基于深度卷积神经网络的视频活动识别

作者：邱杰

摘 要

人体活动识别是让计算机能够自动的理解未知视频或者图像序列中正在发生的事件，是实现计算机的视觉智能的重要手段。其研究成果可以应用到视频检索、智能视频监控以及智能人机接口等领域；同时人体行为识别研究涉及人工智能的诸多领域，是当前计算机视觉研究的热点和难点。基于视频的活动识别旨在通过分析视频内容达到对活动进行识别的目的。本文通过查找参考文献，分析得出卷积神经网络对空间信息的提取有显著的效果，循环神经网络对时间信息的提取有很大的优势，而长短期记忆网络解决了循环神经网络的长期依赖问题。在此基础上，通过结合两者的优势，进行了对基于视频的活动识别问题的研究。首先使用深度卷积神经网络提取视频的空间特征，然后使用“卷积-长短期记忆网络”提取视频的帧间的依赖关系，最后经过一个输出层映射到类别上，预测出样本的类别属性。网络参数的优化方面，使用“Softmax-交叉熵”作为网络的损失函数和 ADAM 优化算法。依赖于 TensorFlow 的自动反向传播，经过不断的循环训练，使得网络参数的估计值不断的逼近于真实值。最后该网络对 10-分类的视频活动识别问题最高可以达到 92%的测试集准确率，平均准确率为 85%。

关键词：视频活动识别；卷积神经网络；循环神经网络；长短期记忆网络；光流；视频预处理

Abstract

Human action recognition is to enable the computer to automatically understand the unknown video or image sequence is happening in the event, is to achieve the computer's visual intelligence an important means. The research results can be applied to the fields of video retrieval, intelligent video surveillance and intelligent human-computer interface. At the same time, human behavior recognition research involves many fields of artificial intelligence, which is the hotspot and difficulty of current computer vision research. The video-based action classification is intended to achieve the purpose of classifying actions by analyzing video content. In this paper, we find that the Convolution Neural Network is very effective in extracting spatial features by searching for references. The Recurrent Neural Networks has a great advantage in extracting the time feature, while the Long-Short Term Memory network solves the problem of long-term dependence of the Recurrent Neural Networks. On the basis of the above, this paper studies the problem of video-based activity identification by combining the advantages of both. First, the spatial features of the video are extracted by using the Deep Convolution Neural Network. And then use the "Convolution - Long-Short Term Memory Network" to extract the inter-frame dependency of the video. Finally, an output layer is mapped to the category to predict the category attribute of the sample. In the optimization of Network parameters, the use of "Softmax - Cross Entropy" as the Network loss function and ADAM optimization algo-rithm. TensorFlow rely on the automatic back-propagation, through continuous loop train-ing, making the network parameter estimates continue to approach the real value. Finally, the network of 10-classified video action classification problem can reach 92% test set ac-curacy, the average accuracy rate of 85%.

Keywords: video action classification; cnn; rnn; lstm; optical flow; video pre-processing

目 录

第 1 章 绪论

1.1 课题背景

计算机的计算能力非常强大，但是却对人的依赖性很强，需要人工去刻意指定计算机要进行的工作，为此人们思考着如何让计算机具有像人一样的自主学习、理解和分析的能力。早在二十世纪 50 年代，人们提出了“人工智能”的概念。人工智能的一个子集，机器学习得到的长足的发展，机器学习的子集。深度学习属于机器学习的范畴[1]，目前，深度学习对人们生活的影响越来越大，越来越多的人工智能产品在推向公众。

基于计算机视觉的人体动作识别算法主要包括动作表示以及动作分类两个步骤。人体动作如何表示涉及到如何编码人体动作信息，它对于后续的人体动作分类十分关键。理想的人体动作表示方法不仅要应对人体外观、尺度、复杂背景、视点以及动作执行快慢等因素的影响，而且要包含足够的信息提供给分类器将各个动作类型区分开来[4]。在视频活动识别中，人体动作的编码主要由视频表示，视频由基于图像的帧序列表示，最终需要提取的特征是帧的空间特征和序列的时间特征。

视频图像分析识别经历了多个发展阶段。最开始的灰度图像识别，比如 MNIST 手写数字识别。之后是彩色图像识别，比如 Google 让计算机在 YouTube 上自助学习识别“猫”，而在 ImageNet 静态图像识别大赛中，深度学习网络对静态图片的识别率已经超过了人类。随着计算机和信息科学的发展，计算机视觉方向的研究已经成为了新的方向。视频分析的最大难点是在时间信息的提取和数据量太大，这两个难点阻碍了网络的搭建和训练。

1.2 视频活动识别研究现状

有关人体识别方法中使用的动作表示及动作分类方法，近几年已经开展了这方面的研究，在日本、西方发达国家(如美国、英国)的专家，已经展开了大量相关的研究：美国宾夕法尼亚州大学的三维人体头部及脸部跟踪系统采用有限元素模型实时跟踪人的脸部动作；联想、IBM 及 Microsoft 等国内外高知名大公司及一些中小型企业也正逐步将人脸识别、行为识别、手势识别应用于商业领域，如：手机移动终端开发、门禁、停车场、监狱等；另外，斯坦福大学、普林斯顿大学、牛津大学等国际

知名大学也做了相应的理论研究[2]。在北京奥运会、上海世界博览会中，一些企业开发的安防系统中也都得以应用[3]。1997 年，IBM 研发的超级电脑深蓝挑战国际象棋世界冠军获胜。2016 年，Google DeepMind 的 AlphaGo 战胜了顶尖职业棋手，成为第一个不借助让子而击败围棋职业九段棋手的电脑围棋程序，2017 年 5 月，在中国使用经过升级的AlphaGo，战胜了围棋世界冠军和多个顶尖职业棋手。2016年，Google推出专为深度学习定制的 Tensor Processing Unit(TPU) 1.0 芯片，它能高速的做推理，2017 年，推出 TPU 2.0，升级后的 TPU 既能够做推理还能够做训练，与之配套的 TensorFlow 深度学习库也迎来了 1.0 稳定版本。

基于深度学习理论和卷积神经网络的静态图像识别领域研究已经取得革命性进展，并在商业环境中得到应用。动态图像的识别仍然存在很大的困难，视频中人类动作识别是目前的基本挑战之一。本文主要基于卷积神经网络和循环神经网络，通过大规模的训练得到网络参数，进而通过该网络对测试集视频数据进行识别分类。

1.3 本文主要内容与结构安排

第二章中分四小节介绍了卷积神经网络的构成。第一节介绍了卷积运算，包括一维卷积、二维卷积和多维卷积，在最后介绍了计算卷积的方法。第二节介绍基本卷积神经网络的结构和网络各层的介绍：卷积层、激活函数层、池化层、全连接层以及输出层。第三节介绍了训练一个网络的基本方法，使用交叉熵代价函数建立损失函数，得到前馈网络的反馈网络，最后介绍学习速率的影响，如何选取合适的学习速率。第四节中引申出深度卷积神经网络，介绍了深度过深带来的一些问题及其解决方案。

第三章中分三小节介绍循环神经网络的构成。第一节介绍计算图及计算图的展开，介绍了计算图和计算图的展开图各自的应用方向。第二节介绍了基本的循环神经网络、双向循环神经网络和深度循环神经网络，介绍了他们各自的应用方向及劣势。第三节介绍了门控循环神经网络、能解决循环神经网络的长期依赖问题的长短期记忆网络(LSTM)和它的改进版，基于卷积的 LSTM 网络。

第四章中分四小节介绍了数据预处理的通用方法和针对于视频的数据预处理方法。第一节介绍数据预处理的作用。第二节介绍数据预处理的归一化方法：简单缩放、消除直流分量、特征标准化。第三节针对活动视频数据介绍了光流特征的提取，涉及到稀疏光流和稠密光流。第四节描述了视频预处理中的一些问题及解决方案：视频长度的调整，针对样本数量不足的随机裁剪和翻转，另外还有一个完整的视频预处理的

流程。

第五章中通过三个小节介绍了视频活动识别的深度神经网络和实验结果。第一小节介绍了网络中用到的子网络模块：卷积子网络、Conv-LSTM 子网络、输出子网络。第二小节介绍了四个网络模型各自的优势、劣势和适用场景。最后通过一个小节展示实验结果和分析。

第 2 章 卷积神经网络

2.1 卷积

卷积神经网络中的卷积和信号处理中的卷积不太相同，在卷积神经网络中不进行数据翻转，并且不仅仅有序列的卷积，还有二维图像和三维空间的卷积。计算机是数字的，只能表示离散的序列，因此机器学习中使用的数据都是离散数据。

2.1.1 一维卷积

一维卷积即序列的卷积。两个序列的卷积就是输入信号经过一个系统后输出信号，输入信号被描述为输入序列，系统描述为卷积核或滤波器，也是一个序列，输出信号就是这两个序列的卷积。

离散一维卷积公式如下：

$$(f * g)[n] = \sum_{m=0}^{M-1} f[n - M + 1 + m]g[m] \tag{2-1}$$

其中，$g(m)$的自变量m满足条件：

$$0 \leq m < M \tag{2-2}$$

定义域之外的值使用零参与计算，不妨设两个序列的长度$N(f(n)$序列的长度)，M的关系为：

$$N \ll M \tag{2-3}$$

那么，$g(m)$称为卷积核或滤波器，$f(n)$称为输入信号。

2.1.2 二维卷积

二维卷积通常使用在图像处理中。和一维卷积类似，都是输入信号经过系统后的输出信号。二维卷积中的输入、卷积核和输出都是二维张量。

离散二维卷积公式如下：

$$(f * g)[m, n] = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f[m - M + 1 + i, n - N + 1 + j]g[i, j] \tag{2-4}$$

其中，g的形状是$[M, N]$，定义域之外的值为零。不妨设f的形状远大于g的形状，则f称为输入图像，g称为卷积核或滤波器。

2.1.3 多维卷积

多维卷积和一、二维卷积类似，离散多维卷积公式如下：

$$(f*g)[m,\dots,n,\dots,o]=\sum_{i=0}^{M-1}\dots\sum_{j=0}^{N-1}\dots \sum_{k=0}^{O-1} f(m-M+1+i,\dots,n-N+1+j,\dots,o-O+1+k)g(i,\dots j,\dots,k) \tag{2-5}$$

其中，g的形状为$[M,N,O]$，定义域外的值设为零。同二维卷积，f称为输入多信号，g称为卷积核。

2.1.4 计算卷积

上述公式中，由于不对序列进行翻转，因此其计算不能使用快速傅里叶变换。公式中具有大量的可同时进行的计算，因此使用 GPU 的大规模并行计算能力可以大大加速上述卷积计算。

在计算时，卷积核不能移位到输入信号定义域之外。这样，对于输入信号有两种处理方式：(1) 不进行处理，即不在定义域外补零；(2) 在定义域外补零以扩展定义域。以一维卷积为例，输入序列长度为N，卷积核序列长度为M。对于方式(1)，输出信号的长度O为：

$$O=N-M+1 \tag{2-6}$$

对于方式(2)，可以这样做使输出序列长度和输入序列长度相等：在序列左边补

$$a=[(M-1)/2] \tag{2-7}$$

个零，在序列右边补

$$b=(M-1)-a \tag{2-8}$$

个零。这样使输入序列变成了长度为

$$(N+M-1) \tag{2-9}$$

的序列，这样输出信号的长度O就和输入信号N的长度相等。根据需要，还可以有其他的补零方案，比如左右都补

$$(M-1) \tag{2-10}$$

个零，这样输出序列的长度为：

$$O=(N+M-1) \tag{2-11}$$

2.2 卷积神经网络

卷积神经网络的主要组成有：卷积层，激活函数层，池化层(即下采样)，全连接层[5]。对于一个完整的卷积神经网络模型，还具有一个输出层，用于把信号映射到分类的类别上。为了避免过拟合的问题，往往使用一种叫 Dropout 的算法加在全连接层和输出层之间，它的算法很简单：按指定的概率随机丢弃一部分的输出(置零)，它可以看作是一种非常有效的正则化[6]。使用了 Dropout 层，就可以不使用其他的正则化了。

以图像卷积为例，实际上就是给定一幅图像和一个卷积核，根据卷机窗口进行像素的加权求和。在信号处理的卷积运算中，卷积核参数是已知的，比如图像处理中的高斯模糊、边缘检测算子等。这些卷积核都是根据特定需求人工设计出来的，相当于人工提取特征。而在卷积神经网络中，卷积核参数是未知的，我们给定一个随机的初始值，然后经过训练，自动学习出参数，得到一个需要的卷积核，这称之为自动提取特征。

2.2.1 卷积层

卷积层是以卷积运算为核心的层。但是对于三通道的 RGB 彩色图像，每个通道都是一个特征图，也即 RGB 彩色图像包含三个特征图。使用卷积神经网络做二维彩色图像分类时，对于通道维，是以全连接(见 2.2.4)的方式与输出通道维连接在一起的，输出可以得到更多的特征图，即通道数增加。

2.2.1.1 **局部感受野** 在出现卷积神经网络之前，人们都是使用的全连接的神经网络，而全连接的参数非常多，训练很困难，并且容易导致过拟合的问题。对于图像来说，启发于生物神经科学的研究，人眼是通过一个局部的感受野去观察外界图像的，因此人们认为图像的空间联系是局部的。由此，我们认为卷积神经网络的神经元不需要像使用全连接的 MLP 网络那样对图像全局都做连接，只需要感受局部的图像。这样不但大大的减小了参数的个数，有利于加快计算机的运算速度，而且能有效的防止过拟合的问题，提高分类器的准确率。

如图 2-1，通过局部感受野，我们把全局感受野的参数由 24 减小到了 12 个，减少了 50%的参数。

2.2.1.2 **权值共享** 经过局部感受野优化后，参数得到了有效减少，但是对于一个超长的序列，参数个数仍然非常庞大。人眼观察世界还有一个特点，对于每一个感受

野，都是通过同样的神经元前馈给大脑的，这个原理可以用于卷积神经网络，用于进一步减少参数，我们成为权值共享。因此我们再使用另一个方法：权值共享，对依然庞大的参数进行优化减少。

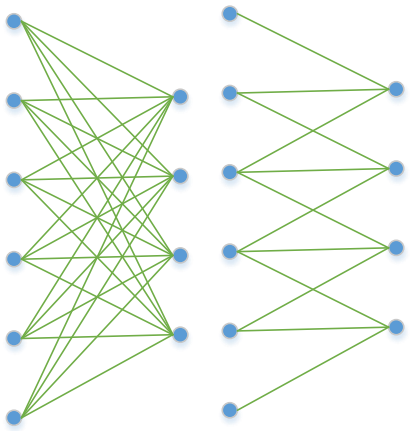

图 2-1 局部感受野和全局感受野比较。输入序列长为 6，输出序列长为 4，图中每条连线表示一个参数。a) 全局感受野，参数个数为 4*6=24 个，b) 大小为 3 的局部感受野，参数个数为 3*4=12 个。

如图 2-2，通过权值共享，我们在局部感受野的基础上把参数减少到了 3 个。通过实验，这样的处理方法并不会明显降低模型的效率，反而因为参数的大量减少，减少了大量的计算，为创建更大规模的分类器打下了一个非常好的基础。

2.2.1.3 **卷积层** 经过上面的讨论，人们选择了用卷积来做这样的优化。卷积具有局部感受野和权值共享的特点，卷积核就是共享的局部感受野。实际上全连接是卷积的一种特殊形式：卷积核形状的大小和输入信号相同，并且使用方式(1)对输入信号进行处理。卷积核形状看做是感受野，如果卷积核形状大小和输入信号相同，如全连接，也即全局感受野，反之卷积核形状大小小于输入信号，如一般的卷积，那么就是局部感受野。

2.2.2 激活函数层

卷积可以简单地描述为：

$$y = w * x + b \tag{2-12}$$

其中w为卷积核，x为输入数据，b为偏置项，$*$为卷积。本质上来说，卷积和全连接一样，都是线性函数，而对于分类器，通常被分类的数据不是线性可分的，为了更好

的拟合非线性可分的数据，需要在卷积神经网络中加入非线性的映射，称之为激活函数。激活函数有很多种：

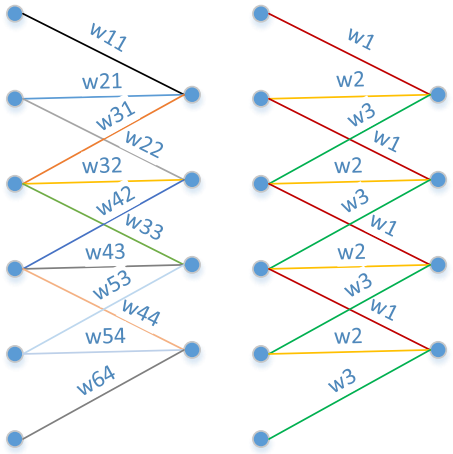

图 2-2 采用权值共享和不采用的比较。输入序列长为 6，输出序列长为 4。a)未采用权值共享，每条连线均为一个参数，参数个数为 3*4=12 个，b)采用权值共享，每个输出节点的三个参数均相同，参数个数为 3 个。

2.2.2.1 Sigmoid **激活函数** Sigmoid 激活函数常用的有两种：Logistic-Sigmoid 和 Tanh-Sigmoid。Logistic-Sigmoid 函数以S函数：

$$S(\mathrm{t}) = \frac{1}{1+e^{-t}} \tag{2-13}$$

为基础建立的。它把输入数据从$(-\infty \sim +\infty)$映射到$(-1 \sim +1)$。Tanh-Sigmoid 函数以双曲函数：

$$y(\mathrm{t}) = \tanh t \tag{2-14}$$

为基础。它把输入数据从$(-\infty \sim +\infty)$映射到$(0 \sim 1)$。它们的共同点是对0附近的信号增益较大，而对远离0的信号增益很小。从生物神经科学上看，这样的映射的0附近信号类似于神经元的兴奋状态，远离0的信号类似于抑制状态，因而在网络自动学习中，就能让数据重点特征集中在0附近区域，非重点分散于远离0的区域。

2.2.2.2 ReLu **激活函数** ReLu 基于 Softplus 激活函数和生物脑神经元激活频率函数[7]。对 Softplus 函数求导即可得到 Logistic-Sigmoid 激活函数。$Softplus$激活函数图像如图 2-3，公式如下：

$$Softplus(\mathrm{x}) = \ln(1+e^{x}) \tag{2-15}$$

ReLu 激活函数图像如图 2-4，公式如下：

$$ReLu(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \quad (2\text{-}16)$$

ReLu 激活函数还有很多的优化版本，比如：RRelu，Leaky ReLu，PReLu 等。

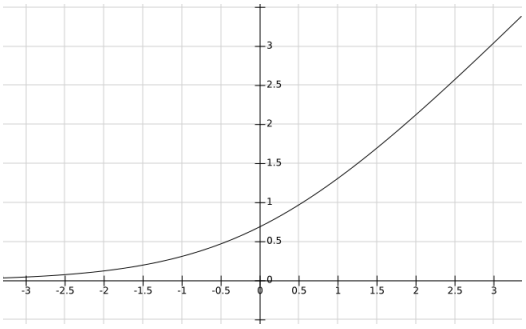

图 2-3 Softplus 激活函数图像。可以看到 Softplus 激活函数对负区有一个截断效果，类似于整流器。

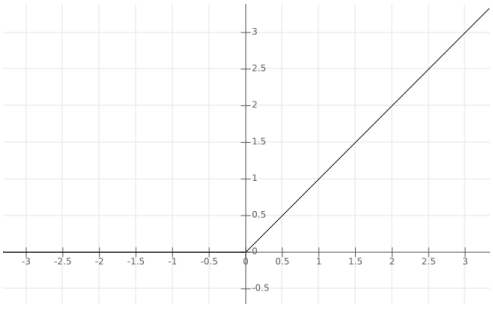

图 2-4 ReLu 激活函数图像。可以看出 ReLu 激活函数就是一个整流器，直接把负区信号给置零，正区信号原样保留。

2.2.2.3 **激活函数的选择** 在 MNIST 手写数字识别实验中，可以发现使用 ReLu 激活函数的效果远远好于 Sigmoid 激活函数。使用 Relu 激活函数在 epoch=50 时，验证集错误率就能降到 1.1%以下，而使用 Tanh-Sigmoid 激活函数在 epoch=150 时的验证集错误率降到 1.4%以下。进行预训练后，还会有更高的准确率。

2.2.3 池化层

池化函数使用某一位置的相邻输出的总体统计特征来代替网络在该位置的输出。通常，人们使用池化层是为了在保留主要特征的情况下减小数据量，以此减小运算量。池化还具有使网络对少量的平移具有不变性，但是却不能保证对大幅度平移的不变性。当我们只关心某个性质是否出现而不关心它的具体位置时，这个少量平移不变性能够有效的抵抗干扰，获得需要的特征，比如检测一张图片中是否有猫时，我们只

关心有和没有，而不关心猫在图片中的位置。然而，当需要关注特征的具体位置时，则池化将会弱化模型的效率。常使用的池化函数有：最大池化，平均池化等。池化层可以看做是一个无限强的先验：这一层学到的函数能够具有评议不变性。当学到了这样的一个函数时，池化层能够极大的提高网络的效率。

因为池化是综合了某个位置与其相邻位置的反馈，这使得池化层输出能够少于输入。这个方法能够很大程度的提高网络的计算效率，不妨设综合了k个像素作为输出，则下一层的输入数量比池化层的输入减少了$(k-1)/k$。

2.2.3.1 **最大池化** 最大池化是使用某个位置的相邻数据区域内的最大值来替代这个位置作为输出的算法。一般使用的最大池化使用的相邻数据的范围和滑动步长相等，相当于对数据进行了最大值下采样，如图 2-5。

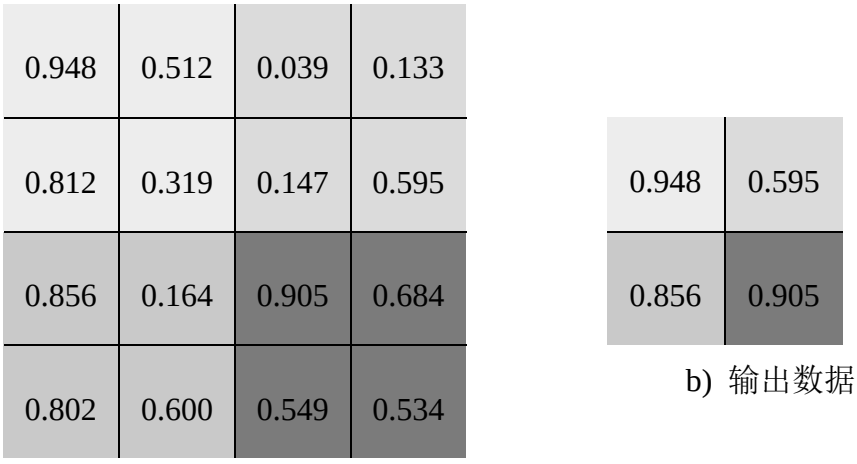

图 2-5 最大池化。当池化范围和滑动步长纵向横向均为 2 时，得到 b)图的数据。

$$\begin{cases} b_{11} = \max(a_{11}, a_{12}, a_{21}, a_{22}) \\ b_{12} = \max(a_{13}, a_{14}, a_{23}, a_{24}) \\ b_{21} = \max(a_{31}, a_{32}, a_{41}, a_{42}) \\ b_{22} = \max(a_{33}, a_{34}, a_{43}, a_{44}) \end{cases}$$

2.2.3.2 **平均池化** 平均池化和最大池化类似，但是是使用某个位置的相邻数据区域内的平均值作为这个位置的输出的算法。同样可用于下采样，综合池化区域内的所有数据作为输出。

2.2.4 全连接层

全连接层作为MLP神经网络的主要部件，在卷积神经网络中成为了辅助的构件。

全连接层使用二维矩阵乘法运算实现：

$$y = \boldsymbol{W} \cdot \boldsymbol{x} + b \tag{2-17}$$

其中$\boldsymbol{W}$是权重，$\boldsymbol{x}$是输入，b为偏置，·为矩阵乘法。这里要求输入是一个列向量，如果输入x不是列向量，需要整理成列向量作为输入。

全连接层是一个非常常用的层，但是由于该层的参数过多，有时甚至能占整个网络参数个数的 80%以上，并且全连接层会丢失掉一部分图像空间位置信息，因此在某些网络中，会使用其他算法替换掉全连接层。在 ResNet 残差网络等网络中，使用了全局平均池化算法替换了全连接层，这样极大的减少了参数[8]。

2.2.5 输出层

输出层作为网络的末端，具有把输入数据映射到各个类别上的作用，最终判断输入数据属于哪一个类别或哪几个类别。

对于单标签多分类问题，一般输出层使用 Softmax 回归模型:

$$\theta_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} \tag{2-18}$$

其中，θ_i表示输入数据是类别i的概率，η_j表示输入数据在类别j上的映射输出，总共有k个类别。得到的输出可以看作是每个输入数据是各个类别上的概率。最终取概率最大的类别作为该输入数据的预测输出。

对于多标签多分类问题，不能使用 Softmax 回归模型。一种方法是对类别训练一个二分类模型：样本在这个标签集合内或不在标签集合内，最终集合多个网络的输出作为最终的输出。另一种是使用其他的分类模型。

2.3 训练网络

网络的训练简单地说是更新参数的过程。通常人们把数据分为三个部分：训练集，验证集和测试集。训练集用于估计网络参数，验证集用于确定网络结构或者控制网络复杂程度的参数，测试集用于检验最终的模型性能。为了简便，在这里只分为两个部分：训练集和测试集，去掉验证集是因为通常我们建好的网络结构已经固定。

对网络的参数进行更新需要先算出倒数，然后根据学习速率进行更新。计算倒数需要用到反馈网络。单层卷积的神经网络结构如图 2-6 所示，通过计算它的反馈网络，可实现对网络参数的训练、学习。

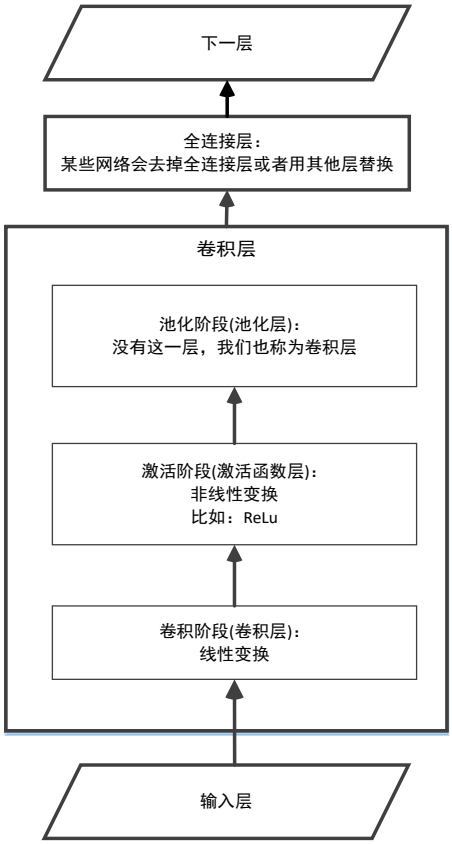

图 2-6 一个卷积神经网络。在下文中，我们把卷积层、激活函数层、池化层综合起来总称为卷积层。其中没有池化层我们也成它为卷积层。

2.3.1 损失函数

损失函数是用来估量模型的模型给出的对每个类别的预测概率$\boldsymbol{y}(x)$ (Softmax 后的值)和真实标签$\boldsymbol{y}_(x)$ (one-hot)的不一致性，它的值是一个非负的标量实值，通常使用：

$$L\big(\boldsymbol{y}\_(x),\boldsymbol{y}(x)\big) \tag{2-19}$$

来表示，损失越小，表明预测标签越接近于真实标签。

最常用的损失函数是交叉熵代价函数：

$$\boldsymbol{C}=-\frac{1}{n}\sum_{i=0}^{n-1}\big[\boldsymbol{y}\_(x_i)\ln\big(\boldsymbol{y}(x_i)\big)+\big(1-\boldsymbol{y}\_(x_i)\big)\ln\big(1-\boldsymbol{y}(x_i)\big)\big] \tag{2-20}$$

其中，$\boldsymbol{C}$表示交叉熵，n表示样本数，i为样本标号，x_i表示第i个样本数据。这里的标量与向量的算术运算是标量与向量的每一个元素都进行相应的算术运算，向量与向量的算数运算时对应于元素做相应的算术运算。然后对得到的$\boldsymbol{C}$以求均值的方式转换

成标量，得到总的损失函数：

$$L = \text{reduce\_mean}(\boldsymbol{C}) \tag{2-21}$$

交叉熵是参数的函数，对交叉熵求偏导即可得到交叉熵对参数的导数，用于指导参数的更新。在单标签多分类问题中，我们只需要真实类别的预测概率最大化即可，因此，一个等效的替代方案是:

$$\boldsymbol{C} = -\frac{1}{n}\sum_{i=0}^{n-1}\left[\boldsymbol{y}\_(x_i)\ln\left(\boldsymbol{y}(x_i)\right)\right] \tag{2-22}$$

上式中，相对于式(2-20)，去掉了$\left(1-\boldsymbol{y}_(x_i)\right)\ln\left(1-\boldsymbol{y}(x_i)\right)$，这更容易计算，只需要计算对应于真实类别的预测概率值即可。

2.3.2 反馈网络

得到了损失函数后，用总的损失函数对每一个参数求偏导数，偏导函数即是反馈网络。训练过程就是使用前馈网络算出损失函数值，然后使用反馈网络算出本次训练步的参数的偏导数，结合学习速率更新参数。当前的很多深度学习库都具有自动推导反馈网络的功能。我们只需要构建好前馈网络即可自动得到反馈网络。

对参数的优化有多种优化算法，最基础的是梯度下降算法(GD)，也叫最速下降算法。它直接根据偏导最大值方向更新参数，以最快的速度逼近最优值。实际应用中由于数据量太大，不能每次训练都把所有数据都输入进网络，因此得到两个小改动的算法：随机梯度下降算法(SGD)和批量梯度下降算法(BGD)。随机梯度下降算法每次只更新一个样本输入的参数，批量梯度下降算法每次更新一个批量输入的参数。另外，还有 Adagrad [9]，Adam [10]等算法。

梯度爆炸，也就是梯度太大，如果梯度过大，可能本次训练步得到了一个很优质的解，对于整个训练集来说，可能是一个非常差的解。使用梯度裁剪的方法能够解决这个问题：预先设置一个最大的梯度值，当某次训练的梯度超过该值时，使梯度等于该值，这样就能解决梯度爆炸的问题。

2.3.3 学习速率

学习速率即是更新参数的速度，学习速率乘上梯度，然后更新参数。学习速率得当能够更快更准确得到性能优良的参数。学习速率过大，将可能跳过参数的最优值或者参数在最优值附近有较大震荡；学习速率过小，可能导致学习速度过慢，更严重的

是陷入局部最优值无法跳出，从而不能学习得到全局最优值。

通常的学习速率的设置方法有：(1) 训练开始时给一个较大的学习速率，因为此时可能离最优解比较远，(2) 经过一定的训练步后，减小学习速率，因为此时的参数可能已经在最优解附近了，(3) 在经过一定的训练步后，再次减小学习速率，(4) 当损失函数不再降低时，此时可能处于局部最优解也可能处于全局最优解，可以先保存此时的参数，然后设置一个较大的学习率，跳出局部最优，可能得到另一个更优的解。

2.4 深度卷积神经网络

深度卷积神经网络可以简单地定义为：两个及以上的卷积层叠加起来的网络。不妨设卷积层定义为$f(x)$，那么深度卷积神经网络可以定义为：

$$f\left(f\left(f(\cdots)\right)\right) \tag{2-23}$$

深度卷积神经网络中，某些卷积层没有池化阶段。研究表明，提取更多的特征可以通过加深卷积网络层数来实现。但是越深的网络计算量越大，就越难以训练，而且并不是堆叠越深的网络就能够得到更好的网络，当深度达到一定程度后，训练网络会遇到梯度消失和梯度爆炸的问题，这个问题可以通过标准的权重初始化和正则化来解决。

对于图片的分类问题，一般每经过一层，就会生成更多的特征图，并且得到的每个特征图的分辨率降低，一般总的数据量会越来越小，这样有利于减少全连接层和softmax 层的参数个数，如图 2-7。

2.5 本章小结

卷积神经网络是对 MLP 神经网络的改进，由于 MLP 神经网络的不能学习到空间信息，参数过多的缺陷，人们基于卷积设计了卷积神经网络，它能够克服基于全连接的 MLP 网络的缺陷。在卷积神经网络的基础上，通过增加卷积层的方式构成深度卷积神经网咯，这能够极大地提高网络的效率。然而，大多数卷积网络都需要在输出层前加一个全连接层用于把卷积网络学到的特征映射到一维张量上，然后在输出层再映射到类别上。信息在进过全连接层时，可能导致两个问题：过拟合和损失空间信息，因此近年来很多网络逐渐放弃了全连接层而用其他结构代替，比如 ResNet 使用全局池化代替全连接，GoogleNet 使用全局池化替代部分全连接。

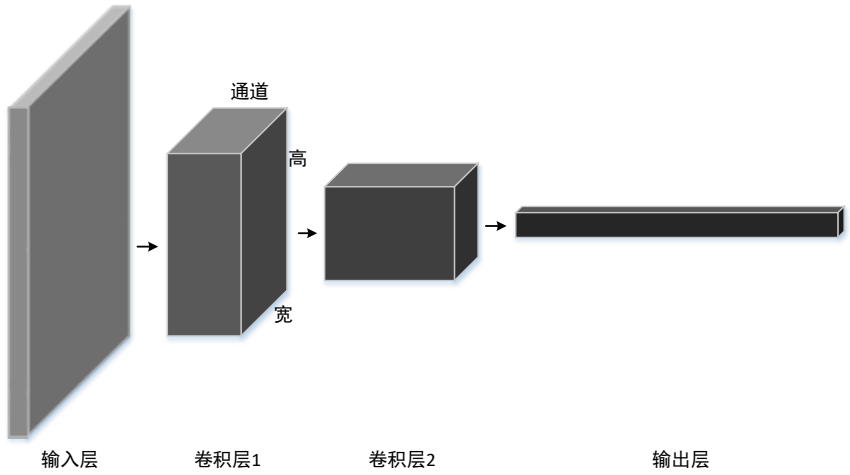

图 2-7 深度卷积神经网络。特征图的宽和高在不断的减小，而深度(通道数)在不断的增加。

然而深度卷积神经网络也不是无所不能，当深度加深到一定程度时会出现新的问题。卷积神经网络要想学习到时间信息，可以把时间维作为第三个维度，使用 3D 卷积核学习时空信息[11]。一个针对于学习时间信息的网络，循环神经网络，可解决卷积神经网络学习时间信息困难的问题。

第 3 章 循环神经网络

3.1 计算图

计算图是形式化一组计算结构的方式，和框图类似，涉及到将输入和参数映射到输出和损失的计算，比如：神经网络。不含分支判断结构和循环结构的计算图称为静态计算图，含有分支判断结构和循环结构的计算图称为动态图。上一章的卷积神经网络就是静态图，而本章的循环神经网络因为涉及到循环结构，因此是动态图。在计算时，循环结构的动态图需要展开成静态图来计算，分支判断结构的计算图需要指定或动态的选定某一个分支，仍然需要转化成静态图。

在 Python 语言中，深度学习的框架有很多，Caffe，Torch，TensorFlow 等都给给出了 Python 语言的 API，并且他们都实现了自动推导反向传播网络和自动求导更新参数。同时他们都能够利用 GPU 的并行计算能力，大大加速网络的训练。

TensorFlow 中还包含一个 TensorBoard 可视化工具，能够实现对输出和网络的可视化显示。TensorFlow 能实现几乎所有的神经网络的训练，比如 MLP 神经网络，深度卷积神经网络和循环神经网络等[12]。它的计算是先构建一个静态的张量流图，然后从输入端获得数据，经过张量流图后，在输出端获得数据。静态图的最大优点就是计算速度比动态图快很多。然而，由于它只支持静态流图，因此对于某些动态的网络，比如循环神经网络的支持会弱一些，目前 Google 官方已经在对这方面的弱点做了改进。

3.1.1 计算图的展开

对于循环图来说，可以简单的重复循环体即可实现展开，如图 3-1。循环图的循环次数一般是根据输入动态的调整的，通常会设置一个上限值，因为循环次数过多会消耗大量的计算资源，并且对网络性能没有明显的提升。在展开图中，每个展开结点都有相同的结构和共享的参数。

3.2 循环神经网络

循环神经网络是以一个 RNN Cell 为循环体的网络，如图 3-2，RNN Cell 可以比作是生物神经细胞，循环比作是细胞记忆[13]。以识别视频为例，视频可以看作是帧

序列，每个循环输入一帧数据。由于 RNN Cell 会向下一个 time-step 传递一个信息，因此循环神经网络可以学习到时间信息，也就是能学习到上一帧(也可以称为时间步)和当前帧的关系特征。

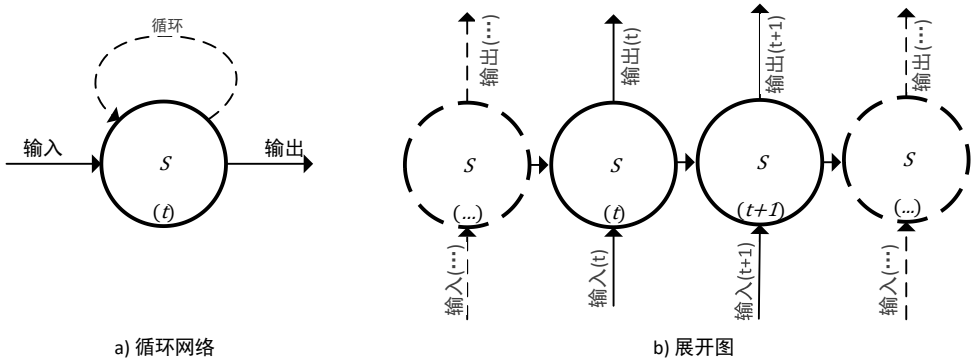

图 3-1 循环图的展开。a)为一个循环图，把它展开后可得到 b)图。

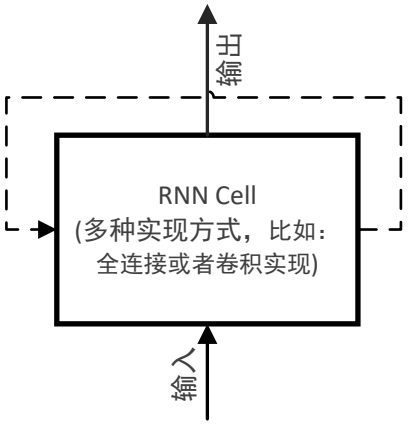

图 3-2 循环神经网络。RNN Cell 为循环体，虚线表示状态传递。

循环神经网络解决了卷积神经网络不能学到时间信息的缺陷，然而最基本的 RNN Cell 内部是以全连接和 Tanh 激活函数实现的，而全连接的参数很多，而且对空间信息的提取效率不高，可以使用卷积代替全连接实现 RNN Cell，得到 Conv-LSTM，这样可以既可以学到时间信息(循环神经网络)和空间信息(卷积神经网络)[14]。

3.2.1 双向循环神经网络

某些数据的分类，比如语言翻译，它的信息可能是因果序列，也可能是非因果的。单向循环神经网络是一个因果结构的网络。一个句子前面的某个词可以成为句子后半部分的推论。许多应用中，要预测指定时间步t的值，可能需要整个序列或者t之后

的值。双向循环神经网络可以打破因果关系成为可能。双向循环神经网络结构如图 3-3 所示。

双向循环神经网络可以扩展到多向循环神经网络，双向一般指前后双向，而循环网络应用于图片识别，那么扩展到两个维度：横向和纵向，每个维度作一个双向循环网络。

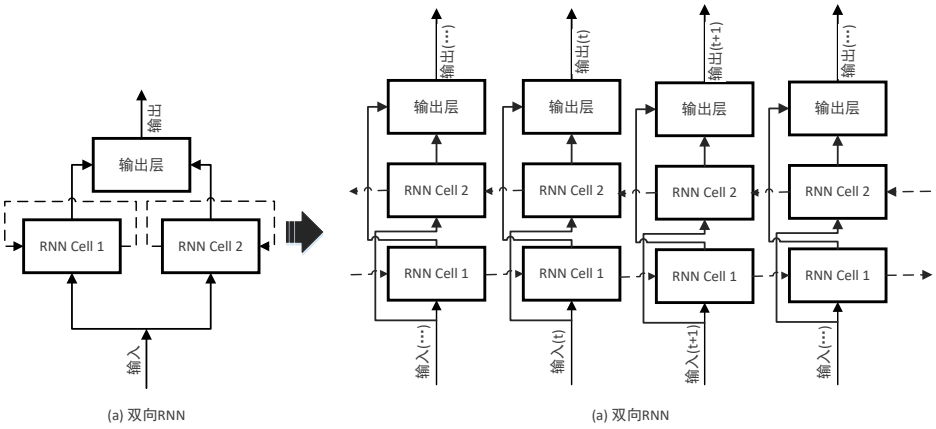

图 3-3 双向循环神经网络。它一层具有两个 RNN Cell，RNN Cell 1 从 $step = 0$ 作为第一个循环，向后递推；而 RNN Cell 2 从 $step = -1$ (-1表示最后一个$step$) 开始循环，向前递推。

3.2.2 深度循环神经网络

和深度卷积神经网络类似，通过堆叠 RNN Cell 层可以得到深度循环神经网络。然而如果没有特别的处理，层数加深同样会导致和深度卷积神经网络类似的问题，解决方法类似于卷积神经网络。在不同层之间加入一个叫 shortcut 的快捷连接可以构建出类似于 ResNet 残差卷积神经网络的深度循环神经网络，如图 3-4。

3.3 长短期记忆网络

循环神经网络只能让相邻的一个或者几个时间步联系起来，而对于长期的依赖信息无法学习到。而实际使用中，会遇到时间步跨度较长的依赖问题，由此诞生了具有门控功能的循环神经网络，他们在 Cell 中维护一个状态信息，这个状态信息横跨左右的 time-step。最经典的门控循环神经网络是长短期记忆网络(LSTM)[15]，基于长短期记忆网络还有很多变体，比如：Gated Recurrent Unit (GRU)等。

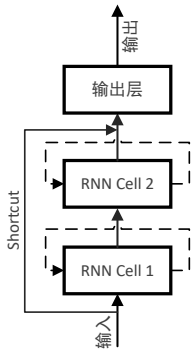

图 3-4 带 Shortcut 的深度循环神经网络。具有两个 RNN Cell 的深度循环神经网络，在输入层和输出层建立一个 Shortcut 快捷连接。

3.3.1 门控循环神经网络

门控循环神经网络利用门控结构控制状态的传递或更新。门控网络一般由多个门控结构组成。门控结构就像阀门一样控制信息的流动，多个门控结构构成门控循环网络的 Cell。典型的门控循环神经网络是长短期记忆网络。

3.3.2 长短期记忆网络

长短期记忆网络(LSTM)是常用的门控循环神经网络，具有忘记门、输入门、更新门和输出门，如图 3-5。在 LSTM 中，(1) 第一步就决定需要从 Cell 中丢弃什么信息，这个决定通过一个叫忘记门层($\sigma 1$)的结构完成，忘记门层会输出一个$0\sim1$之间的实数，0表示完全忘记，1表示完全保留原来的状态。(2) 接下来确定什么样的信息将被加入到新的 Cell 状态中，这其中包含了两个子结构，结构$tanh1$创建一个新的候选值，结构$\sigma 2$决定哪些候选值加入到新状态中和加入的比例。(3) 然后是更新 Cell 状态，由忘记门层和输入门层共同决定更新 Cell 状态。(4) 最后基于 Cell 状态经过输出门层作为我们需要的输出。

某些 LSTM 变体网络认为遗忘门控($\sigma 1$)和部分输入门控($\sigma 2$)可以耦合到一起形成一个门控结构，这样减少了可学习结构，优化了计算性能。

3.4 本章小结

循环神经网络可以克服卷积神经网络不能学习到时间信息的缺点，它可以和当前时间步之前的数据产生关系，它能学到序列的因果关系，并且具有卷积神经网络不

具有的动态性，可以识别不定长的序列。实际上，循环神经网络不仅仅能应用于学习时间信息，还能应用于学习时空信息，比如图片的识别中，可以把图片的宽看成是时间步。但是实际使用中，由于很多待识别序列并不是因果序列，可能后面的时间步对前面的时间步有一定的影响，由此设计出了双向循环神经网络，这样正向的循环网络可以学习到因果性，反向循环网络学习非因果关系。双向循环网络可以扩展到多向循环神经网络，对于图片识别，对宽和高都使用双向循环神经网络，这样就能得到一个四向的循环神经网络。

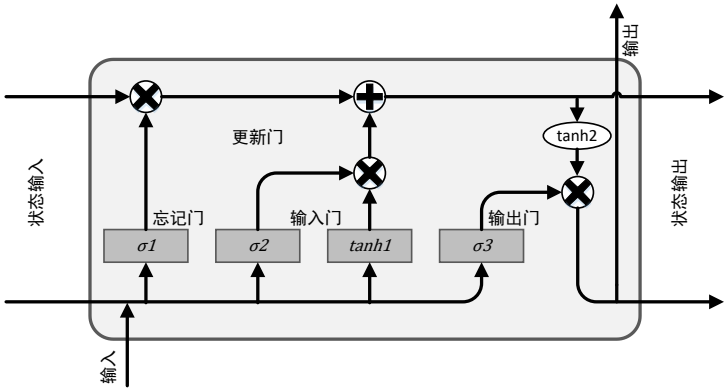

图 3-5 LSTM Cell。长短期记忆网络具有多个门控组件。忘记门：控制忘记多少以前的信息；输入门：控制当前时间步记住多少信息，记住什么样的信息；更新门：更新 Cell 状态信息；输出门：从 Cell 状态中输出学习到的特征。其中的长方形部分表示含可学习参数的层级，可通过反向传播算法学习。σ表示 Sigmoid 函数。

由于循环神经网络存在长期依赖的问题，由此设计了门控循环神经网络，例如 LSTM。LSTM 主要由忘记门控制 Cell 状态的保留与舍弃，可以解决长期依赖的问题。基于 LSTM 有很多小改动网络，它们在某些领域有着比 LSTM 更优的识别效率，然而在大多数问题上，它们的识别率和 LSTM 相同。

通过和卷积神经网络类似的堆叠方式，可以得到深度循环神经网络，它能有效的提高识别效率，然而也和卷积网络有相似的问题。这些问题的解决方案可以类比于卷积网络。

第 4 章 数据预处理

4.1 数据预处理的作用

数据预处理在大多数深度学习算法中起着重要的作用。如果不对数据进行一定的预处理，可能得到的模型参数的识别效果不理想甚至很糟糕。数据预处理的过程就是把原始数据的部分特征暴露出来的过程，处理后交给深度神经网络学习能够够快更好的得到优质的网络参数。实际使用中，数据预处理将经历归一化、白化等处理，然后再由其他算法加以处理。

在进行数据预处理前，首先要观察数据并获取一些显而易见的特征，可以指导预处理，也可以作为一些先验知识用于网络中。在图像数据的预处理中，一般先经过随机裁剪以增加数据量，然后归一化和白化处理等处理。

4.2 数据归一化

在数据预处理中，标准的第一步就是数据归一化处理，下一步就是其他的预处理方法。数据归一化就是让所有数据映射到某两个数值之间，比如映射到$0\sim1$或$-1\sim+1$之间。数据归一化有多种算法，常用的有：简单缩放、消除直流分量和特征标准化等算法。

4.2.1 简单缩放

简单缩放的目的就是对数据的每个维度上的值重新调节，使得变换后数据的值落在一个指定的区间内，通常是$[0,1]$或者$[-1,+1]$。这些维度上数据的分布可能相互独立，也可能有相关性，但是作简单缩放不影响数据的分布。变换后的区间，根据不同的数据可以作不同的选择。在做图像或视频的处理中，原始数据是像素，像素每个通道的值是在$[0,255]$区间内，像素值直接除以最大值 255，即可把数据缩放到$[0,1]$内。

数据缩放对后续的处理十分重要，大多数的后续处理算法中，都是假定数据被缩放到一个合理的区间内，比如：PCA-白化。对于卷积神经网络和循环神经网络，他们中大部分运算都是乘加运算，而计算机表达实数的能力有限，一般使用 32 位浮点型来表示数据和参数，把数据缩放到合理区间后，比如$[0,1]$，可以让参数由更大的

表示空间，如果数据值过大，那么参数可能需要一个非常小的数才能表示某个特征，而这个非常小的数可能在 32 位浮点型中可能溢出，当数据缩放到合理区间后，可以有效缓解溢出问题。

4.2.2 消除直流分量

消除直流分量又叫逐样本均值消除，它的作用就是去除样本的均值，对于那些我们对均值不感兴趣的数据，这样做尤为重要。如果需要处理的数据在每一个维度上都服从相同的分布，那么对每个样本的处理就是减去每个样本的数据统计平均值。

对于图像或视频，消除直流分量可以溢出图像或视频的平均亮度。在大多数视频分类问题中，我们并不关心视频的亮度信息，而只关心帧的轮廓和时间信息。这样我们移除样本均值后能够降低网络对视频的亮度敏感的敏感性，有助于提高网络的准确率。但是在处理彩色图像时，不同通道的像素值并不都服从相同的分布，我们在处理时可以对每个通道分开处理，分别减去它们自己的均值，这样可以避免由于柔和不同通道中不同的分布而导致网络性能降低的问题。

4.2.3 特征标准化

特征标准化是指让独立的让每个样本具有零均值和单位标准差，这个方法被广泛的应用在数据预处理中。实际使用中，先计算出单个样本数据的均值$mean$和标准差std，然后对样本中的每一个数据都做如下运算：

$$y = \frac{(x - mean)}{std} \tag{4-1}$$

和消除直流分量(见 4.2.2)类似，对于彩色图像，因为每个通道上的像素并不一定服从相同的分布，因此可以对每个通道单独作特征标准化，分别计算每个通道的均值和标准差。在视频处理中，通常需要保持整个视频具有相对固定的亮度，在做特征标准化中，对相同的通道的所有帧的像素作特征标准化。

4.3 光流

对于视频数据，相比于单个的图像数据还具有时间信息。对于视频的识别，不仅仅要学习到空间特征，更要在时间特征上进行学习。对于动作(活动)识别，可以通过光流算法，让时间特征压缩到帧图像上，可以减轻网络对时间特征学习的压力，也可以给网络多一个特征用于学习。

光流算法就是计算每两帧数据间的对应关系，从而得到图像的运动信息的方法。光流信息可以由场景中目标的运动产生，也可以由相机的移动或两者共同运动产生。运动信息可以直接加入帧数据中，也可以单独处理。如图 4-1，使用的 FarneBack 算法[16]计算得到光流信息后，在原数据对应的灰度图像上使用$(0, 255, 0)$的绿色线条绘制出光流特征。

a) 源图像　　b) 加入光流

图 4-1 稠密光流图像。a)图中是 RGB 三通道彩色图像，先转化成灰度图像后计算出光流信息，然后绘制到灰度图像上，得到 b)图。b)图中的绿色线条就是光流信息。

4.3.1 稀疏光流

稀疏光流是针对视频中的一组点，一般为顶点进行跟踪的算法，得到这一组点的运动信息。在计算稀疏光流信息前，需要使用其他算法找到这样的一组顶点，然后交给稀疏光流算法计算。

在 OpenCV 视频处理库中，常用 goodFeaturesToTrack 函数寻找一组便于跟踪的顶点，得到的点的坐标就可以在稀疏光流算法上。常用的稀疏光流算法是金字塔 Lucas-Kanade 光流方法，简称为 LK 算法[17]。OpenCV 中内实现了 LK 算法，可以直接使用计算得到光流信息。LK 算法中有三个基本假设：(1) 亮度恒定，图像场景的被跟踪的目标像素在帧间运动时保持不变，不能在运动过程中图像发生巨大的变化；(2) 时间连续或者缓慢运动，这个假设表明目标的运动相对于时间需要时缓慢的，因为视频是以确定的间隔时间采样然后保存下来的，如果运动太快，会导致不连续性；(3) 空间一致性，场景中的同一自图像的像素点具有相同的运动。也就是说，LK 算

法要求摄像头固定，目标物体自身没有相对运动且没有转动，并且运动缓慢或者摄像头采样速度足够高。

4.3.2 稠密光流

相对于稀疏光流的跟踪某些特定点不同，稠密光流会跟踪所有的像素点，跟踪的像素点在数量上相比于稀疏光流有很大的增加，甚至是不在一个数量级上，因此运算消耗会大很多。而且稠密光流还需要使用插值算法在容易跟踪的像素之间插值以解决对运动不明确的像素的跟踪，这进一步增大了性能消耗。稠密光流算法的条件和稀疏光流算法的假定条件类似，如果不满足假定条件，那么得到的光流信息就不清晰，可以通过其他的处理算法得到更清晰的光流信息。

FarneBack 算法是一种稠密光流算法，它能计算出每个像素点的在每两帧间的运动信息。它计算得到的数据是一个和原始图像相同大小的二通道数据，通道一表示像素在x方向上的运动距离，通道二表示像素在y方向上的运动距离。对于没有运动的像素点，得到的x和y均为零，如果要把光流信息绘制到原图像中，需要对光流信息按一定间隔抽样，只绘制部分光流，否则得到的图像将会被同一种颜色完全覆盖掉。

对于摄像头也在移动的视频，即背景也有运动，光流算法也会计算出背景的光流。经过观察，背景光流有一些和目标光流不同的特征：背景的面积相对于目标面积大得多，并且背景光流的运动方向几乎一致，而目标的光流和背景光流方向一般不同且一般不是在同一个方向。可以根据这个特点来去除不需要的背景光流信息。

4.4 视频预处理

视频的预处理中，相比图片要复杂很多，需要处理视频长度不匹配的问题，还有关于时间信息的保留，不能预处理后把时间信息给掩盖了。视频数据一般都是 RGB 三通道彩色视频，由于不同通道的数据可能不服从相同的分布(见 4.2.2 节)，因此在归一化处理中使用消除直流分量的处理有特别的要求。

深度神经网络是大数据分析的极为有效的工具，然而这需要大量的数据才能做到，在大数据面前，只要充分拟合训练集即可对测试集有很好的识别效率。但是往往我们需要在数据量不是很大的情况下做大数据分析，这就需要我们做好如何去有效的扩充数据量。针对于视频数据，有多种扩充数据的方法：视频长度的处理、随机翻转和裁剪等。

4.4.1 视频长度的调整

在机器学习网络中，网络的输入数据只能是一个或几个张量(tensor)，每个 tensor 内，各个维度上的数据长度必须相同，比如每帧的宽必须相等，同样对于不同视频的时间长度，也就是帧数必须都相同，因此对于不同时间长度的视频，需要处理成相同的时间长度。

对于长的视频，可以通过截断的方式截取出需要的长度，这是一个比较好的方法，通常也是这么处理，截断(滑窗)覆盖率设为25%。在对运动速度不敏感的视频识别中，可以通过压缩时间来使长的视频和短的视频的长度相匹配，短的视频也可以通过对时间进行插值的算法和长视频相匹配。然而这种方法对于运动速度敏感的视频识别中，就不再适用，比如：一个“走”的视频进行压缩，就会被压缩到和“跑”的动作相似，这不利于识别 “走”和“跑”。

对于短的视频，要复杂一些。短视频的处理有多种方法：(1) 最简单的是在视频后补零到指定的长度，如果补零的长度过长，有效部分的信息会被零数据给淹没；(2) 另一个方法是对短视频重复最后一帧以补足长度，然而这个方法对于动作识别却是一个不利的解决方案，因为补的数据是有意义的数据，如果补的数据长度过长，将会损失视频的动作信息，使整个视频呈现一种静止的状态；(3) 可以重复原视频来补长，这克服了方法(2)的不足；(4) 交替使用短的零数据和原视频补；(5) 交替使用短的随机干扰数据和原数据补，这种处理方式增加了样本空间，可以增强网络的抗干扰能力。对于(3，4，5)方法，由于补长后，可能长度已经超过了需要的时间长达，这时可以使用对长视频的处理方法来处理。

4.4.2 裁剪、翻转

在没有大量数据的情况下，比如需要在每个类别只有 130 个样本的情况下做多分类识别，如果不进行数据增加，直接进行训练很难训练出一个优良的网络。对于图片和视频，可以使用随机裁剪和翻转的方法增加数据量，增强模型抗干扰能力。对于图像，可以对每一张图片都进行随机裁剪和翻转，而视频则不能。

4.4.2.1 随机裁剪 随机裁剪就是对视频随机取出画面中的某个部分。和图像的裁剪不同，视频的裁剪需要对每一帧的相同位置作裁剪而不是每一帧都是随机裁剪。(1) 对每个视频，先随机生成要裁剪的位置，然后逐帧在这个位置作相同的裁剪，裁剪完

后把帧按顺序拼接成视频，得到随机裁剪的视频。在网络训练过程中，对每一个样本都可能使用很多遍，而每次读样本的时候，都需要重新做随机裁剪，这样就算是读取同样的样本，得到的训练数据也不同，这样就做到了增加样本的目的。因为网络的输入需要每个输入数据有相同的高和宽，因此每次随机裁剪的数据需要裁剪的画面有同样的大小。(2) 另一种随机裁剪方式是先随机选取一个位置的数据作保留，其他位置的数据置零。处理的时候，可以结合这两种处理方式，先做方法(1)的裁剪，然后再做方法(2)的裁剪。对于各个裁剪方法，都不能把视频裁剪得太小，导致得到的视频不能反映原视频的信息，这样的裁剪是失败的裁剪。

4.4.2.2 随机翻转 随机翻转比随机裁剪简单很多，就是对视频数据随机上下、左右的翻转变换，这样可以使得模型对视频翻转不敏感。同样需要注意，单个视频中的每一帧都需要做相同的翻转变换，而不能完全随机。

4.4.3 完整的视频预处理

一个完整的视频预处理需要经过很多步才能完成，比如光流、随机裁剪、归一化、PCA-白化等处理。由于光流处理非常的消耗时间，而且不管如何变换，每次产生的光流信息是相同的。可以做好光流处理后，保存下来，之后每次读取的数据都是含有光流信息的数据，这样可以加速网络的训练，不至于让训练进程因为数据输送不及时导致等待。

采用 UCF50-活动识别数据集[18]的子集作为网络的训练数据和测试数据，从中取出 10 类作为我的训练数据集，总的样本数为 1380，平均每个类的样本数为 138。类别分别为：Baseball Pitch, Basketball, Bench Press, Biking, Billiards, Breast Stroke, Clean And Jerk, Diving, Drumming, Fencing。

如图 4-2，视频数据预处理的流程：

(1) 把视频数据集分割成两个子数据集：训练集和测试集。训练集用于网络的训练，测试集用于最终评定网络的效率。训练集和测试集比例大约为10: 1。同一个人的同一个类别的运动必须被分割到同一个数据集中。

(2) 独立于训练网络编写光流处理程序。使用 FarneBack 计算稠密光流，从得到的每个像素的运动信息中抽样，比如按纵横都为 16 个像素点抽样。彩色图像可能干扰到光流信息的表达，并且灰度图像并不影响动作的表达，因此在原视频转化成灰度图像的基础上，使用绿色线条绘制光流信息到视频中。保存好得到的含

光流信息的视频，后缀名为 “.flow.avi”。

(3) 视频时间下采样。视频数据为了保证人眼观感的流畅性，时间上具有一定的冗余，可以对视频在时间上进行下采样，这样既可以保留时间信息，又能降低网络的训练成本。经过实验发现对于大多数视频，当帧率大于 10 时就可以保留动作信息。

(4) 调整视频长度。在网络训练中，我们需要使网络可以识别任意长度的视频，因此不能把所有视频都调整为一个长度，可以调整为多个长度，比如：60、70、80 帧等或从60~80帧中随机。

(5) 随机裁剪视频。读取含光流的视频，随机在画面中裁剪出宽高比和原视频相同的画面，并且裁剪的宽高不能低于原宽高的$^3/_4$，否者容易丢失视频的目标。然后调整视频分辨率到指定的分辨率，比如96×128。

(6) 随机翻转视频。以上下、左右均为0.5的翻转概率，对裁剪后的视频随机翻转。且视频中的所有帧都做相同的翻转。

(7) 特征标准化。对每个视频样本，先分别按通道维抽取视频，然后作特征标准化，最后按通道维拼接起来。

以上是完整的视频数据的预处理过程。对于(1)和(2)，需要在训练网络前完成并保存下来，(3)到(7)需要和训练进程同步处理，由于视频数据量巨大，如果一次性把所有的视频数据都读入到内存中将会增加内存的压力，可以按需读取，当网络需要数据时，就读取数据，预处理后送入网络中。为了进一步减少训练网络进程的等待时间，可以使用队列，在数据预处理进程中预先处理好一定数量的样本数据缓存到内存中，然后在网络需要时，直接从内存中取出处理好的数据。

4.5 本章小结

数据预处理是数据分析中的常用手段。在数据进入网络前，做一定的预处理可以提高网络的抗噪能力、识别效率和降低网络对某些变换的敏感性。数据归一化处理可以使网络发挥最佳的效果；光流处理是针对动作视频数据的预处理，可以提高网络对时间特征的提取能力；随机翻转可以让网络对视频或图像的方向不敏感。

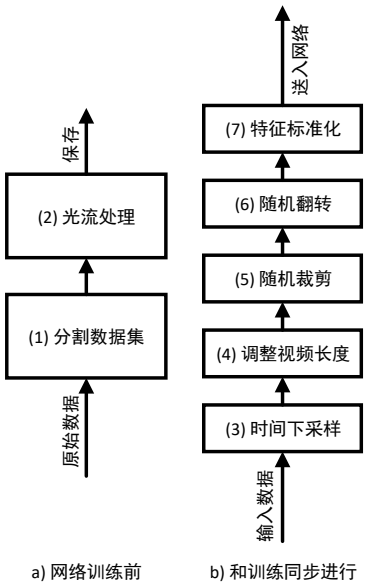

图 4-2 视频预处理。视频预处理有七步，前两步，a)图，独立于训练进程，需要在训练进程开始前处理完；后五步，b)图，需要和训练进程同时进行。

同时预处理还可以提高网络的训练效率。视频预处理中，做时间下采样可以降低数据量，并且还不会损失太多的时间信息。

大多数情况下，由于样本数据量的不足，导致无法实现大数据分析。数据预处理可以在一定程度上解决这个问题。视频预处理中，调整视频长度和随机裁剪视频可以在现有的没有大量样本的基础上认为增加很多样本数据。调整视频长度中，对于过长的视频，随机位置截取；对于短视频，通过重复随机数据和原视频的方式补长视频，当超过指定长度后，再随机截取。随机裁剪视频中，由于每次裁剪的位置随机，得到的样本数据也不同。这样就能在原有小样本的基础上增加大量的数据。

第 5 章 视频活动识别

每个动作的视频的每一帧都具有一定的空间相似性，比如游泳运动中，每一帧中都会有水池，棒球运动中，每帧都有球棒和棒球。基于这个先验知识，可以使用深度卷积神经网络作为前端网络来提取视频帧的空间特征。之后送入深度 LSTM 网络中提取时间特征，这里的 LSTM Cell 使用卷积 LSTM Cell，这样可以进一步提取时空特征。数据流出深度 LSTM 网络时，每个 time-step 都具有一个输出，这时有两个选择：(1) 综合所有的输出得到一个输出，(2) 只选用最后一个 time-step 的输出作为最终的输出。在序列到序列的问题中，选用方法(1)，而在分类问题中，可选用方法(2)，也可以都所有或部分输出做加权求和后作为最终的输出，这儿的权重可以指定，可以作为参数学习。

5.1 子网络

在大网络中，可分为多个子网络(模块)做分析，本文总共分为三个主模块和一个主网络。卷积子网络作为有效的空间特征提取器用作网络前端；Conv-LSTM 子网络作为强大的时间提取器用作网络的核心部分；在输出部分，使用两个全连接层和一个 Dropout 层；最后如果要得到样本在某个类别的预测概率，可以再接一个 Softmax 层。

5.1.1 卷积子网络

在这一层中，使用卷积层作为该层主体，这并不是固定的，还可以选用其他的提取空间特征的网络替换该层作为网络的前端。在这一层中，使用多层卷积和一些辅助层构成该子网络。该子网络中，视频输入是一个五维张量(Tensor，可以看作是多维数组)，张量形状(Shape)为[batch, time-steps, height, width, channels]。第一维是 mini-batch 的大小，即每个批次的视频数量；第二维是视频的长度(time-step)，即每个视频的帧数；第三、四、五维是视频的高、宽和通道数，通道数也可以称为特征图个数。

在 Tensorflow 中，二维卷积的输入和卷积核(滤波器)需要是四维 Tensor，在填充方式上使用“SAME”参数(参见 2.1.4 方式 2)的情况下，输出的 Shape 和输入相同。由于这个特性，导致卷积输入和视频输入不匹配，需要一个 Reshape 层调整，把视频输入的第一维和第二维融合到一起。

如图 5-1，在 Conv-1 层中，输入通道数为 3，输出通道数为 32，卷积核大小为3×3。通常使用3×3作为卷积核大小，比使用5×5卷积核的效率要高。在 Conv-2 层中，输入为 Conv-1 层的输出，因此输入通道数为 32，和 Conv-2 层一样，使用3×3的卷积核，输出通道数提升到 64。这两层中，均有2×2的池化，因此输出的特征图高和宽都有减小为原来的$1/2$。

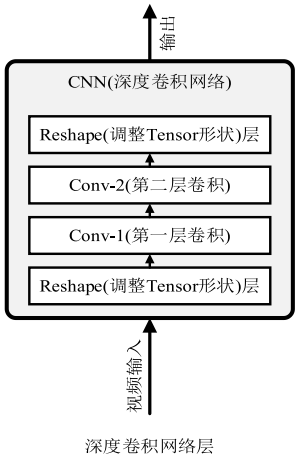

图 5-1 含两层卷积的深度卷积子网络。在这一子网络中，使用两个 Reshape 层作为子网络的输入输出部分，用于适配输入输出的 Tensor Shape；通常多层卷积网络比单层卷积网络有更好的识别效率。

最后，使用一个 Reshape 层作为该子网络的输出层。该层和第一层做了相反的工作：把第一维分开成两维，作为之后的网络的 batch 和 time-step。Reshape 层的运算代价很低，不用担心对网络运算效率有多大的影响。

这一子网络中，使用两个卷积层作为核心，提取样本每个时间步的空间信息。输入为一个以视频为样本的 mini-batch，输出和输入的 Shape 类似。输出的高和宽有所降低，经过两层带池化的卷积层后，特征图的高和宽均降低为网络输入的$1/4$。输出通道数的提高增加提取的特征数量，使网络具有更高的识别率。

5.1.2 Conv-LSTM 子网络

如图 5-2，在这一层中，使用 Conv-LSTM 作为核心，因为它不仅仅可以学习到时间特征，还能进一步学习到空间特征，并且还能大量减少参数个数，节省计算量和内存空间。在这一层中，由两个 Conv-LSTM 层构成，并且预留出了 Cell 状态接口。

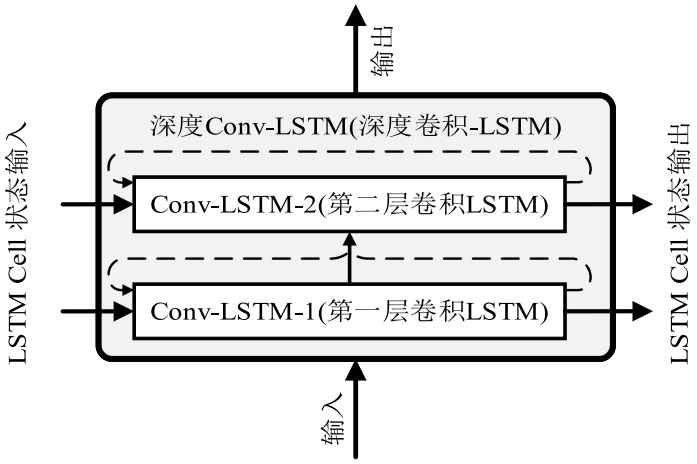

图 5-2 Conv-LSTM 子网络。使用两个卷积-LSTM 层作为主体，对 LSTM 的首尾预留 Cell 状态的输入输出接口备用。

该子网络中，输入输出均为列表(list)，list 的索引表示 time-step，list 的元素是四维张量，类似于卷积层的输入，shape 为[batch, height, width, channels]。该张量表示 batch 内所有样本的当前 time-step 的帧，通道数为当前帧的特征图个数。

核心部分是两层 Conv-LSTM，LSTM 层中如果$time_steps = 1$称为非展开图；小于视频的长度称为部分展开图；等于视频长度称为完全展开图。非展开图和部分展开图的训练方法是在训练网络外部使用循环逐步 feed 数据，每次都更新网络参数，学习速率逐渐增大，下一个训练步来到时，学习速率回到原始值。

本子网络输出部分中，把第二层 LSTM 的输出作为总的输出，是一个 list，元素为四维 tensor。在这一网络中，由于使用了 Conv-LSTM 作为核心层，使得该网络具有提取时空特征的能力，并且该网络具有多种形式：非展开，半展开和完全展开。在 TensorFlow 中，由于只支持静态图，因此非展开形式有最小的内存消耗，完全展开内存消耗最大，半展开介于两者之间。非展开和半展开形式需要在网络外部控制 LSTM 的循环，这两种形式具有一定动态性。

5.1.3 网络主体部分

如图 5-3，网络主体部分主要是用一个卷积神经子网络和一个 Conv-LSTM 子网络。因为卷积神经网络的输出和循环神经网络的输入不匹配：卷积神经网络的输出是一个五维 Tensor，而循环神经网络的输入需要一个元素为四维 Tensor 的 list。可以使

用一个 Split 层按时间轴分裂卷积神经网络的输出，转换为元素为四维 Tensor 的 list，这样就能实现数据匹配。如果主体网络中的 Conv-LSTM 部分使用的是非展开形式，由于网络总输入是四维 Tensor，因此可以去掉 Split 层，并且卷积网络中也可以不使用 Reshape 层。

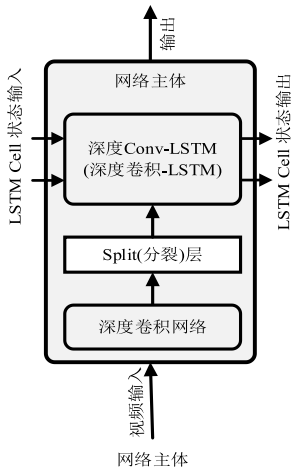

图 5-3 网络主体部分。使用卷积子网络作为前端，LSTM 子网络作为后端，中间使用一个 Split 层作为前后端接口层，同时预留 Cell 状态输入输出接口备用。

当其中的 Conv-LSTM 使用非展开或半展开形式时，运行图的输入是让一个批次(batch)的视频按时间轴分裂为更小的批次(micro-batch)输入到网络中，这时网络外部将会使用到 Cell 状态输入输出接口：循环开始时初始化 Cell 状态为全 0，每次运行完一次图后，读取 Cell 输出状态，用于下一次运行图的 Cell 状态初始化，当下一个 batch 来到时，Cell 状态将再次置零。

5.1.4 输出子网络

如图 5-4，输出网络由一个 Reshape 和两个 Local 层构成，使用 Dropout 的训练策略，当需要得到每个样本在各个类别的预测概率时，可在输出网络后加一个 Softmax 层。

Local 层使用全连接实现，全连接的输入二维 Tensor，Shape 为[batch, values]，batch 为样本数，values 为每个样本的数据量。而无论是卷积神经网络的输出还是 Conv-LSTM 的输出都是四维 Tensor，因此在全连接前需要把四维 Tensor Reshape 为二维 Tensor，进入 Local-1 层做全连接，这时数据将会损失一部分空间信息。接下来数据进入到 Dropout 中，它通过随机丢弃部分(置零，丢弃比率可以通过$keep_prob$参

数控制)数据，它具有正则化作用，可以防止网络过拟合。

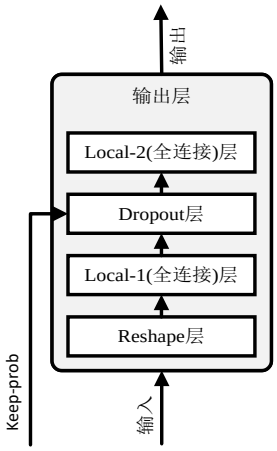

图 5-4 输出子网络。使用两个 Local 层和一个 Dropout 训练策略，Dropout 用于对网络正则化，防止网络过拟合。

接下来的 Local-2 层把数据映射到类别。如类别数为 10，那么经过 Local-2 和 Softmax 层后，每个样本由 10 个数表示，他们分别表示该样本在相应类别上的预测概率。取概率最大值对应的类别作为网络预测的样本类别。

5.2 总网络

总网络的基本思路是使用主体网络加输出层网络构成，损失函数采用“Softmax-交叉熵”加 L2 正则化作为损失函数，选用 ADAM 优化算法，使用 TensorFlow 的自动反馈更新参数。对应于完全展开、半展开和非展开具有不同的网络结构。完全展开网络有两种结构：多输出网络和单输出网络，半展开和非展开网络可以合并为一个网络：部分展开网络。

5.2.1 多输出网络

多输出网络基于完全展开的 Conv-LSTM，然后 Conv-LSTM 的输出中取出需要的 time-step 的值，对于不需要的值做舍弃处理。对每一个取出的输出，都需要经过输出层，训练时还需要经过“Softmax-交叉熵”和反馈网络，以优化参数。

对于完全展开的 LSTM 网络，如图 5-5，如果只取中间某个 time-step 的输出，使用 TensorFlow 的情况下，之后的 time-step 将不会运行，也就不会消耗计算机资源。

比如在一个总 time-steps=80 的 LSTM 网络中，如果只取 time-step=60 的输出，那么 time-step=61 及之后的 time-step 对应的循环将不会运行。这样可以在不额外消耗计算机资源的情况下实现具有一定动态性的 LSTM 网络，其他的循环神经网络也可以同样的处理。

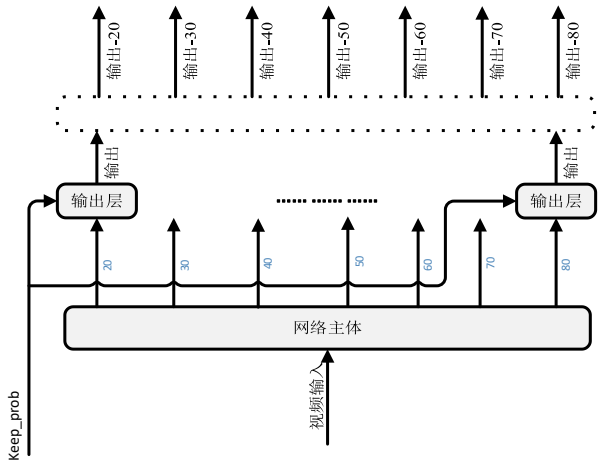

图 5-5 多输出网络。主体网络中，使用最高 80 次循环的 LSTM，取出 LSTM 中的指定 time-step 的输出，实现半动态的 LSTM。

然而这并不是就完美无缺，当要收集某些层的输出时，弊端就就会显现出来了。因为具有多个输出，当需要收集输出网络层中的 Local-1 的输出时，由于具有多个 Local-1 层，这时的处理就会非常棘手，稍不注意程序就会崩溃或者消耗额外的计算机资源。

5.2.2 单输出网络

为了解决多输出网络的弊端，单输出网络应运而生，如图 5-6。它使用一个 Switch 切换层作为 Conv-LSTM 和输出层的接口，它可以根据输入动态的指定某一个 time-step 的输出输入到下一层，阻止 Conv-LSTM 的其他输出进入下一层。它的网络结构更加简洁，更容易理解。然而，由于 Switch 层时基于 Conv-LSTM 的所有输出均算出来的情况下的开关，因此对于多输出中不运算的循环也进行了无用的运算，消耗了额外的计算资源，并且整个网络的输入的时间长度需要等于最大长度，不足的使用 0 值补齐(不影响输出)。

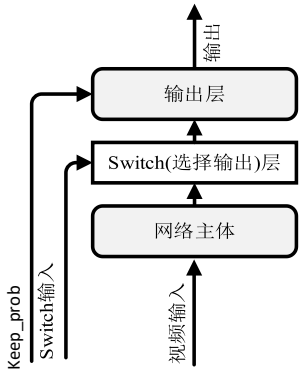

图 5-6 单输出网络。主体网络中，使用最高 80 次循环的 LSTM，在主体网络和输出子网络之间加一个 Switch 开关(筛选)层，实现半动态 LSTM。

该网络的运算结果和多输出网络的运算结果完全相同。同于由于补长的部分是由零填充的，因为计算机的优化，大量含零的运算(比如：乘和除，网络中主要为乘加运算)并不会成倍的增加运算量，然而还是有很多的计算性能被浪费。

5.2.3 非展开和半展开网络

为了弥补基于完全展开的多输出网络和单输出网络的不足，非展开和半展开网络得以设计出来，如图 5-7。

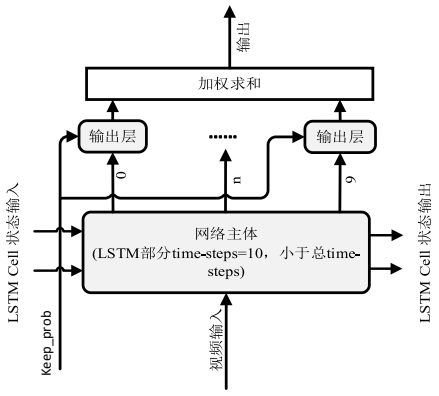

图 5-7 半展开网络。主体网络中，使用最高 10 次循环的 LSTM，预留出 Cell 状态输入输出接口用于实现完全动态 LSTM，最后再输出算做加权求和综合所有 time-step 的输出。当网络中的 time-step=1 时，网络退化为非展开网络。

非展开网络完全不在网络中把 Conv-LSTM 展开，而是在网络的外部通过循环的形式展开计算，每次计算一个 time-step，针对于需要任何完全动态的 Conv-LSTM 网

络，它可以在只额外消耗非常少计算资源的情况下，实现完全的动态网络，并且占用的内存资源也将大大减小。在单输出网络中，参数、网络层数和每批次样本数都需要非常小才能保证不出现内存溢出，而在非展开的网络中，则几乎不用考虑内存溢出的问题。由循环神经网络的特性可以指导，一个完整的循环中，time-step +1 的输出比 time-step 的输出具有更多信息，因此在该网络训练时，随着 time-step 的增加，学习速率需要有相应的提升。然而该网络的资源利用率并不高，可能不能发挥出显卡大规模并行计算的能力。

半展开网络是基于全展开网络和非展开网络之间的网络，对于不是每个 time-step 都需要输出的问题非常适合。它既解决了动态性的问题，又能充分利用资源，加速网络的训练。在网络的最后输出部分，使用一个加权求和层，综合所有 time-step 的输出，以让最终输出更准确。如果某个 batch 的视频长度不能被半展开网络内的 Conv-LSTM 的长度整除，那么需要对数据进行随机截取或其他特殊处理(比如使用只使用有效部分的输出做加权求和)。

5.3 实验结果与分析

如图 5-8，第一层 CNN 内包含四层卷积，两个池化。输出通道数为 32,。第二层 LSTM 包含两层 Conv-LSTM，输出通道数为 64。第三层 out_layers 包含两层全连接和一个 Dropout，输出通道数为类别数 10，映射到分类类别上。

虽然样本很少，然而视频的数据本身很大，而且数据预处理后导致样本空间巨大，这样网络的训练时间将会很长，可以减小数据预处理中的随机化程度，以减小训练样本空间。因此只测试了半展开网络的训练和测试。设置超参数为：

(1) 类别数为 10，(2) 输入数据的 Tensor Shape=[None, 96, 128, 3]；(3) batch=8；(4) 视频长度随机60~80帧，随机梯度为 2；(5) 卷积网络为四层，第一层和第三层有2×2池化，卷积核均为3×3，输出通道数分别为 16、16、32、32；(6) Conv-LSTM 为两层，输出通道数分别为 32、64，其中的卷积核均为3×3，不带池化；(7) Local-1 层使用连接数为 256 的全连接；(8) 对于半展开网络，网络内的 LSTM 长度为 10；(9) 训练时，需要使样本空间的所有样本都要至少输入网络 10 次，计算得到训练步为 157000 步；(10) 训练时，每过 50 训练步记录一次当前 batch 准确率和损失，每过 500 训练步保存一次模型并计算测试集的准确率和损失。

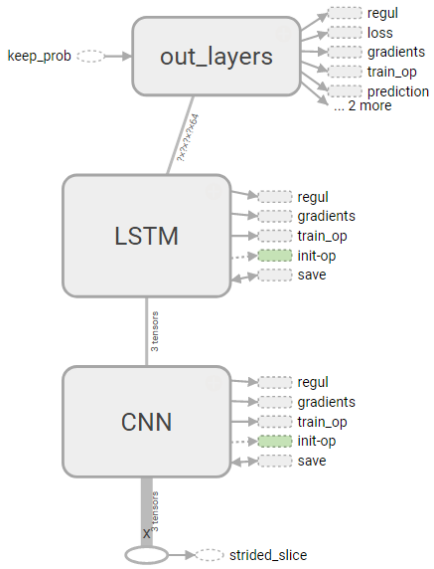

图 5-8 前馈网络。分为三层结构：CNN，LSTM，out_lsyers。out_layers 层输出一个对应于类别的映射，取出最大值对应的类别，即可得到预测类别。

硬件配置上，使用一块 NVIDIA GTX 1080TI(11GB 显存)GPU 作为加速硬件，用于网络的训练和预测。软件配置上，使用 Tensorflow 的 Python 3.5 接口搭建网络，并训练。数据集使用 UCF50-活动识别数据集的 10 个类别的子集。

学习速率指数衰减公式如下：

$$lr = init\_lr \times decay\_rate^{\frac{global\_step}{decay\_steps}} \tag{5-1}$$

其中，global_step 为当前训练步的计数器，decay_steps 一般为总训练步。这个函数可以自动的调整每个训练步的学习速率，使学习速率跟随者训练步的增长不断下降。

如图 5-9，图中含有的实线为经过平滑处理后的曲线，而虚线为原始值得折线。可以看到，虚线上会有一些尖峰毛刺出现，原因是在训练过程中，数据预处理中的随机处理造成的，当随机预处理出一批训练样本是网络以前没有训练过的样本时，就出现了尖峰毛刺，当一批数据中出现了陌生数据越多，毛刺越高。这并不影响模型的训练，恰恰相反，这能够让模型记住这个未出现过的样本。从图 5-9 也可以看出，训练越往后，毛刺出现的越低且越少，到40000步之后，几乎没有出现毛刺，说明从40000步开始，训练样本几乎都是训练过的样本。

如图 5-10，和图 5-9 类似，实线是经过平滑处理的曲线，虚线是原始值得曲线。同样出现了一些毛刺，原因和损失函数值相同。

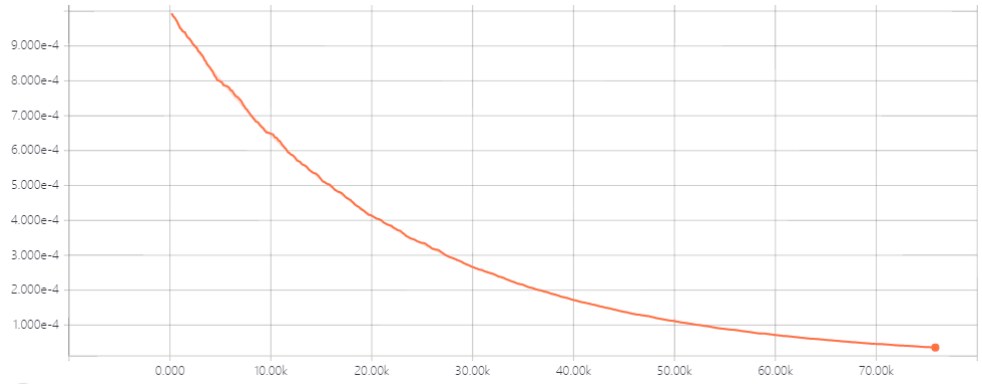

图 5-9 学习速率下降曲线。以学习速率为纵轴，训练步为横轴。使用指数衰减算法自动下降学习速率，初始学习速率为$1e-3$，衰减率为$1e-3$。

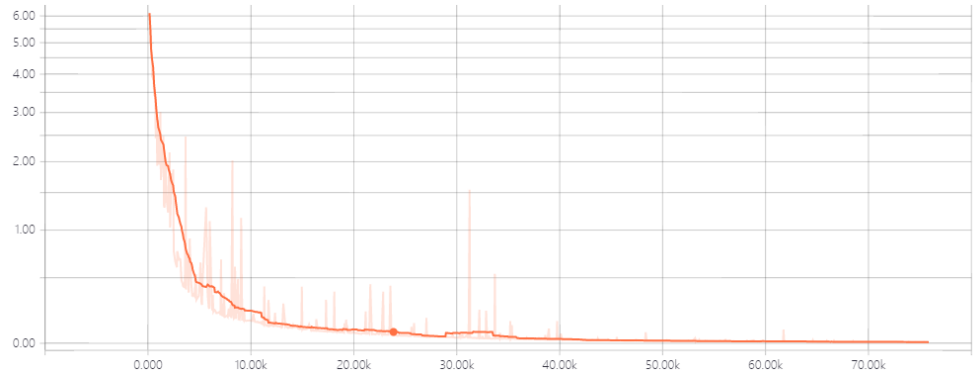

图 5-10 训练集损失函数值曲线。以训练 batch 的损失函数值为纵轴，训练步为横轴。可以看出损失函数值呈现一种持续下降的趋势，并且一开始下降速率大，而后下降速率越来越小，损失值趋近于零。

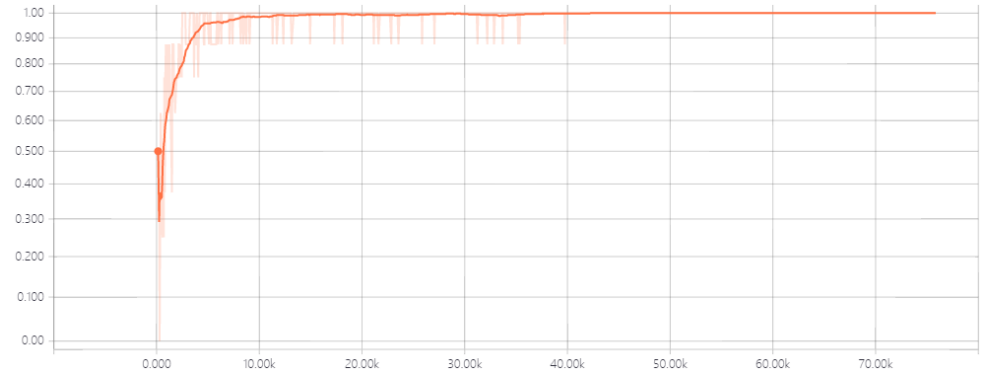

图 5-11 训练集准确率曲线。以训练 batch 的准确率为纵轴，训练步为横轴。可以看到准确率出现一个上升的趋势，最后稳定在1.0上。

准确率和损失函数值呈现一定的反比关系，最终准确率趋近于1.0，损失函数值趋近于0.0。

由于程序内存泄露问题，经过 12 小时 52 分钟，在 76000 步时，程序由于内存不足崩溃，由于做了等间隔训练步的模型保存，此时可以得到 76000 步时的模型参数。此时测试集准确率基本稳定在$82\% \pm 2\%$，训练集准确率为 1.0，训练集损失为 0.0。期间，测试集准确率最高达到 89%，最终稳定在 82%。

5.4 本章小结

网络结构往往决定了识别率的上限值，优秀的网络结构能够通过训练不断地优化，实现更高的准确率。本章通过把网络分块的形式介绍了一个可用于视频活动识别的网络：前端为卷积神经网络，核心为 Conv-LSTM 网络，末端为全连接的输出网络。不同的子网络之间，可能需要一些接口层，以匹配不同网络的输入输出。之后还介绍了视频识别网络的不同结构，一步一步实现了减少额外计算量、减小内存使用量和优化充分利用显卡大规模并行计算的能力。完全展开的网络对于超长视频的训练将会非常容易内存溢出，而非展开网络可以把内存使用量降到最低，半展开网络实现了在内存不会溢出的情况下充分利用计算资源，达到计算资源利用的最大化，加快网络的训练速度。

结论

本文通过分析视频活动识别的特点，作为强的先验，结合卷积神经网络和循环神经网络各自的特点，完成了基于视频的活动识别神经网络。深度卷积神经网络作为前端网络，深度 Conv-LSTM 网络作为核心，加上常规输出网络构成的视频活动识别的网络，在 UCF50-活动识别数据集的子集上取得了预定的识别率目标。

基于视频的样本具有一些明显的特点，可以作为一个无限强的先验知识用于网络和数据预处理中，比如动作具有一定的连贯性，并且目标和背景的区分度很高，这可以使用对样本生成稠密光流特征送入目标网络中，提供给网络更多的数据特征。针对于训练样本过少的问题，使用随机裁剪和翻转的方式人为的增加数据量。

本文介绍了循环神经网络展开的特征，实现了一个半展开的网络，克服了 TensorFlow 深度学习计算库只支持静态图而使得循环神经网络完全展开时非常占用内存资源的问题。该网络能够在充分利用计算资源的情况下有效的降低内存使用量。并且不影响最终的预测准确率，取得了成功。

参考文献

[1] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.

[2] 王松林. 基于 Kinect 的手势识别与机器人控制技术研究[D]. 北京交通大学, 2014.

[3] 覃耀辉. 视频中的人体动作行为识别研究[D]. 电子科技大学, 2011.

[4] 江焯林. 基于计算机视觉的人体动作检测和识别方法研究[D]. 华南理工大学, 2010.

[5] Bouvrie J. Notes on convolutional neural networks[J]. 2006.

[6] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.

[7] Glorot X, Bordes A, Bengio Y. Deep Sparse Rectifier Neural Networks[C]//Aistats. 2011, 15(106): 275.

[8] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 770-778.

[9] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of Machine Learning Research, 2011, 12(Jul): 2121-2159.

[10] Kingma D, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[11] Ji S, Xu W, Yang M, et al. 3D convolutional neural networks for human action recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(1): 221-231.

[12] Abadi M, Barham P, Chen J, et al. TensorFlow: A system for large-scale machine learning[C]//Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA. 2016.

[13] Elman J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.

[14] Donahue J, Anne Hendricks L, Guadarrama S, et al. Long-term recurrent convolutional networks for visual recognition and description[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 2625-2634.

[15] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[16] Farnebäck G. Two-frame motion estimation based on polynomial expansion[J]. Image analysis, 2003: 363-370.

[17] Lucas B D, Kanade T. An iterative image registration technique with an application to stereo vision[J]. 1981.

[18] Reddy K K, Shah M. Recognizing 50 human action categories of web videos[J]. Machine Vision and Applications, 2013, 24(5): 971-981.