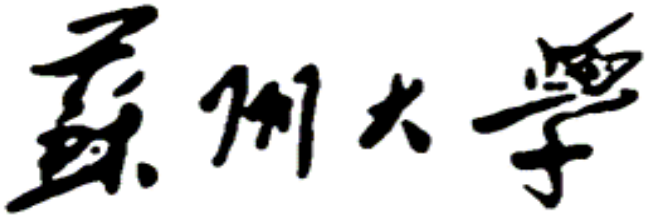

本 科 毕 业 设 计（论 文）

学院(部)	计算机科学与技术
题　　目	新闻视频中的logo检测与识别的设计与实现
年级	大四
专业	计算机科学与技术
班级	1
学号	0827401004
姓　　名	沈智闻
指导教师	刘纯平
职称	副教授
论文提交日期	2012/5/7

目　录

摘　要

随着计算机网络的普及，人们可以越来越多地接触到各种各样的视频。在这个趋势之下，对视频的所有权的保护也就变得越来越重要。所以在网络视频中我们经常可以找到 logo 的身影。Logo 起到了对所有权申明作用的同时也包含着视频相关的各种信息。如果可以借助计算机对这些 logo 进行自动检测与识别，那么在面对大量视频的时候工作就会变得简单很多。

本文就将描述一个视频 logo 检测与识别的具体设计与实现。首先对整体的算法流程进行描述。接下来展示算法中各种部分的具体实现及改进。这些部分为以下几点 1. 应用帧差法进行 logo 检测 2. 应用神经网络进行 logo 检测 3. 应用颜色直方图的 logo 识别 4. 再接着对一组数据进行测试，得出检测和识别的结果 5.是对整个设计的总结和展望。

关键词: 视频 logo 检测与识别，图像分类，帧差法，神经网络，颜色直方图

Abstract

With the spreading of network, people are getting used to meet different videos everyday. Under this circumstance, the protection of video ownership is getting higher position as well. That is why we can find logos in most online videos. Logos serve the work of ownership declaration as well as of containing video related information. If we can use the help of electronic computers with automatic logo detection and recognition, the work with huge amount of videos will become much more like a pleasure.

This article will describe a design and implementation of video logo detection and recognition. First, we will see the go-through of the algorithm. Second, individual part of the algorithm will be illustrated. At the same time, improvement suggestions and ides will be given. These parts are listed as following: 1. Logo detection with frame difference, 2. Logo detection using neural network, 3. Logo recognition with color histogram.. Third, we feed the implementation with a set of data then show the result. The last, a summary and future expectation will be settled.

Keyword: Video logo detection and recognition, image classification, frame difference, neural network, color histogram.

前　言

Logo 的识别与检测是一个图像分类与识别问题。图像识别，是利用计算机对图像进行处理、分析和理解，以对不同模式的目标和对象进行的识别技术。随着计算机图像识别技术的发展以及大家对 logo 检测与识别研究的加深，出现了应用于不同场合的 logo 检测与识别技术。文献[1]中提及了 logo 检测与识别在文档处理方面的重要性，它可以清晰地告诉文档的来源以及相关信息。在文献[2]中作者提出了一种应用于 TV 视频的 logo 检测和移除的方法，对于 TV 视频来说，往往会有移除原有电台 logo 再加上新的 logo 的需求。相对于文档 logo 的检测，TV 视频中的 logo 检测拥有源数据多的特点，但同时数据往往会有很多的干扰，处理过程中经常会应用到许多不同的技术和概念。Logo 不仅仅有静态的（图 1），现在也出现了对于动态的 logo 进行检测的技术，如图 2。在基于改进颜色直方图的体育视频 logo 的检测[3]一文中，作者就应用了颜色直方图的相关知识对动态的 logo 进行检测。

图 1 静态的 logo[2]

图 2 动态的 logo[3]

各种各样的技术被应用于 logo 检测与识别，如帧差法[2]、神经网络[4]、SVM(Support Vector Machines)[5]、Fisher Classification[1]、颜色直方图[3]都得到了应用。本文将对视频 logo 检测与识别的设计与实现进行描述。基于 Automatic video logo detection and removal[4]一文中提出的 logo 检测方法，应用 C++在 OpenCV 的环境下对其进行实现与改进。并进一步应用基于颜色直方图的方法对 logo 进行识别。

第一章 概述

1.1 引言

现代的 logo 设计早在 1876 年就出现了，Bass Triangle 是当时最早注册的 logo(图 1.1)。现在，各个公司都会为自己或自己的产品设计 logo，一方面为了宣传，另一方面也申明了 logo 所在产品的所有权。Logo 有成千上万，但能让人们一眼就认出的并不多，这些 logo 除了出现在人们面前的机会比较多以外，它们自己的特点也直到了很大的作用。Logo 的设计者为了吸引人们的的眼球，logo 的颜色组合会选择得尽量不同，形状也会避免采用过多过复杂的线条（图 1.2）。视频 logo 也具有相同的特点，而我们正可以 logo 颜色和形态的特点对来进行检测和识别。

图 1.1 最早的注册 logo, Bass Triangle

图 1.2 麦当劳的 logo 具有鲜明的特点

logo 的检测问题是根据 logo 本身的特征，对图像中的像素或者像素块进行分类。而视频 logo 本身又具有着与其他 logo 不同的特点。如视频 logo 本身的位置基本是固定不变的。虽然会有些特殊情况，在如前言中就提到的动态 logo 的位置就一直在变化，但这一特点可以很好地帮助我们对 logo 进行检测和识别。对于人来说，辨别一个 logo 是非常简单的事情，但是如果用计算机来进行 logo 检测和识别就显得没那么容易了，尤其是在数据的特点并不明显，或是所处理的情况过于复杂的情况下。由于这点，大多数的 logo 检测和识别软件都具有一定的针对性，有的专注于文档中的 logo 识别[1]，有的针对体育视频 logo[3]。本课题将会针对新闻类视频进行优化。

相对于视频 logo 的检测问题，视频 logo 的识别问题是一个图像识别问题。根据相应 logo 自己的特征，在已有的数据库中寻找有相似特征的 logo，并输出相关的 logo 信息。

尽管视频 logo 的检测和识别问题对人来说是小菜一碟，少量的工作可以通过人眼来进行很好地识别。但随着信息流通量的日渐增大，各种各样的 logo 也层出不穷。通过人来进行识别工作量非常巨大，也很难对 logo 背后的信息进行有效的提取。如果所需要识别的视频量再增大，那通过人工来进行 logo 检测和识别就显得不太现实了。所以通过计

算机来进行 logo 的检测和识别有着重大的意义。

1.2 研究背景

由于 logo 检测和识别的重要实用意义，人们尝试用各种不同的方法对其进行研究。各种方法都在特定的环境下有着很好的效果。有部分也已经产生了很好的实用价值，尤其是在文档 logo 检测出现大师的文献[1, 7, 8]。早在 90 年代初期，就有大量文档图像分析和识别的科学研究，一些商业应用也应运而生。而对于视频 logo，大家更在意 logo 给视频带来的瑕疵，所以更多的研究是关注于如何将 logo 移除来进行的。而在移除之前就必须先检测 logo，因此相关的技术也得到了很好的发展。在文献[4]中就将 logo 的检测和移除相结合起来，在检测的过程中为图像修复做好必要的准备。此外，还有对于自然场景中的 logo 进行检测的研究[9]。在自然场景下的 logo 检测有些很好的发展前景。在智能手机等移动设备越来越普及的今天，如果可以通过设备上的摄像头对现实场景中的 logo 进行检测和识别，那么我们就可以迅速地得到这个 logo 以及产品的相关信息。如果自然场景中的 logo 检测可以在检测速度和效率上有所突破，甚至可以替代条形码和二维码的功能。

1.3 论文组织结构

全文分为四章。第一章介绍 logo 检测与识别的背景。第二章介绍 logo 检测与识别的相关方法。第三章描述本文应用到的 logo 检测方法，设计和实现过程中遇到的问题与解决方法，方法存在的优点和缺点等。第四章为应用测试，对一组视频数据进行测试，统计测试结果。第五章对课题进行总结并对未来的发展进行展望。

第二章 Logo 检测与识别方法

2.1 Logo 检测方法

2.1.1 帧差法

帧差法是用于检测两帧之间差别的方法，它可以有效地检测一个视频中的不移动物体，由于视频logo的位置和形态大多是不变的，所以帧差法可以被用于视频logo的检测。假设一个具有logo的视频为$V=(v_{i,j,k})_{L\times W\times H}$, $v_{i,j,k}$是颜色通道，L是视频的总帧数，H和W视频的宽和高。两个相邻帧中对应像素的颜色距离由以下公式定义

$$d_{i,j,k}=|v_{i+1,j,k}-v_{i,j,k}|,\ i=0,1,\ldots,L-1. \tag{2.1}$$

我们可以应用所得出的帧差值 $d_{i,j,k}$ 对视频中移动的部分和不变的部分进行区分。最简单的方法就是定义一个数组，当超过一定的阈值之后我们就认定这一个像素发生变化，反之这个像素就发生了变化。这个做法的一个优点就是简单，易于理解，并且非常容易实现。但在它在实际情况中的表示却不近如人意。首先一点这个阈值非常难确定，一个特定的值在一部分情况下可能会有非常好的效果，但换一些数据结果就会变得不理想，如在干扰比较大，噪点比较多的情况下，logo 的像素位置可能会发生偏移，这个时候这种简单的方法会认为所有的像素都发生了变化。再加上我们只能调节这么一个值，在测试数据多，并且覆盖面广的情况下效果差强人意。

虽然上面提到方法效果并不理想，但它为我们提供了一种基本思路：在得到颜色距离$d_{i,j,k}$之后，再对其进行处理及比对。在文献[2, 6]中作者提到了用统计学的方法来进行移动与非移动物体的区分。假设 d(k)为相邻帧的灰度颜色距离，它遵从以下 $N(0,\sigma)$高斯分布：

$$p(d_k \mid H_0)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{d_k^2}{2\sigma^2}\right) \tag{2.2}$$

其中 H_0 表示某一个假设，如假设在像素 k 不改变[6]。我们可以看到 $p(d_k \mid H_0)$只与灰度颜色距离的二次方与方差的比有关，也就是 d_k^2/σ^2。因些我们可以根据 d_k^2/σ^2 来确定像素 k 是属于移动的像素还是非移动的像素。其中 σ 也可以通过事先统计不变区域得到。更进一步，我们可以通过比对一小块区域里的像素变化情况，来确定这个区域是否发生了变化。由于在大多数情况下我们需要得到一个长方形的 logo 检测区域，这个方法可以让我们更方便地得到 logo 的外围长方形以确定 logo 的位置。文献[6]中提出了移动窗口的概念来对一块区域内的像素进行统计，通过计算区域中 d_k^2/σ^2 的和——$\overline{\Delta^2}$ 来对区域进行分类。如果 $\overline{\Delta^2}$ 超过了一定的值 t_α，区域就被划分为改变的，反之则为不改变的。而这个阈值 t_α 则

可以在 $p(\Delta^2 | H_0)$ 已知的情况下通过显著性测试得出(α为显著级数)。图 2.1 可以看出经帧差法处理后的视频效果。

图 2.1 应用帧差法对静态和动态物体的捕捉，$\alpha=10^{-6}$，$t_\alpha=74.5$[6]

除了上面所讲述的用高斯分布统计的方法外，有些文献中还提到了用一些后期处理的方法来对结果进行细致化处理[2, 4, 6]。 如应用 Robert edge detector[4]来选择清晰的背景，以及 maximum a posteriori(MAP)[4, 6]消除结果中独立的点。

结合统计的帧差法可以对视频上动态和静态的物体进行很好的区分。由于 logo 自身位置不变的特性，帧差法可以很好地应用到 logo 的检测问题上。但同时也有着不可避免的缺点，它并不是直接检测 logo 而是检测视频中不变的区域。这样就导致如果视频中大多数的区域都静态的，那么结果就完全是错误的了。

2.1.2 神经网络

神经网络是模仿生物神经网络的数学模型。作为一种数学工具神经网络可以隐性地模拟非线性的关系[4]。在图像识别领域，神经网络也有很多的应用。在文献[10]中，作者应用了神经网络开发出了人脸识别的算法，并在测试中取得了不错的效果。尽管在用神经网络来进行图像的分析听起来有些玄乎，但现在计算机硬件的快速发展对解决这类问题提供了很大的帮助，相关技术和算法和趋于成熟。如图 2.2 就展示了一个神经网络学习的基本结构的数学描述，神经网络的本质是一个寻找最值的问题，如图 2.2 中所示，我们用多层的神经元定义了一个函数空间，并确定了一个目标函数来把所需解决的问题量化，再通过训练算法来在所定义的函数空间中找到使目标函数最优的解[11]。

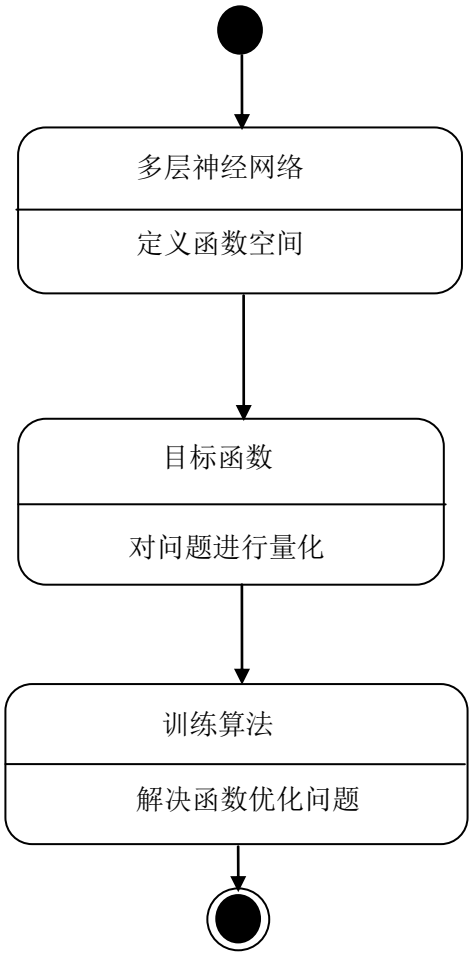

图 2.2 多层神经网络的基本结构[11]

Logo 检测问题对于神经网络来说，可以抽象为一个模式的识别或分类问题。模式识别问题可以理解为把一个具有特征的模式归入先前定义好的类别中的过程[11]。如在引言当中提到的那样 logo 本身的颜色和形状具有特点，我们可以用人眼很直观地对它进行辨别。在电视或是视频上，我们可能一眼就分别出哪块区域是 logo 区域，哪一块不是 logo 区域。一系列的研究和实践证明这种特征是可以通过数学模型来表达的[4]。所以用神经网络来进行 logo 检测是可靠的。

基于多层神经网络的 logo 检测主要可以分为以下几个步骤：

1) 确定神经网络输入的数据格式。神经网络是一个数学模型，最先需要考虑的就是输入的形式。对于 logo 来说可以通过输入灰度值、RGB 值、HSV 值等 logo 的颜色和开头信息。而这个输入的大小的取决则需要通过一些神经网络工作的经验和知识来决定，如果输入的单个数据太大会使得整个神经网络过于臃肿，运算速度低下，而如果输入的单个数据太小则会使神经网络的输出结果不精确。

2) 确定神经网络的规模和复杂程度。正如神经网络的输入，神经网络本身的规模，具体如隐藏层的数目，各个隐藏层中神经元的大小等，对整体的运算速度和准

确性都有很大的影响。规模越小速度越快，随之准确性也随之降低，反之亦然。

3) 寻找合适的训练数据。训练数据的好坏直接决定了神经网络的训练结果，对后期的使用有着很大的影响。准确并且适量的 logo 数据和非 logo 数据的搜索需要额外的关注。

4) 选择合适的训练方法对神经网络进行训练。训练往往需要很长的时间，不适当的训练方法会让这个阶段显得尤为漫长。

5) 使用神经网络进行 logo 的检测。

我们可以看到应用多层神经网络对 logo 进行检测并不容易，有很多的细节需要仔细考虑和权衡。它对神经网络设计者的经验要求也比较高[11]。不过应用多层神经网络进行 logo 检测的优点是非常明显的，首先这种方法没有如帧差法的诸多限制，对动态和静态的背景有着很好的兼容性。其次，神经网络的运算速度也是很快的，虽然它的训练需要花费大师的时间，但一旦得到了训练完成后的神经网络后，那么对单个图像检测的复杂程度是线性的，也就是 O(n), n 表示图像的大小。最重要的是神经网络可以准确而有效地帮助我们区分 logo 区域与非 logo 区域。

2.1.3 SVM 和 Fisher Classifier

SVM是一种机器学习方法, 它依据Vapnik 提出的小样本情况下的统计学习理论可以很好地解决人工智能领域的一些问题，SVM可以解决与神经网络类似的问题，也可以有效地解决神经网络中过学习的问题[5]。所谓的过学习就是在学习后神经网络的结果与训练数据过于吻合以至不能普遍地反映数据特征。相应的SVM也可以用来解决与模式识别的相关问题，所以logo的检测也可以借助SVM来实现。

在SVM在数字水印中的几种应用方式[5]一文中，作者讲述了SVM在数字水印中的一些应用。数字水印作为logo中的一类在处理的难度上要更加大一些，由于数字水印本身与背景是相融合在一起的，应用他的RGB颜色特征来处理并不理想。文中指出“水印技术是依赖于HVS的某些不敏感特性产生作用”。应用SVM对数字水印进行处理可以分为以下三种情况:

1) 利用SVM对数字水印进行嵌入。

2) 利用SVM对数字水印进行提取。

3) 利用SVM对数字水印进行攻击。

对于logo的检测来说最有借鉴意义的就是第二种对数字水印的提取操作。SVM可以模拟人眼的视觉特性[5]。由于视觉特性是很难描述的，所以可以通过机器学习的方法来进行描述并结合SVM和HSV特征来达到提取logo的目的。

由于logo的检测是一个线性的分类问题[1]，Fisher Classifier 也可以被用于logo的检测。

先前介绍的神经网络和SVM方法对训练样本的要求都是比较高的，两者都需要很多的训练数据，并用需要保证训练数据的准确性和可靠性，由于每次训练需要花费的时间都比较多，重复地对训练样本进行改变和训练势必会消耗大量的时间。而Fisher Classifier则有需要有样子数量少的优点，并且它可以进行快速分类，对数据进行直接的描述。

在文献[1]中，作者给出了如下的方法对logo区域进行分类。先用Fisher Classifier将一个特征向量x以方向w映射到一个了平面上，并达到区分数据的效果。

$$y = w^T x \tag{2.3}$$

并定义了两个离散矩阵:

$$S_W = \sum_{i=1}^{2} \sum_{x \in D_i} (x - m_i)(x - m_i)^T \tag{2.4}$$

$$S_B = (m_1 - m_2)(m_1 - m_2)^T \tag{2.5}$$

m_i 表示第i个类的均值向量。Fisher Criteria就会通过将 S_B / S_W 最大化来得到方向 w_0

$$w_0 = S_W^{-1}(m_1 - m_2) \tag{2.6}$$

最后Fisher Classifier的决定函数如下:

$$f_{(x)} = \begin{cases} w_1 & y \geq \min(\hat{y}, (m_1 + m_2)/2) \\ w_2 & Otherwise \end{cases} \tag{2.7}$$

$$\hat{y} = \min w_0^T x_i \quad x_i \in D_1 \tag{2.8}$$

应用Fisher Classifier可能很好地对不同大小的logo进行检测，在文献[1]中作者对文档logo进行检测，准确率达到了84.2%.

2.2 Logo 识别方法

2.2.1 颜色直方图

Logo的识别也就是图片的对比问题，将所提取的logo与数据库中的logo进行比对以找到最合适的结果。由于检测到的logo很可能是已经通过一些放大，缩小，拉伸等变化的，所以如果直接将其与数据库中的logo进行对比效果很差。这时通过颜色直方图的方法可以在一定程度上避免这些因素的干扰。颜色在图像中是比较稳定和直观的内容，颜色直方图对图像的颜色进行统计可以很好地反映图像的特征信息[3]。

在图像的应用中颜色的表达有很多种,一般图像像素都是由RGB来表示的。但是RGB颜色并不可以很好地反映人眼对图像的观察，如光照等因素都会使人眼对RGB颜色产生偏差[3]。此时我们就可以采用HSV颜色空间来对图像进行描述。HSV颜色描述与人眼感觉颜色的方式类似，感知度较强。因此在生成颜色直方图之前往往会先把RGB颜色为HSV，如下所示:

$$H=\begin{cases}2\pi-\arccos\left\{\dfrac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right\} & B\le G\\ \arccos\left\{\dfrac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}}\right\} & B>G\end{cases} \tag{2.9}$$

$$S=\frac{\max(R,G,B)-\min(R,G,B)}{\max(R,G,B)} \tag{2.10}$$

$$V=\frac{\max(R,G,B)}{255} \tag{2.11}$$

其中R、G、B分别代表红、绿、蓝三原色，取值范围为[0,255]。转换后即可生成颜色直方图，如图2.3所示。

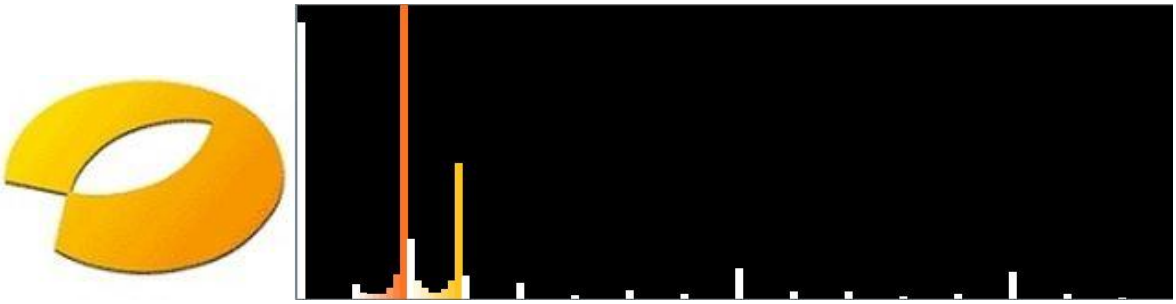

图2.3 一个logo和它的HSV颜色直方图

接下来就是对选择颜色直方图的比较方法，方法有以下几个[12]:

1） Correlation

$$d(H_1,H_2)=\frac{\sum_I(H_1(I)-\overline{H_1})(H_2(I)-\overline{H_2})}{\sqrt{\sum_I(H_1(I)-\overline{H_1})^2\sum_I(H_2(I)-\overline{H_2})^2}} \tag{2.12}$$

$$\overline{H_k}=\frac{1}{N}\sum_J H_k(J) \tag{2.13}$$

其中N为直方图中所划分区域的总数。

2） Chi-Square

$$d(H_1,H_2)=\sum_I\frac{(H_1(I)-H_2(I))^2}{H_1(I)+H_2(I)} \tag{2.14}$$

3） Intersection

$$d(H_1,H_2)=\sum_I\min(H_1(I),H_2(I)) \tag{2.15}$$

4） Bhattacharyya distance

$$d(H_1,H_2)=\sqrt{1-\frac{1}{\sqrt{\overline{H_1}\overline{H_2}N^2}}\sum\sqrt{H_1(I)\cdot H_2(I)}} \qquad (2.16)$$

用颜色直方图来进行对比的方法得出结果后，我们需要确定一个阈值来将确定两个图像（在我们的应用中是 logo)是否是相似的。应用颜色直方图可以对两个图像的颜色进行准确的比较，同时我们也可以应用它对图像的轮廓形状进行比较。如果我们将直方图的输入换成是与形状相关的量，那么直方图就可以反映 logo 形状相关的信息了。具体来说，可以应用如卷积(convolution)的方法找出 logo 的的轮廓边缘，并在边缘上取得各个矢量值并将其作为直方图的输入[13]。这样就可以应用直方图来对 logo 的颜色和形状进行对比，以在数据库中找出最合适的匹配。

2.3.1 关键点+ 决策树法

在两个图片的对比中，如果我们把所有的像素都进行对比会消耗很多的时间，形象地来说在两副 500*500 像素的图像对比中，至少需要 250000 次的循环，这是极度不现实的。在计算机视觉中关键点匹配一个图像匹配的基本方法。在一幅图像上有些部分总是比其他部分包含了更多有用的信息，如物体的边缘等。如果我们可以通过只对比这些关键点来对两副图像进行分析，那么运行的速度就会大大地提高。同时通过关键点的对比，我们也能对经过放大、缩小或拉伸的图像进行有效的控制，可以达到预期的对比效果。这一点对于 logo 的识别是很有效，因为检测到的 logo 与数据库中的 logo 在大小总是会有些差异。SIFT（scale-invariant feature transform）就是一个很流行的关键点匹配方法。SIFT 可以在图像中提取特征，并可以有效地对一个物体在不同观察角度进行匹配，它可以避免图像的大小变化或是旋转变化对图像匹配带来的影响[14]。

虽然关键点匹配的方法很有效，但是对于实际的运用来说还是太慢了，尤其是对我们的 logo 识别来说。不过有的研究提出了用决策树法与关键点结合的方法来对图像进行匹配[15]。文中作者表示他们的方法可以有效地控制关键点对比算法的复杂程度，通过对原图像的学习并结合 Randomized Trees 得到图像的一个分类器，在实际运行的时候通过这个分类器对待匹配图像进行分析，因此在运行的速度是非常快的。

2.3.2 图像感知 Hash 函数

图像感知 Hash 函数 对一张图片进行 hash，基于图像本身的特点产生一个 hash 值[16]。相似的图像生成相似的 hash 值，反之不同的图像生成不同的 hash 值。这样通过对比这些 hash 值，我们就可以知道两个图像是相似还是不同。正如它的名字所表达的那样，它所

产生的 hash 值是基于对图片的视觉认知的基础上的。应用到 logo 识别上，我们可以为数据库中的 logo 生成 hash 值，在需要 logo 识别的时候，将等识别 logo 的 hash 值与数据库中的 logo 的 hash 值相比较就可以得到所需的结果。这一方法的一大优点就是速度非常快，由于对每个 logo 只需存放一个 hash 值，所需要的存储的容量是很小的，这样就可以通过尽量扩大 logo 数据库来提高识别的准确性和效率，而不用担心存储量和生成数据库的速度问题。

在文献[16]中作者对文章中提到的方法进行了实现，并给出了4个感知 hash function 的benchmark：一个基于 discrete Cosine transform(DCT)的方法，一个基于 Marr-Hildreth based的方法，还有基于 radial variance和基于block mean value的 image hash function. 在作者的网页上[17]，可以方便地对这些方法进行demo。

第三章 视频 Logo 检测与识别的设计与实现

3.1 检测与识别方法的选择

在第二章介绍了那么多方法后，接下来就要根据具体的情况来确定如何将其应用到设计与实现当中。我们主要将针对新闻视频进行优化处理。

3.1.1 检测方法的选择

在第二章中提到了应用帧差法、神经网络、SVM 和 Fisher Classifier 来进行 logo 检测。这些方法中帧差法有着易于实现，并对于动态背景的视频有着很好的检测效果，但同时帧差法不能很好处理背景变化程度不大的视频，如图 3.1 可以直观地看到效果。对于神经网络、SVM 与 Fisher Classifier 这三种方法，在原理上有着相似之处，都需要对样本进行特征提取，进而对 logo 进行区分。尽管 Fisher Classifier 在对样本的要求上没有神经网络与 SVM 如此高，处理的速度也较两者快，但这视频 logo 的检测对这两者并不十分敏感。而神经网络易于实现和理解，也有很多现成的神经网络实现和教程。综合以上所述，神经网络似乎是我们最好的选择。

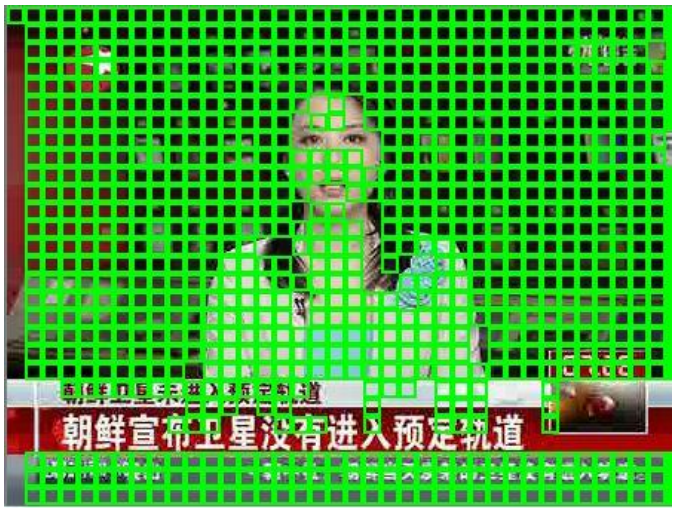

图 3.1 帧差法对背景变化小的视频的处理效果，绿框表示不变化的区域

在实际应用中，总是会遇到一些之前没有考虑过的问题，应用神经网络检测视频 logo 的时候，如果遇到背景变化比较多的情况，那么往往要对视频中的好几帧来进行对比，此时检测的速度就大大地降低了。神经网络视频背景变化不明显的情况下对 logo 的检测有着稳定的准确率。结合帧差法和神经网络方法各自的优势，我们可以将两者结合起来，以形成一个更加完善的视频 logo 检测方法。在文献[4]中作者就得到了将帧差法与神经网络

结合对视频 logo 进行检测的方法，本文之后将给出具体的实现与修改。

3.1.2 识别方法的选择

前文提到了应用颜色直方图、关键点和感知 hash 进行图像识别，这些方法也可以被应用到 logo 的识别当中。在单独使用这些方法对 logo 识别进行简单的实现后，我发现各个方法都会有一些不足之处。

利用颜色直方图可以有效地过滤掉 logo 被放大、缩小、压缩因素的影响，同时也可以有效地识别出 logo 的颜色特征。但如果只对颜色特征识别是远远不够的，两个形状完全不同但颜色相近的 logo 很可能是有相似的颜色直方图，如图 3.2 所示。

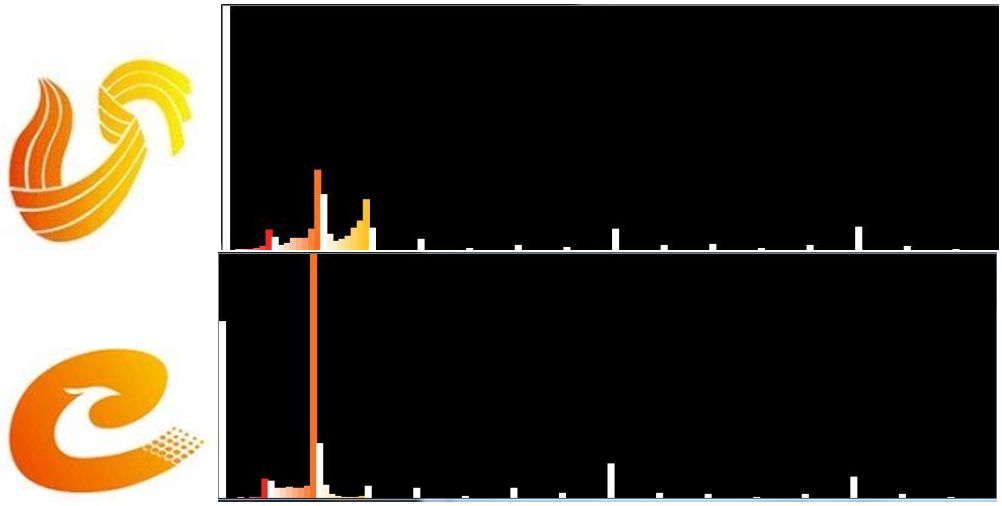

图 3.2 颜色直方图相近，Correlation 结果为 0.837(1 为完全相同)，但形状却不同的两个 logo

尽管可以如上文提到的那样生成一个与 logo 形状相关的直方图，但这种方法是要求源图像和匹配图像相似度很高才可以达到比较好的效果[13]。而我们在 logo 检测过后所提取的带有背景的 logo 很难满足这个要求，由于背景的影响它与数据库中 logo 的直方图相似度远没有想像中的那么高。

关键点方法看人去十分诱人，它涉及到了高级的计算机视觉知识，在很多文献中对它的实用性也给出了证明[13, 14, 15]。再结合决策树法，它的速度也可以得到保障[15]。正是由于这种方法应用到了比较高级的计算机视觉知识，对它的实现也是很困难的[13]。幸运的是在网上我找到了一个关键点 + 决策树法的一个免费软件，名字为 BazAR[18]。其中作者为我们提供了一个可以检测平面物体的基本模块 Garfield，首先我们可以通过 Garfield 对一张图片进行学习，找出关键点，这个阶段被称为离线阶段。在学习之后，我们可以将学习的结果以 randomized trees 的形式存储起来，在需要进行检测的时候再其导入。通过匹配检测图像与源图像的关键点来决定两者是否相似[18]。我进行了一个小的测试来看看这种方法是否适于 logo 识别问题，图 3.3 展示了一个 logo 的关键点。

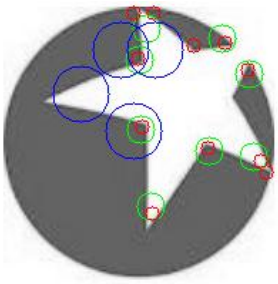

图 3.3 展示一个 logo 的关键点

然而在应用所生成的 trees 对视频中的 logo 进行识别时效果就不这么理想。由于我只做了很简单的测试，关键点 + 决策树法的方法到底能不能很好地应用到视频 logo 的识别上还没有定论。从原理上来说，找到关键点并进行匹配的方法是很合理的。如果可以进一步的研究，相信还有很大的发展空间。

应用感知 hash 来对 logo 进行识别最大的特点就是速度快，在一个完善的数据库的帮助下，有着很好的实用性。但经过一些测试以后发现，单纯用感知 hash 来对 logo 进行识别，对颜色的影响是不敏感的。而颜色直方图对颜色匹配有着很高的准确性，所以如果能将颜色直方图与感知 hash 结合起来对 logo 进行识别，效果会更好。本文也就采用了这种方法，在之后会给出具体的实现。

3.2 Logo 检测的具体实现

在经过测试和对比之后，利用帧差法与神经网络相结合的方法来对 logo 进行检测有着准确率高、覆盖面广的特点[4]。由于在大多数的视频当中背景都是会有变化的，所以我们采用先进行帧差法检测，如果得不到满意的结果再应用神经网络对进行检测。如图 3.4 所示，为算法的流程图。

3.2.1 帧差法的实现

3.2.1.1 像素过滤

在得到公式 2.1 的情况下我们定义一个数组，$A=(a_{m,n})_{W\times H}$，$a_{m,n}=True$ 或者 $False$。在初始化的时候，$a_{m,n}=True$，$m=0,\ldots,W-1;n=0,1,\ldots,H-1$；并同时定义一个阈值 ε，如果颜色距离 $|V_{p,i,j}-V_{q,i,j}|>\varepsilon$，那么该像素 $a_{i,j}=false$，反之 $a_{i,j}=true$. 在比较任意两帧的差之后，对数组 $a_{m,n}$ 进行更新，从而得知各个像素的变化情况。在这里我们采用了比较严格的比较方式，只要任意两帧中某像素的颜色距离超过阈值，那么我们就认定它是变化的。本文采用了将首帧与间隔为 k 的各帧进行比较的方法，其中 k=TotalFrame/MaxIteration + 1。同时采用了 RGB 值来进行颜色距离的计算，也就是对

R,G,B 分别进行比较，只要有一个值的差超过了阈值，那么该像素就被认为是变化的。可以看到这种实现所以采用了严格的比较，在下文中会提到原因。

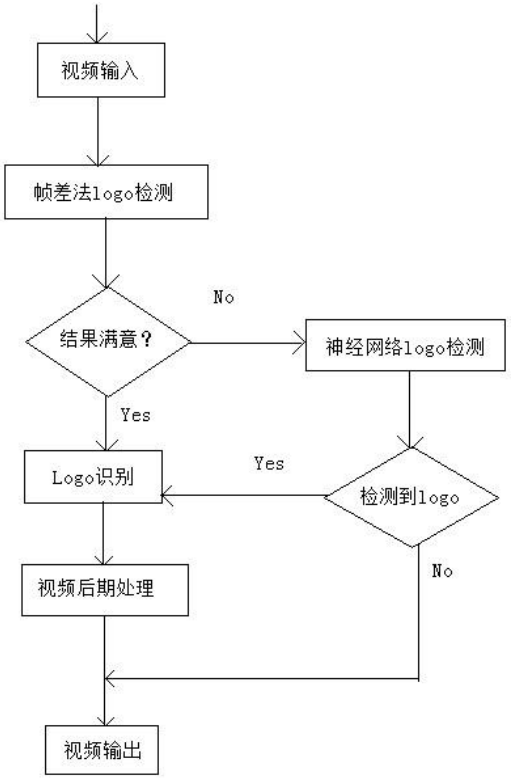

图 3.4 logo 检测与识别的设计流程图

这种方法非常容易实现，但同时也带来了一个问题，在对比得到了数组 am,n 之后，由于各个像素是离散的，我们如何将他组合起来？

3.2.1.2 像素组合

受文献[4]中对图像进行分格处理的启发，我尝试了利用以下的方法：

1）得到 am,n 之后，对其进行划分，形成若干个 N*N 的小区域(设计中采用了 N = 12)，我们把这些小区域记为 Ai,j(i,j 为区域左上角的坐标比上 N 所得)。

2）各个区域中的每个像素标记为 true 的记为 1，标记为 false 的记为 0。将这些值加起来，得到 Sum(Ai,j)，它在区域[0,N*N]中.

3）如果 Sum(Ai,j)大于 M(本文中采用了 10)，那么就将这个加入到集合 S 中。

4）在对所有的小区域 Ai,j 进行处理之后，我们得到了 S，现在就可以将 S 中的区域组合起来。如果 S 中的区域之间的距离小于 Q(本文中采用了 Q=3.1),就 将他们加到同一个集合 Gk 中。

5）Gk 中的区域的个数如果大于 P(本文采用了 P=3)，那么就可以通过计算 Gk 中各个小区域的最小位置和最大位置，来确定组合后的区域的位置和大小。从 而得到 logo 的位置和大小。

现在就可以知道为什么之前就采用如此严格的比较了，由于我们采用了灵活性比较高的组合方法将离散的点组合成块，对于像素过滤过于宽松势必会让这种组合方法丧失可靠性。如果在像素的过滤之后我们仍得到很多错误的点，再加上应用这种宽松的组合方法，那么原本背景变化的区域也有很大的可能被识别为不变的。

3.2.1.3 帧差法相关的其他事项

由于本文针对新闻类的视频进行了一定的优化，在应用帧差法时，也加入了一些限制。比如在检测出的 logo 大小过大时，我们就会把这个 logo 抛弃掉，这是由于在新闻类的视频会一些放置文字的框栏(如图 3.5)，而这些区域也往往是不变化的，他们与 logo 相比一个较大的区别就是大小，由于文字显示的需要，这些区域会比 logo 大不少。还有的情况就是在一个视频中出现了多个不变的区域，那么此时我们就选择将帧差法的检测结果抛弃，进而用神经网络的方法来进行 logo 检测。

图 3.5 红框标出的区域为需要排除的区域

当然如果视频的长度足够，并且我们可以经过尽量多的帧的对比的话我们可以排除大量的错误检测结果，但是视频帧的处理是很昂贵的，过多的帧对比会让帧差法的检测显得十分缓慢。但较长的视频长度对检测往往是有益的，长视频中包含了更多变化区域的信息，由于我们采用了首帧与视频中均匀分布的各帧的比较，对长视频中变化区域的信息有着更好的捕捉效果。

3.2.2 神经网络方法的实现

帧差法对 logo 的检测有着自身难以克服的缺点，即使对其进行优化和限制之后，仍然很难解决部分视频的处理，如图 3.1 所示的背景变化小的视频。而利用神经网络的方法可以有效地解决这些问题，与帧差法形成互补。

3.2.2.1 神经网络基本结构

在文献[4]中作者提出了应用一个 432 个元素的输入层，一个有 20 个神经元的隐藏层以及含有 1 个元素的输出层的神经网络来对 logo 进行检测。logo 自身的颜色经常会采用如红、绿、蓝这些颜色，所以直接应用 RGB 颜色作为神经网络的输入是有效而合理的[4]。而 logo 的形状尽管在整体上有着不少的差异，局部形状却有着很多的共同特性，如图 3.6 所示，尽管三个 logo 整体形状不同，局部上的水平、垂直的线条却是很相似的[4]。

图 3.6 整体形状不同但局部形状却相似的 logo[4]

所以作者引入了 logolet 的概念，将 logo 中 12*12 的小区域称为 logolet，它对 logo 局部区域有着很好的描述。正是基于这种想法，文中将输入层的元素数定义为 12*12*3=432 个。

由于为 logo 检测这一特定问题来进行构造神经网络是非常困难的，首先要求有很好的神经网络使用经验，再者也需要很长的时间对神经网络进行测试来得到合适的参数(我在进行测试的时候，得到一个神经网络的训练结果就需要花费 10 小时以上)。由于我自身经验和知识的匮乏，以及时间上的限制，所以我将以文献[4]中的神经网络结构为基础，进行基于神经网络的 logo 检测的设计与实现。

如文献[4]中一样我采用了（432, 20, 1）的神经网络结构。首先对输入进行以下转换：

$$ScaledInput = -1.0 + Input*2/255 \quad (3.1)$$

将输入层中的 weight，bias 与 ScaledInput 进行线性组合，在输入层与隐藏层之间的 activation function 设定为 tangent sigmoid，如下所示:

$$A(x) = 2/(1+\exp(-2*x))-1 \quad (3.2)$$

将隐藏层中的 weight, bias 与前一层的输入进行线性组合，并将隐藏层与输出层之间的 activation function 设定为 linear。最后将得到的结果进行逆转换:

$$Output = (ScaledOutput + 1)*1/2 \quad (3.3)$$

整体的结构如图 3.7 所示。对于一个输入向量，神经网络的输出越大则这个区域是 logolet 区域的可能性越大，反之可能性越小。

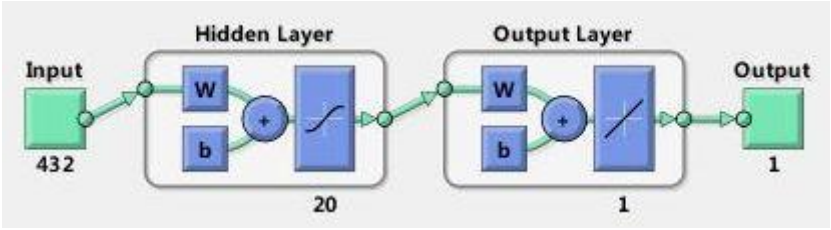

图 3.7 logo 检测神经网络的基本结构

3.2.2.2 神经网络的训练样本

前文提到 logolets 的概念，即 12*12 的 logo 小区域。在为神经网络的训练寻找数据的时候，我们也将图像裁成 12*12 的区域，并将其 RGB 值转换为长度为 432 的颜色向量。由于我们将采用 supervised training，我们需要分别寻找属于 logolets 的 12*12 区域与 nonlogolets 的区域。

正如第二章中对神经网络所介绍的那样，神经网络的训练样本对结果有着至关重要的作用。在训练的过程中，我尝试了不同的训练样本，有的采用了背景干扰比较小的 logo 及非 logo 输入，有的则从实际视频中提取了干扰较大的输入，也有将两者混合起来的。在这些训练样本中我发现应用背景干扰小的输入可以得到更准确的训练结果。

在提取 logolet 区域时，我将各个 logo 放置在一张或是多张背景为白色的图片上，如图 3.8 所示。再将整张图片分为 12*12 的一个个小区域，在这些小区域中过滤掉纯白色的区域，把剩下的转换为长度 432 的 RGB 向量，作为 logolet 输入，target 值为 1。

在对非 logolet 区域进行提取的时候，我准备了若干个不包含 logo 或是类似 logo 特征的视频。同时这些视频也覆盖了比较广的范围，有从演唱会、风景写真等各种体裁的视频中截取的片断。在每一个视频中均匀地提取若干帧，并对其进行与上述 logo 图片相同的操作，得到了 RGB 向量之后，将 target 值设为 0，作为 nonlogolet 输入。

图 3.8 用于 logolet 提取的图片

3.2.2.3 神经网络的训练

在构造完了神经网络以及得到训练数据之后就应该开始对神经网络进行训练了。首先我们确定应用 MSE(Mean Squared Error)来作为我们的 Objective Functional，也是用来衡量训练结果的函数。再接着考虑神经网络的训练方法，最基本就是 BP(Back Propagation)算法。然而在应用 BP 算法的时候，需要计算新的参数与旧的参数之间的差值 delta。计算这个值有很多不同的方法，如 Gradient Descent、Newton's Method、Conjugate Gradient、Quasi-Newton Method 等[11]。在这之中 Gradient Descent 是最容易实现和理解的方法，也是我最先尝试的方法，有如下公式:

$$\varsigma_{i+1} = \varsigma_i - \nabla f(\varsigma_i)\eta_i \quad (3.4)$$

其中 ς_i 表示参数向量(由 weight 和 bias 组合，在我们的神经网络中 weight 有 432*20+20*1=8660 个，bias 有 20+1=21 个，共 8660+21=8681 个参数)，当 i=0 时为初始值，$\nabla f(\varsigma_i)$ 为梯度值，η 为学习速度，可以设为固定值，或随着参数的改变而改变。

虽然 Gradient Descent 的方法很容易理解与实现，占用内存少，每一轮的训练速度也很快。但对我们这样较大的神经网络来说，由于每次训练对整体结果的提升太小而导致很难得到想要的结果。当 η 为固定值时，对它的确定也是很困难的事情，如果设定地太大那么 MSE 会不停地在一定范围内摆动，如果将其设定地太小 MSE 降低又过于缓慢。而如果将 η 设定成跟随着训练的进度而不停改变的话，又很难找到恰当的算法，况且即使 η 可以随着训练的进展而改变，MSE 降低过于缓慢的情况很难得到缓解。不过由于 Gradient Descent 方法所需的计算量很少，很适合将它的结果作为其他方法的初始值来进行应用[11]。

在应用 Gradient Decent 进行神经网络的训练过程当中，同时也发现了另一个问题，由于 Gradient Descent 对神经网络参数的初始化是随机的，在训练的过程中经常会得到局部的最小值。如图 3.9 所示，就展示了一个到达局部最小值的神经网络，红点为其所在的位置，实际的神经网路的情况不会如此简单，多数是在更加复杂的平面上。这个问题可以通过重新选取初始参数来解决，但同时利用这种方法有很大的不确定性，因为下一组随机的参数达到局部最小值的可能性也很大。还有一种方法就是可以通过 Nguyen-Widrow 初始函数来对神经网络进行初始化。在 matlab 的神经网络工具箱中主提供了这种方法[19]，它可以根据我们的输入对神经网络进行初始化，同时也具有一定的随机性。应用这种方法可以将得到相对好的训练结果。

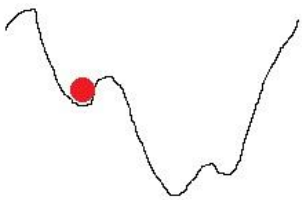

图 3.9 训练结果达到局部的最小值的示意图

从上面的描述可知，采用 Gradient Decent 的方法来对我们的神经网络来进行训练实在是太慢了，而且结果也很难满足检测需求，在几次测试当中发现应用 Gradient Decent 检测得到的神经网络往往都会输出十分相似的结果。此时就需要更加高级的方法来帮助我们进行神经网络的训练，在经过资料查询之后，文献[11,19]发现可以尝试用 BFGS Quasi-Newton Method 来进行训练。其中 BFGS 是对逆 Hessian 矩阵进行近似的方法。

$$\varsigma_{i+1} = \varsigma_i - Gf(\varsigma_i)\cdot\nabla f(\varsigma_i)\cdot\eta_i \quad (3.5)$$

$Gf(\varsigma_i)$ 为逆 Hessian 矩阵的近似，这里为 BFGS 方法。在 Quasi-Newton Method 的前身

Newton Method 中，直接应用逆 Hessian 矩阵来进行训练，而在每一轮中都对 Hessian 矩阵及它的逆矩阵进行计算的开销是非常昂贵的。在这里应用近似的矩阵来对逆 Hessian 矩阵进行代替，可以大大加快运行的速度。这种方法相比 Gradient Decent 来说有着更快的速度，更少的训练次数及更稳定的训练结果[11]。

当然 BFGS Quasi-Newton Method 也不是没有缺点的，在提高了运行速度的同时，它也带来了极高的内存消耗。试想在我们的神经网络中总共有 8681 个参数，一个逆 Hessian 的将由 8681*8681 个数构成，如果在 double 变量的大小为 8 byte 的 c++环境中，一个矩阵所需要的内存大小为 8681*8681*8/(1024*1024) ~= 575MB。而在一次训练中会用到多个这样的矩阵，内存的消耗是巨大的。我在应用 Flood 3 神经网络库对数据进行训练的过程中，由于内存释放以及 c++深度拷贝的问题，经常在第一次的训练当中就遇到内存不足的问题。在后来改用 matlab 进行数据训练的时候，内存的占用也达到了恐怖的 3.5GB，这也可见文献[4]中作者对神经网络的各项参数选择上的用心良苦，如果隐藏层中神经元数据继续增加，或是输入的元素数再增加势必会造成更大的训练难度。

图 3.10 展示了一次应用 BFGS Quasi-Newton Method 进行训练时 MSE 的变化过程。

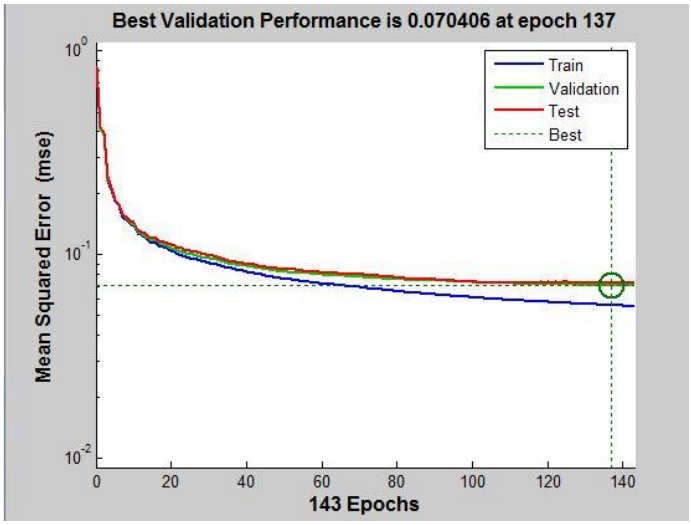

图 3.10 用 BFGS Quasi-Newton Method 进行训练的 MSE 趋势图

以下为应用 matlab 进行神经网络训练的代码:

```
load inputData.data
target = inputData(:, 433)
target = target'
input = inputData(:, 1:432)
input = input'
net = newff(input,target,20,{},'trainbfg')
net = configure(net, input, target)
net.trainParam.epochs = 200
[net tr] = train(net, input, target)
```

其中 inputData.data 为输入文件，由 3.2.2.2 中所描述的方法生成，第一行共 433 个数，前 432 个数为颜色向量，最后一个数为 target 值。

3.2.2.4 神经网络的应用

在经过训练得到神经网络之后，就可以对视频 logo 进行检测了，具体方法如下:

1) 取视频的第一帧，将其划分为 12*12 的小区域。

2) 将各个小区域的 RGB 值提取出来转换为长度 432 的向量，将其作为神经网络的输并得到输出值，若值超过了 0.4 则将其认定为 logolet，反之则为 nonlogolet。由于在最后隐藏层与输出层之间我采用了线性的 Activation Function，使得最后的输出结果出现了小于 0 或是大于 1 的情况，这里将阈值定为了 0.4 而不是文献[4]中的 0.5。

3) 在视频中均匀取几帧重复 1)，2)，将每一位置的区域被归为 logolet 的次数记录下来。若这个次数超过了 T(文中采用了 T=（总循环数+平均次数)/4)，那么就正式将其认定为 logolet，并加入到一个集合 S 中。

4) 应用 3.2.1.2 中像素组合的方法把 logolet 组合为 logo。

图 3.11 为神经网络应用的流程图。

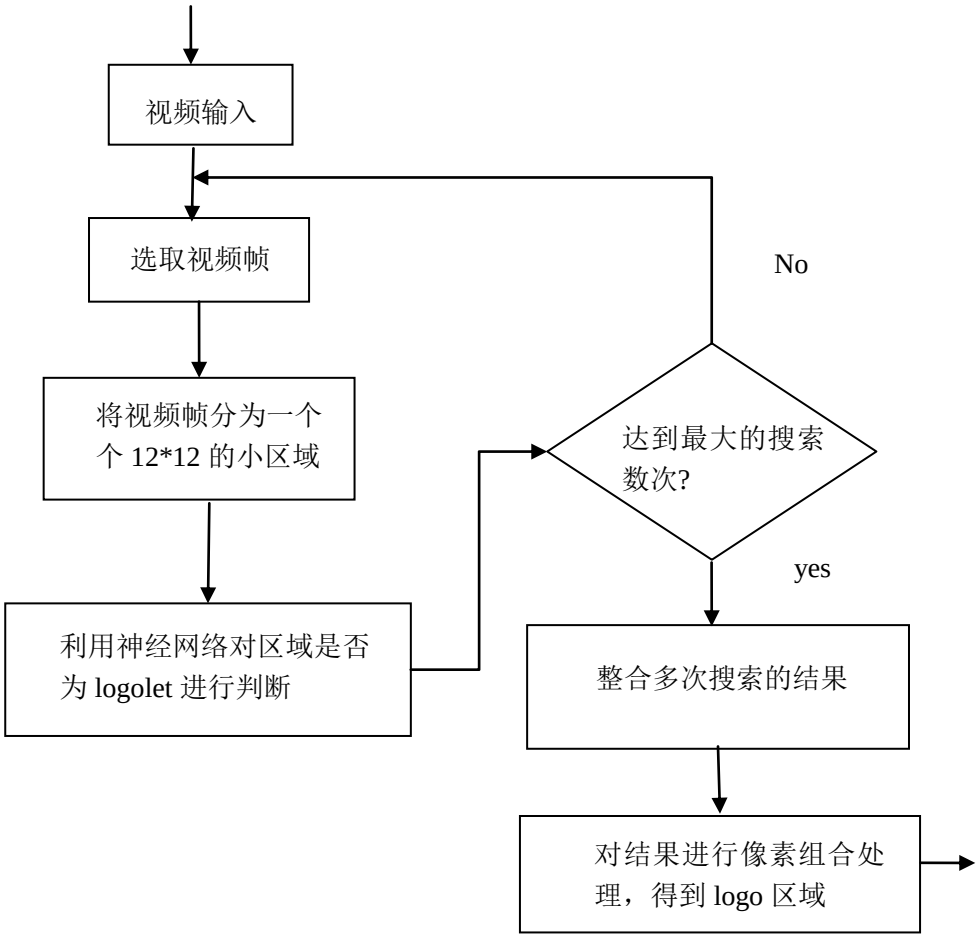

图 3.11 神经网络应用的流程图。

采用多帧的分析可以有效避免背景对神经网络结果的影响，如果只对单独的一帧进行检测那么神经网络的检测结果就不是非常可靠。相应于帧差法中的检测限制，神经网络的实现中也应用到了类似的方法，尤其对 logolet 的位置进行限制。在文献[4]中作者提出了应用统计的方法，利用高斯分布来对 logolet 的位置进行限定，对不同位置的区域都会有一个特定的概率 $P(r_R = 1|l_R)$，也就是给定区域位置，这个区域为 logolet 的概率。由于计算 $P(r_R = 1|l_R)$ 需要大量的数据统计，因此在我的设计中并没有对其实现。我采用了比较简单的限定方法，对视频的边缘与中央位置的区域加了一个 0.3 的权值(由测试所得)，如图 3.12 为加上权值与不加权值的对比。相信如果对位置信息进行统计的话可以达到更好的效果。

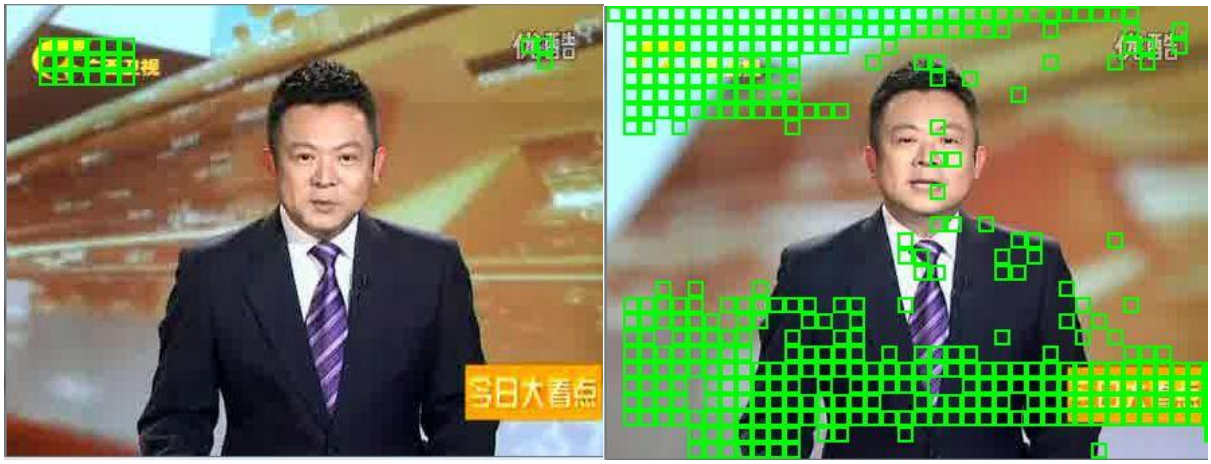

图 3.12 同一视频在对位置进行限制(左图)，与不对其进行限制(右图)的对比

3.3 Logo 识别的具体实现

在检测出 logo 区域之后，接下来要实现的就是对 logo 进行识别。在 3.1.1 节中提到了应用感知 hash 与颜色直方图相结合的方法来进行 logo 识别。接下来就具体描述这种方法。

3.3.1 logo 数据库的构造

由于时间的限制，在对 logo 数据库的构造上我采用了最简单的方法。在 logo 数据库的针对性上也只针对电台 logo 进行了收集。在搜集到一些电视台的 logo 及其名称之后，将其图片存入到一个固定的文件夹中，并以电视台的名称进行命名。在这之后再通过程序来对文件内容进行提取，形成数据库。

3.3.2 利用感知 hash 的 logo 识别

我们主要利用 pHash[16, 17]将 logo 用一个 8 byte 的长整型来表达。利用 DCT(discrete Cosine transform)结合一系列的图形变化来达到对图片的量化表达。具体方法如下[16]:

1) 先将图片进行灰度化处理。

2) 利用[7 7 1 1 1]的 mask 对图片进行卷积处理，并达到中值过滤效果。

3) 将处理后的图片变成 32*32 大小。

4) 计算得到 32*32 的 DCT 矩阵，并通过矩阵运算得到 DCT 矩阵的转置矩阵。

5) 将 DCT 矩阵，处理后的图片以及转置 DCT 矩阵相乘得到 DCT 图像。

6) 利用 64 低频的 DCT coefficient 对 DCT 图像进行 hash, 以得到最后的值。

在得到 hash 值之后，利用汉明距离来对 hash 值进行比较，汉明距离的定义如下：

两个长度为偶数(这里是 64)的值 x 与 y，将其进行位与位的对比，如果某一位不同那么 hamming distance 就加 1（初始值为 0）。

可见 hamming distance 展示了两个值之间位的偏移情况。在计算待检测的 logo 的 hash 值与 logo 数据库中的 hash 值的 hamming distance 之后就可以知道哪个是最合适的匹配。找出最小的 hamming distance，如果这个值大于阈值 28，那么就可以算是匹配成功，否则就找不到匹配。如图 3.13 为两个 logo 之类的比较。

图 3.13 两个 logo 间应用 DCT hash 进行比较，得到的 hamming distance = 36

3.3.3 图像感知 Hash + 颜色直方图

利用图像感知 hash 尽管十分方便，但是由于图像的 hash 值是在应用对图像进行灰度转化之后得到的，相关的颜色信息并不充分。在利用图像感知 hash 进行 logo 识别的时候，经常会出现过多的匹配，其对 logo 颜色的不敏感这是造成这种结果的原因。所以本文采用了颜色直方图与图像感知 Hash 相结合的方法来对 logo 进行识别。

在 logo 数据库构造的过程当中，我们不仅存储 logo 的 hash 值，也将其颜色直方图存储起来。对 logo 识别的算法伪代码如下:

```
match(img)
begin
	imgHash = dct_hash(img);
	imgHistogram = get_histogram(img);
	bestMatchDistance = 100000;
	bestMatchName = "";
	Loop: iterate instance it in logoBase
	{
		distance = hamming_distance(img_hash, it.hash);
		histogramMatch = compare_histogram(imgHistogram, it.histogram)
		if(bestMatchDistance > distance
			&& histogramMatch > matchThreshold)
		{
			bestMatchDistance = distance;
			bestMatchName = it.name;
		}
	}
if(bestMatchDistance > matchDistanceThreshold )
		bestMatchName = "";
	return bestMatchDistance, bestMatchName;
end
```

第四章 Logo 检测与识别的测试

4.1 测试样本的选择

在对测试的视频进行选择时，我尽量包含了所能想到不同情况，并对正反情况都进行了考虑，如背景变化明显与不明显的视频，受干扰比较大的与比较小的视频，不包含 logo 的视频，包含一个 logo 的视频及包含多个 logo 的视频。由于设计与实现主要对新闻类的视频进行了优化，测试视频的选取时也包含了较多的新闻类视频。

4.2 测试结果

在手动对样本进行测试之后得到了以下的结果（表 4.1）。

表 4.1 测试统计结果

logo 总数	22
检测到的 logo 数	20
未检测到的 logo 数	2
检测错误的区域	4
准确检测的 logo 数	13
识别出的 logo 数	6
检测率	91.0%
准确检测率	65.0%
识别率	46.2%

在表 4.1 中我们可以看到 logo 检测率为 91.0%。图 4.2 展示了正确检测与错误检测的情况，从表中我们也可以看到在 logo 总数为 22 的样本中，错检的有 4 个。准确检测率为 65.0%，它表示在检测到的 logo 中，logo 的位置和大小被准确地标识出来的情况，如图 4.1 所示。表 4.1 中的识别率则为在准确检测的前提下，我们的系统可以列出 logo 名字的概率，这个概率为 46.2%。

图 4.1 左图为 logo 被准确识别，右侧为非准确识别

图 4.2 左上角为检测正确的区域，右上角为检测错误的区域

由于时间的原因，这次测试的样本数比较少，只有 20 个视频，而其中也只包含了 22 个问题，只能在一定程度上反映问题。

在我们的实现中 logo 的检测的率是比较高的，也就是说在检测到 logo 都可以被很好地定位。而 logo 检测的过程中出现了一定的错检，为 4 次。这一点也与视频的来源有一些关系，由于视频都来自于 youku 网，所有的的视频上在某特定的时间在左上角和右上角出现了 youku 网的水印 logo，这对于视频 logo 的检测出现了影响。而从中还发现的一个问题就是我们的检测方法对半透明的 logo 并不敏感，往往不能进行比较精确的定位。

而在检测准确的情况下视频 logo 的识别的成功率为 46.2%。在经过观察后发现，如果待识别的 logo 不在我们的 logo 数据库中，那么识别的机率为 0。只有一个不在数据库中的 logo 正确地被我们的识别系统排除。从这一点可以看出如果可以为我们的数据库增加足够数量的 logo(这一点实现的花费并不大，见 2.3.2 节)，那么识别的成功率将会得到提升。

第五章 总结与展望

5.1 课题总结

Logo 检测与识别的方法千姿百态，本文扬描述的设计和实现只利用了其中的一小部分。方法越多就会为我们带来更多的选择，相信也会得到更加强大和完善的 logo 检测与识别的实现。虽然综合了很多的方法，但 logo 的检测率，尤其是 logo 的识别率并不理解，在很多方面仍需要很多改进。

5.2 后续研究展望

从第四章的测试结果可以看到，本文所用的实现仍然有着比较大的局限性。虽然总结了不少的方法，但由于资源和时间有限，真正进行尝试的方法不过一半，希望有机会在 logo 的检测与识别或是其他的领域上进行尝试。在文章的结尾写下目前为止想到的一些改进策略和展望，希望对这方面的研究有所帮助：

1) 在利用帧差法对移动背景过滤的时候，可以尝试用再高级的方法，如利用统计有关的知识来进行更系统和准确的过滤[2, 6]。

2) 文中提到的像素组合方法依然有改进的空间，在编码进行测试的时候有些特殊的情况仍然很难完美地应对。

3) 利用神经网络对 logo 进行检测时，可以尝试对检测区域的位置信息进行更全面的考虑，结合贝耶斯分类的方法对区域进行分类[4]。

4) 完善 logo 数据库。

5) 尝试利用关键点的方法来对 logo 进行识别，相信这种方法有着很好的应用前景[14, 15]。

参考文献

[1] G. Zhu, D. Doermann. Automatic document logo detection[J]. Ninth International Conference on Document Analysis and Recognition, 2007, 2: 864-868.

[2] Katrin Meisinger, Tobias Troeger, Marcus Zeller, Andr´e Kaup. Automatic TV logo removal using statistical based logo detection and frequency selective inpainting[C]. Chair of Multimedia Communications and Signal Processing, 2005.

[3] 向志敏，景晓军，孙松林．基于改进颜色直方图的体育视频 Logo 检测[OL]. http://www.paper.edu.cn/index.php/default/releasepaper/content/200807-301, 2008 ,07, 16.

[4] Wei-Qi Yan, Jun Wang, Mohan S. Kankanhalli. Automatic video logo detection and removal[J]. Multimedia systems, 2005, 10(5): 379-391.

[5] 佟雨兵，常青，张其善，吴今培．SVM 在数字水印中的几种应用方式[J]．计算机应用研究, 2005, 22(3): 147-149.

[6] T. Aach, A. Kaup, R. Mester. Statistical model-based change detection in moving video[J]. Signal Processing, 1993, 31(3):165-180.

[7] Tuan D. Pham.Unconstrained logo detection in document images[J]. Pattern Recognition, 2003, 36: 3023-3025.

[8] The Anh Pham, Mathieu Delalandre, Sabine Barrat.A contour-based method for logo detection[C]. 2011 International Conference on Document Analysis and Recognition, 2011: 718-722

[9] Jim Kleban，Xing Xie, Wei-Ying Ma. Spatial pyramid mining for logo detection in natural scenes[C].Multimedia and Expo, 2008 IEEE International Conference, 2008: 165-180.

[10] Rowley, H.A., Baluja, S. Kanade. Neural network-based face detection[J]. IEEE trans. pattern anal. mach. intell. 1998, 20(1): 23–38.

[11] Roberto Lopez. Flood user's guide[OL]. www.cimne.com/flood, 2010, 08.

[12] http://opencv.willowgarage.com/documentation/cpp/imgproc_histograms.html

[13] http://stackoverflow.com/questions/843972/image-comparison-fast-algorithm

[14] David G. Lowe. Distinctive image feature from scale-invariant keypoints[J]. International Journal of Computer Vision, 2004, 60: 91-100.

[15] Vincent Lepetit, Pascal Fue. keypoints recognition using randomized tress[J]. EPFL Computer Vision Laboratory, 2006, 28(9): 1465-1479.

[16] Christoph Zauner. Implementation and benchmarking of perceptual image hash

functions[D]. Hagenberg: eingereicht am Fachhochschul-Masterstudiengang Sichere Informationssysteme, 2010.

[17] http://phash.org/

[18] http://cvlab.epfl.ch/software/bazar/index.php

[19]http://www.mathworks.com/help/toolbox/nnet/ref/initnw.html

致谢

在最后我希望感谢周围的人对我的毕业设计和毕业论文的帮助。感谢我的父母为我提供了一台不错的机器来跑那些似乎永远都跑不完了的神经网络训练程序;感谢我的指导老师——刘纯平老师对我的耐心指导，同时我也要表示一下 sorry,上次我们讨论了很久的关于 logolet 位置概率的算法到最后我也没有能实现,由于自身的松懈与最近如潮水一般的各种事务，我把它掠一边去了；同时感谢一下耐心阅读这篇文章的老师或是同学，读一篇不好的文章比写一篇确实是要辛苦多了；感谢参考文献中所有的作者，没有他们这篇文章也不可能出现。最后祝福一下所有在做毕业设计和写毕业论文的大四同学和我自己——答辩顺利！