目录

第一章 前　言

1.1 研究背景与意义

随着计算机硬件水平的发展，计算能力的不断提高，机器学习重新得到了研究者们的广泛关注，并在最近几年得到了迅速的发展。Tom Mitchell[1]将机器学习定义为：一个程序被认为能从经验 E 中学习，解决任务 T，当且仅当，有了经验 E 后，经过 P 评判，程序在处理 T 时的性能有所提升。简单来说，研究人员期望计算机像人一样去解决任务。而其中核心问题之一就是机器视觉。机器视觉的目标是让计算机可以模仿人，自动地处理图像信息。目前，我们接触最多的机器视觉技术就是人脸识别技术。

人脸识别，是基于人的面部特征进行身份识别的一种生物识别技术。人脸虽然没有指纹，虹膜等其他生物特准确，但是具有采集简单、方便的优势。所以，被广泛用于智能手机的解锁功能中，并被经常作为数字密码的替代方案。

人脸识别技术主要解决的问题是将输入的多张图片转化为一张由多个特征可区分的特征向量组成的“特征脸”，然后利用“特征脸”来完成分类问题。因此在对人脸照片转化为人脸特征向量之后选择合适的分类器也十分重要。目前围绕机器学习的分类器有很多，选择合适的分类器对于提升面部识别类产品的效果至关重要。本文研究多因素下不同分类器在人脸识别问题中的表现情况。

1.2 人脸识别研究的历史

人脸识别起源于 20 世纪。最早的人脸识别技术为基于人脸的几何特征(geometric feature based)。Bledsoe 等人[2]提出，将人脸五官、眉毛等特征点之间的距离、大小作为特征向量后，使用最近邻方法进行分类，从而进行判断。这种方法虽然在计算速度和内存方面有优势，且对照片质量要求不高但很容易受到表情、拍摄角度等因素的影响，所以只能停留在理论研究阶段，无法应用于实践中。

随后，在 1991 年麻省理工学院 Turk 和 Pentland 提出的特征脸方法(Eigen face) [3]具有划时代的意义。该方法的流程如下：

1) 准备一组面部图像训练。构成训练集的图片应该满足在相同的光照条件下拍摄的,具有相同的像素，且眼睛和嘴巴对齐。将每个图像根据灰度转化为一个矩阵。
2) 计算平均图像后，将各张图片与平均图像相减。计算平均图像的方法是对训练集里的样本求平均值。之后再用每张图片得到的矩阵和平均图片得到的矩阵相减。
3) 计算协方差矩阵 S 的特征向量和特征值。由于每个特征向量具有和原始图像相同的维数，因此可以将其视为图像。我们将协方差矩阵的特征向量称为特征脸。特征脸的方向经常和平均图像的方向不同。
4) 选择主成分。按降序的方法对特征值进行排序。主成分 k 的个数通过所有变量的方差上的阈值确定。

5）通过分类器，对新传入的图像进行分类，完成人脸识别。

之后，Belhumer 在他的论文中将线性判别分析应用到人脸识别中，提出 Fisherface 算法。总体来说，这一阶段的人脸识别技术得到了较大的发展，在一些识别系统中有不错的效果。

在这之后，传统的机器学习技术也被应用到了，许多更有效的如遗传算法[4]，AdaBoost [5],贝叶斯分类器，支持向量机，神经网络等方法被应用到人脸识别中并有不错的效果。

最近几年，随着硬件技术的快速发展，硬件特别是 GPU 不再对人脸识别技术有十分大的制约。人脸识别技术发展重心也逐渐从传统的机器学习方法，转变到了深度学习技术。Facebook 的 DeepFace[6]，Google 的 FaceNet[7]等框架都取得了超过传统机器学习方法的成就。

于此同时，人脸识别这个概念也发生了变化。人脸识别不再仅限于人脸检测以及对检测出的人脸进行分类者两块内容。随着传感器的进步，人脸模型从二维的模型转化为三位的立体模型，对人面部表情的追踪和替换等其他内容也被纳入到人脸识别的相关研究领域中。

虽然深度学习技术使得人脸识别技术有了很大的发展，但是由于应用范围的扩大所产生的如下不确定性因素使得人脸识别技术在实际应用种面临更大的挑战：

1)人脸在图片中的实际占比

2)复杂的背景光照条件及不同的面部表情

3)面部额外的如口罩、胡须、墨镜等遮挡物

4)面部因受伤、疾病和年龄等情况所带来的变化

相信随着计算机的发展和研究人员的进一步研究，人脸识别技术也将会有进一步的发展。

1.3 分类器介绍

在人脸识别的过程中，获取了人脸图像的特征向量之后，此时需要利用分类器根据特征向量对人脸图像进行分类处理，以确定当前人脸的类别。在这个过程中，分类器起着决策机制的作用，对最终的判别非常关键。分类器性能的优劣也将直接关系到人脸识别的好坏。常用的分类器有以下几种：

1.3.1 K 最近邻(k-NN，k-NearestNeighbor)

k-NN 算法[8]是对近邻算法的改进，是一种直观、易于实现的算法。k-NN 算法通过测量不同特征值之间的距离进行分类。它的思路是：将样本标记为在训练集中 k 个最相似（即特征空间中最邻近）的样本中大多数属于的标签。因此，该分类算法只依赖少数邻近的样本。

k-NN 算法的优点在于：

1) 简单有效，易于实现。k-NN 算法十分简单好用。实际上，k-NN 是机器学习中最简单的算法之一。有效在于，k-NN 的理论成熟。有许多实验证明，k-NN 算法在很多数据集中有不错的效果。
2) 对异常值不敏感。因为 k-NN 算法只依赖于少数邻近的样本。因此，对于异常值，是没有其他分类算法那么敏感的。
3) 训练速度快。由于 k-NN 算法属于懒惰学习算法，因此他的训练时间较快，重新训练的成本也比较低。

但是 k-NN 算法同时存在着一些致命的缺点：

1) 由于 k-NN 算法是懒惰学习算法，虽然它在训练时间上相对于积极学习算法短，但是在计算时间上，尤其是在大训练集上，会比积极学习算法上的多。不仅是计算量大，对内存的占用也非常大。
2) 样本不平衡问题。当训练集中某一类的数量相对于其他类较大的时候，会导致该算法忽视样本间实际的距离分布情况，而倾向于将待分类样本归为在训练集中占较大比例的样本的一类。
3) 可解释性差，无法给出决策树那样的规则。

1.3.2 支持向量机(Support Vector Machine, SVM)

支持向量机（Support Vector Machine, SVM）是由 Vladimir 和 Alexey 于 1963 年提出的，用于在分类与回归分析中分析数据的监督式学习模型与相关的学习算法。通过使用不同的核函数，SVM 不仅能够进行线性分类，还能将现有特征维度映射到更高维的特征空间中，有效地进行非线性分类，并且这个高维度特征空间能够使得原来线性不可分数据变成了线性可分的。

一般来说，如果特征维数特别低，样本数远超过特征维数，使用非线性核如高斯核效果会比较好。如果两类有较多重叠，则非线性 SVM 的支持向量特别多，选择稀疏的非线性 SVM 会是一个更好的方案。但是当特征量很大时，线性核往往表现得比非线性核更好。所以需要根据实际情况选择线性核和非线性核。

如图 1-1 所示，在一个二位环境中，R,S,G 三点和其他靠近中间黑线的点可以看作支持向量。分类器，也就是黑线由它们决定。

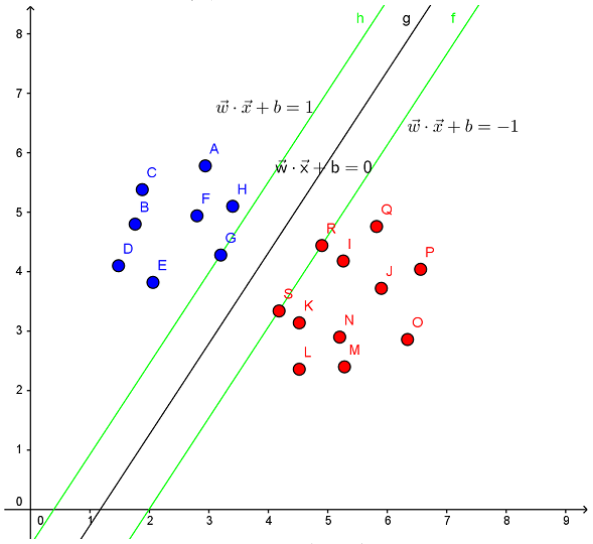

图 1-1 支持向量机解释

线性支持向量机的核心思想是确定线性分类的最佳超平面$f(x) = x\omega^T + b = 0$中ω和 b 的值。具体方法如下：

1) 通过两个分类的最近点，找到$f(x)$的约束条件。
2) 在约束条件下，通过拉格朗日乘子法和 KTT 条件求拉格朗日子乘$\propto_i$和 b。
3) 将$\propto_i$和 b 作为分类其的最终参数。

其中，对于异常点可以通过加入松弛变量ξ处理，并剔除用 SMO 求拉格朗日乘子α_i和 b 时$\alpha_i = 0$的点。

由于 SVM 算法本质上是一个二分类的算法，将 SVM 应用到多分类的场景中，有两种方案。一种是 OVO(One-Versus-One)，另一种是 OVR(One-Versus-Rest)。

OVO 方案具体做法是，我们选择对任意两个分类之间训练一个 SVM 分类器，因此对于 k 个类别，我们需要训练 k*(k-1)/2 个 SVM 分类器。当对于一个新数据进行分类时，将它放入这 k*(k-1)/2 个 SVM 分类器中进行分类，最终得票最多的类别即为该数据的类别。这种方案的性能虽然很好，但是当类别数目很大时，分类器的数目是 k*(k-1)/2，这是很大的代价。

OVR 方案的具体做法是，我们选择对与每一个类别，训练一个 SVM 分类器，将该类别归为一类，其他类别归为另一类。因此对于 k 个类别，我们需要训练 k 个 SVM 分类器。当对于一个新数据进行分类时，将它放入这 k 个 SVM 分类器中进行分类，最终将新数据分类到具有最大分类函数值的那一类。

1.3.3 人工神经网络(Artificial Neural Network, ANN)

人工神经网络，简称神经网络(Neural Network)是由 Warren McCulloch 和 Walter Pitts 在 1943 年提出的非线性计算模型。在机器学习领域中，它是一种模仿生物神经网络（动物的中枢神经系统，特别是大脑）的结构和功能的数学模型或计算模型，用于对函数进行估计或近似。神经网络是由最基本的单位-神经元(如图 1-1 所示)连接在一起形成的一个类似生物神经网络的网状结构。

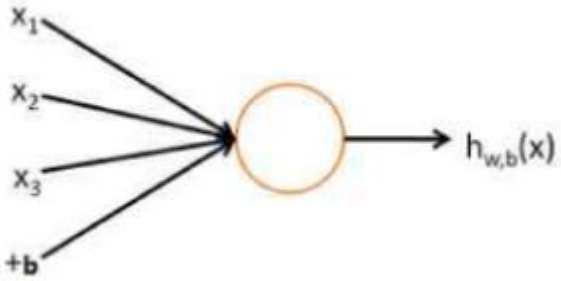

图 1-2 神经元示意图

神经网络是一个能够学习，并进行总结归纳的系统，也就是说它能够通过已知数据的实验运用来学习和归纳总结。与之不同的基于符号系统下的学习方法，它们也具有推理功能，只是它们是创建在逻辑算法的基础上，也就是说它们之所以能够推理，基础是需要有一个推理算法则的集合。

神经网络按照网络架构（Connectionism）分类的话，主要有：前馈神经网络、递归神经网络、强化式架构。

一种常见的多层结构的前馈网络（Multilayer Feedforward Network）由三部分

组成:

1) 输入层（Input layer），众多神经元（Neuron）接受大量非线性输入消息。输入的消息称为输入向量。
2) 输出层（Output layer），消息在神经元链接中传输、分析、权衡，形成输出结果。输出的消息称为输出向量。
3) 隐藏层（Hidden layer），是输入层和输出层之间众多神经元和链接组成的各个层面。隐层可以有多层。隐层的节点（神经元）数目不定，但数目越多神经网络的非线性越显著，从而神经网络的强健性（robustness）（控制系统在一定结构、大小等的参数摄动下，维持某些性能的特性。）更显著。

除了输入层，每一层的神经元都拥有输入和输出。通常来说，每一层的每一个神经元都会与前一层中的所有神经元有关联。我们通常沿用生物学的说法，把神经元和与之对应的神经元之间的连线叫做突触（Synapse）。每个突触都有一个加权数值，我们称作权重(Weight)。因此，我们可以描述第 i 层上的某个神经元所得到的输入，就等于该神经元的每一个突触的权重乘以第 i-1 层上对应的神经元的输出，然后求和得到了第 i 层上的某个神经元所得到的输入。然后输入值通过该神经元上的激活函数（activation function）以控制输出大小，因为其可微分且连续，方便差量规则（Delta rule）处理，求出该神经元的输出。要注意的是，输出经过激活函数之后，会变成一个非线性的数值。即通过激活函数求得的数值，根据阈值来判断是否要激活该神经元，而一个神经元的输出是否为线性并没有任何意义。

神经网络主要通过由 Rumelhart 等人[9]在 1986 年提出的误差反向传播算法（error BackPropagation）进行训练。训练过程包括正向传播和误差的反向传播两个子过程。在正向传播过程中，训练集中的样本由输入层进入网络，经过所有的隐藏层计算后，从网络的输出层中得到最终输出值。随后进入误差的反向传播过程。误差反向传播的核心是得到了最终输出值与期望值之间的误差。通过一些数学公式，可以由后一层的误差值算出给前一隐藏层中的所有单元的误差值。而各层的神经元即可通过误差值更新神经元的权值。网络的训练过程就是通过不断使用正向传播与误差反向传播去调整各个隐藏层中神经元的权值，直到网络输出层的输出值与期望值的误差降低到了预先设置的一个阈值，或是达到了预先设置的训练次数。

1.3.4 卷积神经网络（Convolutional Neural Network, CNN）

卷积神经网络（Convolutional Neural Network, CNN）是由 Hubel 和 Wiesel 在 20 世纪 60 年代受到猫和猴子的视觉皮层所启发而提出的。

卷积神经网络（Convolutional Neural Network, CNN）是一种深度前馈神经网络，对于大型图像处理有出色表现。

卷积神经网络（Convolutional Neural Network, CNN）[6]是基于人工神经网络的深度学习方法，成功应用于图像识别领域。神经网络通常建立在前馈神经网络模型之上，与前馈神经网络不同的是，卷积神经网络的“隐藏层”被“卷积层”，“线性整流层”，“池化层”，“全连接层”所代替。这一模型也可以使用反向传播算法进行训练。相比较其他深度、前馈神经网络，卷积神经网络包含的参数更少，使之训练时间相比之下能够缩短不少。

- **卷积层（Convolutional layer）**，CNN 中每层卷积层由若干卷积核组成，每个卷积核的参数都是通过反向传播算法优化得到的。卷积运算的目的是提取输入数据中不同的局部特征。第一层卷积层可能只会提取一些低级的特征如边缘、线条和角等层级，随着网络深度的增加，CNN 能从低级特征中迭代提取更复杂的特征。因为这一特性，CNN 能够自动提取特征而无需预处理去突出特征。但是由于图像噪声可能很大，很多时候我们仍会选择预处理，避免 CNN 将噪声作为图像特征，并通过噪声来识别图像，而并非通过我们想要其学习的特征。
- **线性整流层（Rectified Linear Units layer, ReLU layer）**，这一层神经元使用使用线性整流（Rectified Linear Units, ReLU）f(x)=max(0,x)作为它们的激活函数（Activation function）。实际上还有其他不同的激活函数，如双曲正切函数（tanh），Sigmoid 函数。相比于之前经典神经网络常用的激活函数 Sigmoid 函数，ReLU 函数并不会在算法训练后期出现训练速度骤降的情况，从而对神经网络的训练速度提升数倍，并且并不会对模型的泛化准确度造成显著影响。
- **池化层（Pooling layer）**，是卷积神经网络中另一个重要的概念，它实际上是对上一层的输出进行降采样。池化层通常会分别作用于每个输入的特征并减小其大小。目前最常用的池化层类型是每隔 2 个元素从图像划分出 2*2 的区块，然后对每个区块中的 4 个数取最大值。这将会减少 75%的数据量。池化层通常在卷积层的后面，通过将卷积层得到的维度很大的特征切成几个区域，取其最大值或平均值，得到新的、维度较小的特征，来减少了参数的数量，防止过拟合现象的发生。
- **全连接层（ Fully-Connected layer）**，是把之前分割出来的所有局部特征组合变成全局特征，将其用于计算最后每一类的概率。

由于全连接层占了整个卷积神经网络大部分的参数，因此很容易出现过拟合。在经典神经网络中，有一种很流行的防止过拟合的方法，叫做交叉验证（Cross-Validation）。但是它用在卷积神经网络中通常并不合适，因为卷积神经网络解决的问题通常有很大的数据集，在大数据集和深度网络中使用交叉验证的开销太高。现今防止卷积神经网络过拟合的方法有很多种，流行的有：批标准化（Batch Normalization）和 Dropout。

1) 批标准化（Batch Normalization）是由 Sergey Ioffe 和 Christian Szegedy 于 2015 年提出的[11]。它最开始是为了解决随着神经网络深度的加深而导致的难以训练的问题，却发现对于解决过拟合也很有帮助。根据统计机器学习中的一个经典的假设，若源空间和目标空间的数据分布是不一致的话，通过训练集数据得到的模型是很难在测试集上获得好的效果的。但是，对于神经网络的各个隐藏层的输出，由于经过了隐藏层内的计算，其数据分布显然会与各个隐藏层对应的输入数据分布有所差异，而且越往后的差异会随着网络深度增大而加大了，但每一层所指向的 Label 仍然是不变的。这就是为什么随着神经网络的深度加深，常常会导致梯度消失（Gradient Vanishing）和梯度爆炸（Gradient Exploding）问题，使得神经网络的训练会越来越困难，收敛速度会变得很慢。一般的解决方案就是根据训练样本和目标样本的比例对训练样本进行标准化处理。所以，可以通过引入 Batch Normalization 来实时标准化指定层的输入，从而固定该层的输入数据的均值和方差。Batch Normalization 一般用在激活函数之前，对神经元未经激活函数的输出值做标准化，使输出值的各个维度的数据分布稳定为均值为 0，方差为 1。通过这样，模型可以使用较大的学习速率，加快了训练过程。并且将数据分布稳定下来，可以缓解了由于数据分布波动过大而造成的过拟合。
2) Dropout 是由 Geoffrey E. Hinton 于 2012 年 7 月提出的，它能很好的优化过拟合

问题。Dropout 具体是指在模型训练时，让神经网络中的某些隐藏层中的某些神经元有 P 概率不工作，有 1-P 的概率工作。不工作的那些神经元在这轮训练中被认为不是神经网络中的一部分，但是它们会被保留下来，因为在下一轮训练中它们可能需要工作，具体如图 1-2 所示。概率 P 属于神经网络中的超参数（Hyper parameter），需要根据实际情况调节。对于 Dropout 的直观解释是，每次用训练集中的样本进行训练时，隐藏层的神经元都是以一定概率工作，因此不会出现某几个隐藏层神经元每次都工作的情况。这样训练模型的结果就不会出现，需要某些隐藏层节点的共同作用才能得到的结果，防止出现某些仅仅在某些特征存在的情况下才有效果的特征。

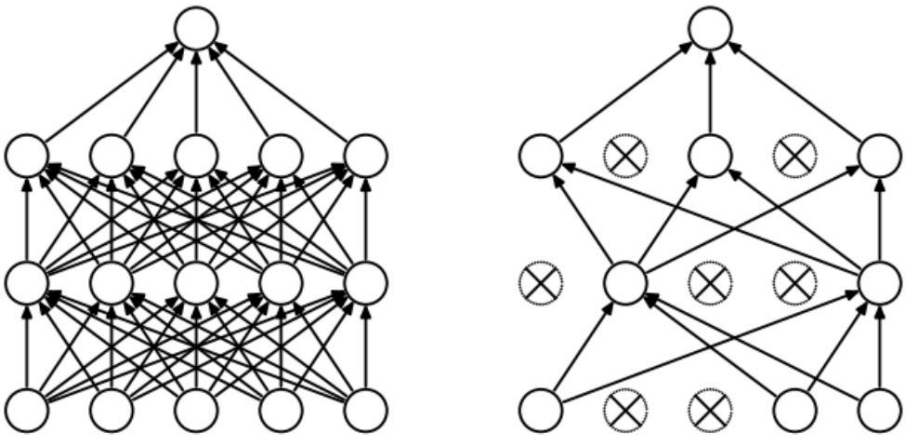

图 1-3 Dropout 示意图

1.3.5 K-Means 算法

K-Means 的算法思想可以追溯到 1957 年 James MacQueen 的论文《Sur la division des corps matériels en parties》，但是直到 1967 年 James MacQueen 才提出术语“K-Means”。虽然 1957 年 Stuart Lloyd 就将 K-Means 的标准算法应用于脉冲编码调制技术中，但该算法直到 1982 年才被贝尔工作室对外公布。因此，我们常引用 1965 年 E.W. Forgy 发布的几乎相同的方法，并将其称为 Lloyd 算法。

K-Means 算法是最常用的无监督学习的聚类算法，其基本思想是：以空间中 k 个点为中心进行聚类，对最靠近它们的对象归类。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

K-Means 虽然有简单易用，只需要调节实际的分类数量 k 且具有出色的速度和可扩展性的优点外，也有十分突出的缺点。其中最主要的缺点是(1)对离群点十分敏感(2)当训练样本很大的时候，开销会非常大。

对于缺点(1),我们可以用 LOF 算法对离群点进行检测并去除离散点。而对于缺点（2），由于当特征数和训练样本数很大时，K-means 的开销巨大，基本无法使用。

因此，D. Sculley 于 2010 年提出了 Mini-Batch K-Means 算法。不同于传统 K-Means 算法，Mini-Batch K-means 算法受到 SGD（Stochastic Gradient Descent）算法的启发。它将每次用所有训练样本数据去更新各聚类中心的值，改成用从训练样本中随机选取的大小为 b 的样本子集去更新各聚类中心的值。这样可以在训练样本很大的时候，大大降低训练的开销。Mini-Batch K-means 算法的缺点在于最终聚类的效果会降低。根据 Javier Béjar 于 2013 年发表的《K-means vs Mini Batch K-means: A comparison》论文，效果降低了 2%-8%。

1.4 本论文的工作

本论文主要研究不同分类算法在人脸识别问题上的效果。首先，本文对人脸识别领域的发展过程进行了概括，并介绍了 4 种分类算法和 1 种聚类算法。接着，在假设日本、韩国和中国人三国之间面部特征相似但依旧可以进行区分的基础上，收集各国国民的图片，按照 8：2 的比例划分训练集和测试集。对经过 0 均值处理后的全图像和裁剪人像的图像用不同的分类算法进行训练和测试，记录各项性能指标。根据测试集的正确率验证了假设：中国人、韩国人、日本人是否存在可以区分的特征。然后，将这些性能指标与 alllooksame.com 中人工识别的平均情况作对比，验证机器学习的算法能否超越人的水平。最后，我们对算法的效果进行了比较并解释了未裁剪图片的正确率高于裁剪图片的原因。

1.5 本论文的结构安排

本论文的结构如下：

第一章首先介绍了本文的研究背景和意义，并对人脸识别的发展进行了概括。接着，对人脸识别中主要应用到的分类及聚类算法进行了介绍。最后，简单介绍了本文所做的工作和结构。

第二章主要描述了实验的具体过程和结果。首先介绍了通过爬虫获取训练样本的过程.然后，通过增强处理扩展了训练样本的数量，并对训练样本进行了人像裁剪及 0 均值处理。最后，展示了不同算法对于训练样本的训练结果。

第三章通过统计学和数学建模的方法对第二章的内容进行了进一步的分析。首先检验了实验的基本假设：中国人、日本人、韩国人有可以通过机器学习进行区分的面部特征。随后，我们在将运行时间纳入考虑范围的基础上，对不同的算法进行了比较。最后，将算法的结果与 alllooksame.com 中人类的分类结果进行对比，验证目前机器学习的算法在近似图片的识别上能否达到甚至超越人的水平。最后，我们对未裁剪图像的正确率高于裁剪图像的正确率做出了解释。

第四章提出了在此次实验中的一些不足和可以改进的地方。供作者在以后的学习过程中进行继续改进和深入研究。

第二章 训练过程及结果

2.1 训练集的获取

虽然在网上有大量人脸的数据集，但由于这些数据集都是根据种族而不是国家进行划分的，所有目前并没有一个可以直接对应我们研究的数据库。因此，我们需要通过爬虫技术来收集相关的照片，作为训练集。

网络爬虫又称网络蜘蛛，网络蚂蚁、网络机器人等，是一种按照一定的规则自动浏览网页并根据我们制定的规则获取信息的脚本。网络爬虫由控制节点、爬虫节点、资

源库构成。

网络爬虫的工作原理可以由图 2-1 来展示

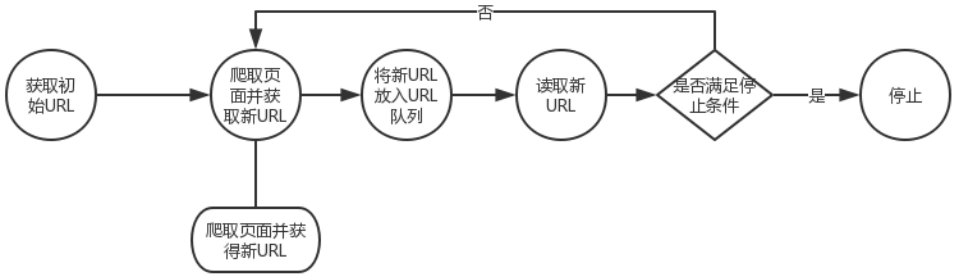

图 2-1 通用网络爬虫的实现原理及过程

我们选用 Python 的 request 模块作为 HTTP 库，用于处理网络协议；selenium 用于模仿真实浏览器，解决 JavaScript 的渲染问题；BeautifulSoup 模块为网页转码，从网页中抓取数据。

我们通过爬虫技术从清华大学，东京大学，首尔大学获取了总共 2051 张不同的照片。其中，中国人的照片合计 706 张，日本人照片 653 张，韩国人照片 692 张。

2.2 训练集的数据增强

我们首先对由爬虫得来原始数据集以 8:2 的比例分割成训练集和测试集，这样避免了在训练集和测试集中有以相同的原始图片增强而来的图片。

在分割出训练集和测试集后，由于我们通过爬虫得到的数据非常有限，但是深度学习在数量足够多的时候才有更好的效果，我们计划通过改变数据的亮度、饱和度、对比度、锐度，小角度旋转图片的方法，做一个数据增强。下面是我们做数据集增强的方法：

1) 对图片进行-23 度到 23 度的随机旋转。
2) 对图片进行颜色抖动。包括，以 0.6 到 1.4 之间的程度随机调节图片的饱和度、亮度、对比度和锐度。
3) 对图片增加均值为 0.2，方差为 0.3 的随机高斯噪声。

重复上述这三个步骤各五次之后，我们获得了样本容量为原来 15 倍的新的数据集。

2.3 训练集的预处理

由于我们需要做对比实验，所以我们首先需要截取数据增强后的数据集中的人脸，得到人脸数据集。

在这里，我们用到了 OpenCV 库中的 Haar Cascade 人脸识别算法(Haar Cascade Face Detection algorithm) [10]对照片中的人脸部分进行识别。在对图片使用 消除由背景产生的噪音。由于我们获取到的训练集的图片大小各异，为了方便神经网络算法和卷积神经网络算法训练，同时也为了加速算法的训练速度和减少训练的内存占用，我们统一将训练集归一为 224(pi)×224(pi)大小。

2.4 数据的封装

tensorflow 官方推荐用 tfrecord 高效地数据读取、存储文件。由于 tfrecord 数据文件是一种将图像数据和标签统一存储的二进制文件，所以我们需要将图片转为二进制格式后通过 example=tf.train.Example()语句将数据通过字典格式赋给变量 example。接着，我们将 example 转换成字符串并完成写入。

2.5 训练过程

我们用训练集分别训练 k-NN, SVM, 2-layer Neural Network, CNN, K-Means 这五种算法以比较他们之间的差别。

2.5.1 k-NN 算法

k-NN 算法是最简单的算法，我们用如下的步骤完成了对训练样本的训练。

输入：设训练数据集$D \in \mathbb{R}^{m \times n}$，其中 m 是图片的数量，n 是每张图片的总像素。

输出：对测试集 D 给出它的标签 y，$y \in L$

对于每一个 $x^i \in X$以及$d^j \in D$,做如下的操作：

计算测试点x^i和第 j 个训练点y^j的距离的平方，并存储为 dist[i,j]中。给出所有测试点中训练点和测试点间的距离。

对于每一个x^i做如下操作：

找到第 i 个训练点x^i最近的 k 个邻居的标签，以及所有标签列表中最常见的标签。通过投票的方法对训练样本进行归类。如果超过一个的标签的投票情况出现平局的情况，通过标记为出现次数最少的标签来打破平局。

2.5.2 SVM 算法

由于训练样本具有高维度的特点，所以我们可以用样本对线性 SVM 模型进行训练。对于训练集$x^{(i)} \in \mathbb{R}^n$，它的铰链损失（Hingle Loss）可以用公式(1)来表示：

$$\frac{1}{N}\sum_{i}\sum_{j \neq y_i}\left[\max\left(0, \omega_j^T x_i - \omega_{y_i}^T x_i + \Delta\right)\right] + \lambda\|\omega\|_2^2 \qquad (1)$$

通过随机梯度下降的方法，不断降低铰链损失函数，最终得到训练好的线性 SVM 模型。

2.5.3 两层神经网络

我们选择了 tensorflow 实现了一个两层神经网络：

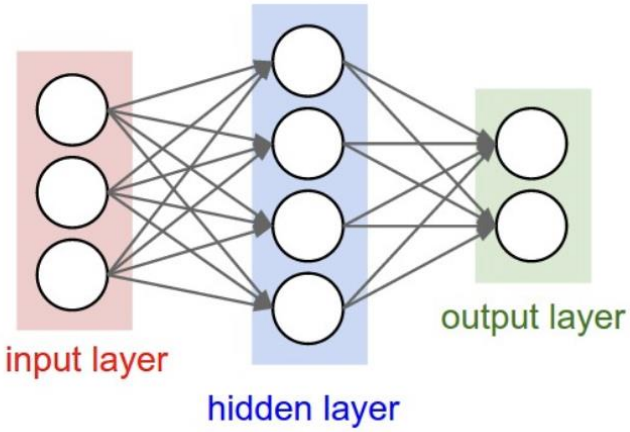

图 2-2 典型两层神经网络的结构

经过预处理的照片被重构成大小为 224*224*3 的行向量，接着所有的行向量以规定好的 Batch 大小，依次输入到网络中。通过交叉验证的方法，我们确定了隐藏层是由 2048 个以 ReLU 函数作为激活函数的神经元组成的全连接层。

最后，会进入一个由 3 个神经元组成的全连接层，这是神经网络中的输出层。在这一层中，将前面所有的操作结合，输出对三个类别的预测值。这一层的输出并不会经过任何处理，直接进入到 Soft-max 层处理。Soft-max 的作用是，将对三个类别的预测值，转换为对三个类别的预测概率。之后我们通过交叉熵函数计算出 loss 值，并通过反向传播来更新所有参数。

2.5.4 卷积神经网络算法

我们使用了经过自己改进的 VGG 网络结构对样本进行训练，并用 TensorFlow 实现。VGG 各种大小网络的结构如图所示

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

图 2-3 VGG 的各种架构

我们采用了 19 层（不计池化层）的 VGG19 结构，并进行自己的改进。假设一张图片进入了 CNN 进行训练，则具体过程为：

1) 图片以大小为 224*224，编码为 RGB 的格式输入，输入时会经过图片标准化处理，使其图片均值为 0，方差为 1。之后会进入第一个卷积大层。第一个卷积大层由 2 层小卷积层组成。这两层小卷积层都是由 64 个大小为 3*3，步长为 1 的卷积核组成的。卷积方式都是 0 填充（在图片周围填充 0，保持卷积出来的图像大小和输入图像大小一致）。在每一个小卷积层中，卷积提取出来的 64 张图片会经过批标准化处理。批标准化后的输出会经过 ReLU 激活函数，然后经过 Dropout 操作。我们选取的 Drop 概率为 0.5。卷积大层的输出会经过池化核大小为 2*2，步长为 2 的最大池化层（Max-Pooling Layer）。在这里，64 张被提取出来的图片大小会被缩小。
2) 接下来的卷积大层操作与第一步相同。具体参数为：
   第二卷积大层：由两层小卷积层组成，每层小卷积层都由 128 个大小为 3*3，步长为 1 的卷积核组成。
   第三卷积大层：由四层小卷积层组成，每层小卷积层都由 256 个大小为 3*3，步长为 1 的卷积核组成。
   第四卷积大层：由四层小卷积层组成，每层小卷积层都由 512 个大小为 3*3，步长为 1 的卷积核组成。
   第五卷积大层：由四层小卷积层组成，每层小卷积层都由 512 个大小为 3*3，步长为 1 的卷积核组成。
3) 第五卷积大层的最大池化输出后，会被重构为一个行向量，这个操作被称为 Flatten。之后会进入第一个由 4096 个神经元组成的全连接层。全连接层的输出会经过首先经过批标准化处理，然后通过 ReLU 激活函数，最后进行 Dropout 操作。

Dropout 操作后的输出会进入到一个完全相同的由 4096 个神经元组成的全连接层，进行相同的流程操作。最后，会进入一个由 3 个神经元组成的全连接层。这是将前面所有的操作结合，输出对三个类别的预测值。这一层的输出并不会经过任何处理，直接进入到 Soft-max 层处理。Soft-max 的作用是，将对三个类别的预测值，转换为对三个类别的预测概率。

4) 之后我们通过交叉熵函数计算出 loss 值，并通过反向传播来更新所有没有被 Dropout 的参数。

2.5.5 Mini Batch K-Means

当数据量很大的时候，K-Means 算法会耗费大量内存，训练速度也会变得非常慢。因此我们采用 MiniBatchKMeans 算法。它的思想就是对数据进行随机抽样，每次使用所有的数据中的随机子集来计算。这样就能够节省很多内存，但是也会导致准确率的损失。

MiniBatchKmeans 继承自 K-Means，因此 MiniBathcKmeans 本质上还是属于 K-Means 的思想。从构造方法和论文能看到 MiniBatchKmeans 需要的各种参数的含义，了解这些参数会在实现算法的时候有很大的帮助。batch_size 是每次选取的用于计算的数据的子集大小，一般选为 100。

MiniBatchKmeans 算法采用小批量的数据子集减小计算时间的同时，仍试图优化目标函数。每次训练算法时，采用这些随机抽取的数据子集进行训练算法，能够大大缩小计算时间。与传统的 K-Means 算法相比，它减少了 k-均值的收敛的时间。与此同时，小批量 k-均值产生的结果，一般只略差于传统的 K-Means 算法。

该算法的迭代步骤有两步：

1) 从数据集中随机抽取一些数据形成小批量，把他们分配给最近的质心。
2) 更新质心。与 K-Means 算法相比，质心的更新是通过使用每一次随机抽取的数据子集。对于每一个小批量，把小批量里的数据根据距离分配给离其最近的质心，通过计算分配到该质心的数据的平均值来更新该质心。随着迭代次数的增加，这些质心的变化量是在逐渐减小的。我们会持续这两步，直到质心的变化量降低到了我们设置的阈值，或者达到指定的迭代次数。更新的具体步骤如下：
    1) 任意选取 3 张照片为初始中心点 C_1，C_2，C_3；
    2) 对于任意的照片 X_0 至 X_n，分别与 C_1,C_2,C_3 比较，根据就近原则对照片进行标注；
    3) 对于所有标记为 i（$i \in \{0,1,2\}$）的点，重新计算 C_i，新的 C_i 为所有样本点的平均值；
    4) 重复(2)(3)，直到所有 C_i 值的变化小于给定阈值或者达到最大迭代次数。

2.6 训练结果

2.6.1 平均脸

我们用图 2-3 所示的平均脸来直观描述我们使用的训练集的情况。平均脸是通过

对对应国籍的所有图片的每一个像素进行计算平均值得到的。

图 2-4 平均脸 从左到右为依次为中国、日本、韩国

2.6.2 正确率

我们使用了前文提到的方法去预测测试集的标签。裁剪人像和未裁剪人像的测试集正确率如图 2-5 所示。

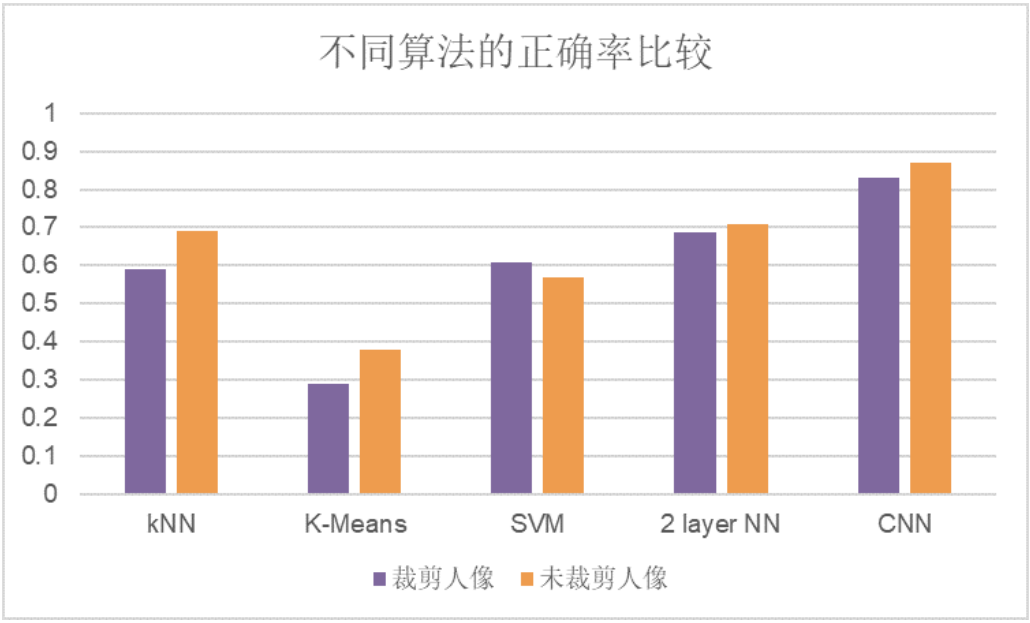

图 2-5 不同算法的正确率对比

从柱状图中我们可以看到，CNN 算法的正确率最高。除了未裁剪人像的 k-NN 算法和 2-layer NN 的正确率(Z=1.957<1.96, 显著性水平 0.05 自由度 2050)和裁剪人像的 k-NN 算法和 SVM 算法的正确率(Z=1.841<1.96,显著性水平 0.05，自由度 2050)外，其余算法都能通过显著性检验（显著性水平 0.05），在算法的正确率上存在显著的区别。因此，CNN 算法的正确率优于两层神经网络，两层神经网络优于 k-NN,k-NN 优于 SVM。且这三种算法的正确率远高于随机猜测的正确率的数学期望。

但是 K-Means 的正确率和随机选择的数学期望接近，无法对测试集做出合理的判断。也就是说，人脸间没有足够大的差异供聚类算法确定有效的中心点。聚类算法可能不适合人脸识别问题。

值得注意的是，除支持向量机外，其他算法模型在未裁剪人像的训练集上的正确率高于在裁剪人像的训练集上的正确率。这个和去除噪音能提高正确率的直觉相反。我们会在第 3.4 节对该问题展开讨论。

第三章 结果分析

3.1 假设检验

我们用机器学习的方法对中、日、韩三国的人脸进行了分类。其中，CNN 在裁剪人像中的正确率达到了 86.9%，其余算法在裁剪人像后的训练集的正确率也都高于随机猜测的正确率，所以我们认为中日韩三国人之间可能存在差异。

3.2 和人进行比较

在机器视觉能否超越人脸的识别能力上，我们选择了 alllooksame.com 网站的测试问卷结果来进行对比。Alllooksame.com 随机选择了 18 张中国人、韩国人、日本人的照片，设计了一个对中国人、韩国人和日本人进行分类的测试。网站上的平均正确率为 7/18，略高于随机猜测的数学期望 1/3。我们对于同样的测试集，用 CNN 进行了测试，正确率达到了 12/18，显著地超越了人工识别的平均水平。也就说，机器视觉可以在没有明确特征的分类问题中超越人类的水平。

3.3 算法间差异的比较

由 2.5.2 部分可知在不考虑时间限制的情况下，CNN 算法有最好的效果。但是，我们也需要注意两种神经网络算法都需要注意不同的训练时间有显著的差异：

算法	KNN	K-Means	SVM	2-layer NN	CNN
训练时间	30分钟	30分钟	30分钟	5-6小时	10小时以上

图 3-1 不同算法在 GTX1060 上的训练时间

图 3-1 显示了不同算法在 GTX 显卡上的训练时间。两层神经网络和卷积神经网络都需要消耗大量的时间。而 kNN,k-Meanss 和 SVM 算法的训练时间有十分相似的训练时间，基本都在半小时及以内。

3.4 裁剪人像对于正确率的影响

我们通过 Z 检验来比较裁剪人像对于正确率的影响。从图 3-2 中可以发现，裁剪人像和使用原图片的正确率有显著的区别。

	kNN	SVM	2 layer NN	CNN
Z	-6.7	2.6	-1.47	-3.41
差异程度	非常显著	非常显著	不显著	非常显著

图 3-2 裁剪人像对于正确率的影响

我们认为，出现这样的原因是由于我们照片集中来源于学校。学校之间的背景存在明显的区别。对于全图进行训练的时候，分类器更倾向于通过背景，也就是噪声来进行训练。

第四章 总结与展望

综上所述，机器学习在按照国籍对于人脸进行分类的问题上有不错的效果，能超过人类的平均水平(7/18)。中日韩三国国民的人脸可能存在区别。CNN 算法的正确率在测试集中能达到 83.1%。由于对全图像进行分类的时候，算法会更倾向于依据背景（噪声）而不是人脸进行分类，所以未裁剪人像的正确率普遍高于裁剪人像的正确率。

本文还有一下问题有待探索：

1) 本文在进行分类时，只考虑了按照中国人、韩国人、日本人分为 3 类，而没有再根据男女，分为 6 类进行研究。
2) 本文的照片来源相对单一，实际图片的数量较少。。
3) 本文由于篇幅和时间等原因的限制，并没有对未裁剪的原因进行深入的实验和研究，这是本文研究的一个遗憾。

致　谢

本论文是在青岛大学陈宇老师的悉心指导下完成的。感谢陈宇老师在毕业设计过程中的建议与指导，使我们得到了很多知识，积累了科研的经验，让我们受益匪浅。

岁月匆匆，我们在青岛大学的四年学生生涯正式结束。回首这四年，正是有各位同学和老师的热情帮助、悉心指导才能使我们顺利完成学业，受益匪浅。论文完成之际，谨此表达我的谢意。在此特意感谢陈宇老师，周强老师对我们专业知识的指导，让我们了解并熟悉了计算机这门有趣的学科。

还要感谢的是我们的父母，感谢他们无私的爱，感谢他们支持我们求学的脚步。

感谢侯晓凯师兄、何熙师兄、金睒师兄等量子人工智能实验室的师兄师姐，在科研学习中和本科毕设中对我们的帮助与指导。师兄们对科研的热情深深的感染了我们，让我们对科研有了全新的认识和无限的憧憬。

特别感谢温丁丁同学，给我们免费提供了高性能计算机，为我们的毕业设计省下了很多时间。

感谢评审组的老师们，为我们的论文提出了宝贵的意见。

最后，再次感谢家人和各位老师、同学、朋友对我们的帮助，我们会在以后的工作与生活中更加努力。

参考文献

[1] Mitchell T M. Machine learning (mcgraw-hill[J]. Machine Learning, 1997.

[2] W. Bledsoe, et al. Man-Machine Facial Recognition. Technical Report, RTI:22, Panoramic Research Inc., PaloAlto, USA, 1996

[3] M. Turk, A. Pentland. Eigenfaces for recognition [J]. Journal of cognitive neuroscience, 1991, 3(1):71-86

[4] D. Goldberg, Genetic Algorithms in Search, Optimization, and Search, Optimization, and Machine Learning, Adison-Wesley, 1989

[5] Paul Viola, M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features, IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[6] Y.Taigman, M.Yang, M.Ranzato, et al. DeepFace: Closing the Gap to Human-Level Performance in Face Verification[C] // Conference on Computer Vision and Pattern Recognition (CVPR), 2014:1701-1708

[7] F.Schroff, D.Kalenichenko, J.Philbin. FaceNet: A unified embedding for face recognition and clustering[C]. Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on. IEEE, 2015:815-823.

[8] Hastie T, Tibshirani R. Discriminant Adaptive Nearest Neighbor Classification[M]. IEEE Computer Society, 1996.

[9] D.E. Rumelhart, G.E. Hinton, R.J. Williams. Learning Internal Representations by Error Propagation [M]// Parallel Distributed processing: explorations in the microstructure of cognition, vol. 1. MIT Press, 1986:318-362

[10] Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift, https://arxiv.org/abs/1502.03167

[11] Haas Cascades face detection library,http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

[12] Artificial neural network：https://en.wikipedia.org/wiki/Artificial_neural_network

[13] Convolutional neural network：https://en.wikipedia.org/wiki/Convolutional_neural_network

[14] Support vector machine：https://en.wikipedia.org/wiki/Support_vector_machine

[15] K-Means: https://en.wikipedia.org/wiki/K-means_clustering