(06):34-41.

[19] 冯国徽. 基于卷积神经网络VGG模型的小规模图像分类[D].

[20] 刘建伟, 刘媛, 罗雄麟. 深度学习研究进展[J]. 计算机应用研究, 2014, 31(7):1921-1930.

[21]孙志军, 薛磊, 许阳明,等. 深度学习研究综述[J]. 计算机应用研究, 2012, 29(8):2806-2810.

[22] 陈先昌. 基于卷积神经网络的深度学习算法与应用研究[D]. 浙江工商大学, 2014.

[23]Park Y , Jang U , Im J , et al. Fake Fingerprint Detection Based on Statistical Moments[J]. 2017.

[24]Beom K J , Wonjune L , Heeseung C , et al. An Incremental Learning Method for Spoof Fingerprint Detection[J]. Expert Systems with Applications, 2018:S0957417418305694-.

[25]Shin Y N , Jun I K , Kim H , et al. Performance Assessment Method for a Forged Fingerprint Detection Algorithm[M]// Advances in Information Security and Its Application. Springer Berlin Heidelberg, 2009.

[26]Sharma R P , Dey S . Fingerprint liveness detection using local quality features[J]. The Visual Computer, 2018(3).

[27]Bhanarkar A , Doshi P , Abhyankar A , et al. Joint time frequency analysis based liveness fingerprint detection[C]// Image Information Processing (ICIIP), 2013 IEEE Second International Conference on. IEEE, 2013.

[28]Choi, M.J., McBean, K.E., Ng, P.H.R. et al. J Mater Sci (2008) 43: 732. https://doi.org/10.1007/s10853-007-2178-5

[29]Mohammadi S , Farajzadeh A . Fingerprint Reference Point Detection Using Orientation Field and Curvature Measurements[C]// IEEE International Conference on Intelligent Computing & Intelligent Systems. 0.

[30]Fischer R., Kiertscher T., Gruhn S., Scheidat T., Vielhauer C. (2012) From Biometrics to Forensics: A Feature Collection and First Feature Fusion Approaches for Latent Fingerprint Detection Using a Chromatic White Light (CWL) Sensor. In: De Decker B., Chadwick D.W. (eds) Communications and Multimedia Security. CMS 2012. Lecture Notes in Computer Science, vol 7394. Springer, Berlin, Heidelberg

[31]Baldisserra D., Franco A., Maio D., Maltoni D. (2005) Fake Fingerprint Detection by Odor Analysis. In: Zhang D., Jain A.K. (eds) Advances in Biometrics. ICB 2006. Lecture Notes in Computer Science, vol 3832. Springer, Berlin, Heidelberg

[32]Ellingsgaard J., Busch C. (2017) Altered Fingerprint Detection. In: Tistarelli M., Champod C. (eds) Handbook of Biometrics for Forensic Science. Advances in Computer Vision and Pattern Recognition. Springer, Cham

[33] Lecun Y , Bottou L , Bengio Y , et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

[34] Krizhevsky A , Sutskever I , Hinton G . ImageNet Classification with Deep Convolutional Neural Networks[C]// NIPS. Curran Associates Inc. 2012.

[35] Simonyan K , Zisserman A . Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

[36] Szegedy C , Liu W , Jia Y , et al. Going Deeper with Convolutions[C]// 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015.

[37] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[C]// IEEE Conference on Computer Vision & Pattern Recognition. 2016.

# 致谢

# 附录

### 附录1 毕业设计文献综述

### 附件2 毕业设计开题报告

### 附件3 毕业设计外文翻译（中文译文与外文原文）