学　　号：2016215473　　密　级：

合肥工業大學

Hefei University of Technology

本科毕业设计（论文）

UNDERGRADUATE THESIS

类　　型：	设计
题　　目：	城市环境中车道线检测方法设计
专业名称：	交通运输
入校年份：	2016 级
学生姓名：	周正
指导教师：	陈佳佳　　讲师
学院名称：	汽车与交通工程学院
完成时间：	2020 年 5 月 25 日

合　肥　工　业　大　学

本科毕业设计（论文）

城市环境中车道线检测方法设计

学生姓名： 周正

学生学号： 2016215473

指导教师： 陈佳佳　讲师

专业名称： 交通运输

学院名称： 汽车与交通工程学院

2020 年 5 月

A Dissertation Submitted for the Degree of Bachelor

Research on Lane Detection in Urban Environment

By

Zhou Zheng

Hefei University of Technology

Hefei, Anhui, P.R.China

May, 2020

毕业设计（论文）独创性声明

本人郑重声明：所呈交的毕业设计（论文）是本人在指导教师指导下进行独立研究工作所取得的成果。据我所知，除了文中特别加以标注和致谢的内容外，设计（论文）中不包含其他人已经发表或撰写过的研究成果，也不包含为获得___合肥工业大学___或其他教育机构的学位或证书而使用过的材料。对本文成果做出贡献的个人和集体，本人已在设计（论文）中作了明确的说明，并表示谢意。

毕业设计（论文）中表达的观点纯属作者本人观点，与合肥工业大学无关。

毕业设计（论文）作者签名：　　签名日期：　　年　　月　　日

毕业设计（论文）版权使用授权书

本学位论文作者完全了解__合肥工业大学__有关保留、使用毕业设计（论文）的规定，即：除保密期内的涉密设计（论文）外，学校有权保存并向国家有关部门或机构送交设计（论文）的复印件和电子光盘，允许设计（论文）被查阅或借阅。本人授权__合肥工业大学__可以将本毕业设计（论文）的全部或部分内容编入有关数据库，允许采用影印、缩印或扫描等复制手段保存、汇编毕业设计（论文）。

（保密的毕业设计（论文）在解密后适用本授权书）

学位论文作者签名：　　指导教师签名：

签名日期：　　年　　月　　日　　签名日期：　　年　　月　　日

摘　要

车道线检测是自动驾驶系统和高级驾驶辅助系统中重要的构成技术。当前城市规模不断扩大，机动车保有量日益增长，城市交通环境复杂多变，让车辆在城市环境中快速、准确地检测车道线对发展自动驾驶、提高交通环境的安全性具有重要意义。

传统的车道线检测方法依靠复杂的图像处理算法与繁杂的特征设计，受路面阴影、遮挡和不良天气的影响较大，准确率低、适应性差，无法应对复杂的城市环境。随着深度学习技术近些年的迅猛发展，卷积神经网络在图像处理方面展现出了强大的能力，能够较好的完成车道线检测任务，取得了比传统方法更加优秀的效果。但是目前绝大多数车道线检测算法均为处理单帧图片，而车辆的行驶是一个在时间上连续的过程，车道线的结构也是连续的，这些算法没有能够充分利用多帧图片的时间信息。

为了能更好地在复杂城市环境下检测车道线，本文使用了基于深度学习技术的时序神经网络，以 U-Net 网络为基础，构建了能够融合时间信息的网络模型，充分利用了时间信息进行车道线的检测。为了对比网络的性能，本文也实现了使用长短期记忆来联系时间信息的网络和只处理单帧图片的网络。

实验结果表明，本文提出的网络模型相较于处理单帧图片的网络模型具有更高的准确性，且相较于使用长短期记忆的方法，运行速度更快。

关键词：车道线检测；深度学习；时序神经网络；语义分割

ABSTRACT

Lane detection is important component technology in the automated driving system and Advanced Driving Assistance System. With the accelerating urbanization process, the motor vehicle is increasing every day, the vehicle can detect the lanes rapidly and accurately in the complicated and changeable urban traffic environment, which means a lot to improve its safety.

Relying on the complex image processing algorithm, and the complex image processing workflow, the traditional lane detection does not perform well facing the intricate urban environment, especially affected by road shadow, shielding and bad weather. With the rapid development of deep learning techniques in recent years, the Convolutional Neural Network (CNN) has shown a strong capability of image processing and better completed the lane detection. However, the current lane detection algorithm is mostly designed for processing the single frame image, so that it has no continuous information making use of multiple images. As the vehicle driving is a series of continuous processes in time.

In order to better detect lane lines in complex urban environments, this paper uses U-Net and temporal convolutional network based on deep learning technology, that can integrate time information is constructed, and the time information is fully used to carry out Lane line detection. In order to compare the performance of the network, this paper also implements a network that uses long and short-term memory to contact time information and a network that only processes single-frame pictures

It is indicated from the results that the network model proposed by this paper has higher accuracy than the one processing the single frame image, and significantly speeds up the detection compared to that applying the long short-term memory method.

KEYWORDS: lane detection; deep learning; temporal convolutional network；semantic segmentation

目　录

插图清单

表格清单

1 绪论

1.1 背景与意义

近年来，自动驾驶一词不断出现在人们的视野中，相关的技术在学术界不断推陈出新，在工业界更是已经加以使用。而在自动驾驶中，最困难的部分之一就是交通场景的感知。让车辆认清自己在道路上的位置是一个复杂、困难且必须要解决的问题。目前，车辆主要通过计算机视觉的方法判断车道位置，装在车辆上的摄像机捕捉行车时道路的图像，由计算设备对图像进行分析，找到车道线或车道的位置，将位置信息传递给车辆上其他设备来实现进一步的功能。

城市道路环境复杂，通过图片分析车道位置时很容易受到光照、遮挡、阴影和不良天气等不利因素的干扰。城市环境中道路交通情况复杂，车道线标志磨损程度大、清晰度低，而随着城市规模的不断扩大，机动车保有量的不断增长，社会各界越来越需要一种在城市环境中表现良好的车道检测方法。自从 AlexNet 在 ImageNet 图像分类大赛中获得冠军以来，基于深度学习算法的图像处理方法展现出了强大的实力，基于深度学习的算法极大地提高了图像分类、语义分割等任务的准确率。在图像处理任务中，应用最多的就是卷积神经网络，这种算法近年来也得到了极大的发展。在车道检测任务中，不断有新的网络结构被提出，不断有新的数据集被公开，也进行了大范围的应用尝试，但是在面对复杂的城市环境时，学术界和工业界依然希望进一步提高车道线检测的准确率与速度。

因此，本文以深度学习技术为基础，在此之上研究新的网络结构，提高车道检测的速度和准确性，对自动驾驶的发展和道路安全的提高有一些帮助。

1.2 国内外技术研究现状

1.2.1 传统基于图像处理的车道检测算法

传统基于图像处理的车道检测算法大致可以分为两类，即基于特征的方法和基于模型的方法。

基于特征的方法，依赖于车道的独特特征来实现检测，比如车道的颜色、结构和边缘。D.J.Kang[1]等人利用 Sobel 算子检测车道边缘特征，将道路图像沿垂直方向分成多个子区域后进一步检测。为了消除图片中由于摄像机本身的特性带来的图像畸变，可以使用透视变换的方法。Collado 等人发明了一种生成鸟瞰图的

方法[2]，该方法基于空间车道特征与霍夫变换算法。有些研究中也使用了车道的颜色特征，在[3]中，使用无监督自适应分类器来识别车道，它将彩色图像转化为HSV格式来增强对比度，然后使用基于亮度值的阈值方法对二值特征图像进行处理。

基于模型的方法通常将车道用模型来描述，如线性模型、抛物线模型、样条模型等。在此前的研究中，样条模型因为其灵活的特性而大受欢迎，可以描述任何车道曲线的形状。Wang等人使用不同的样条模型[4]、[5]进行车道拟合。在[6]中，使用基于近场区域的霍夫变换和远场区域的河流法（river-flow）进行车道边界的检测，最后用样条模型对车道进行建模，用卡尔曼滤波进行跟踪。

1.2.2 基于机器学习的车道检测算法

近年来，随着机器学习的迅猛发展，许多人将目光放在了用机器学习和深度学习来解决车道线检测的问题上。尤其是深度学习，已经变成了近年来最热门的研究领域之一。Li等人[7]提出了一种基于深度卷积神经网络和循环神经网络的车道线检测系统。Gurghian[8]提出了一种使用两个摄像头组成立体视觉的车道线检测系统。

Davy Neven等人[9]设计了LaneNet算法，提出了一种多任务的车道线检测模型，分别进行车道线检测和车道分割，该算法在TuSimple数据集上的准确率达到了96.4%。潘新钢等人[10]提出了空间卷积神经网络（SCNN），该方法将传统的层状卷积变为片状卷积，将行与列也看作是层，特别适用于检测目标为条状的物体，在TuSimple数据集检测的准确率为96.53%。2019年，ENet-SAD被提出来[11]，它是一种基于ENet的轻量级车道检测模型，与SCNN相比，它的参数减少了20倍，运行速度提高了10倍，在CULane测试集上的表现优于SCNN。在TuSimple测试集中，它的准确率达到了96.64%同样优于SCNN的96.53%，在BDD100K测试集中，它的准确率达到了36.56%，也优于SCNN的35.79%。Visin F等人[12]将RNN应用在了车道识别任务中。Zou等人[13]利用LSTM，将多帧图片同时处理，使模型能够检测时间维度上的信息。[14]、[15]同样使用RNN改进网络。

目前，基于计算机视觉的车道检测系统已经达到了很高的准确度，但是在复杂环境中的检测准确率与稳定性还有待提高。现有的方法需要依赖大量的计算，虽然模型的计算速度不断提高，计算机硬件计算能力也不断提升，但是在对移动端硬件实现的需求与日俱增的情况下，还是需要研究更高效的检测算法。[16]提出

了改进的 U-Net 模型，可以用来快速提取特征，减小计算成本。同样[17]提出了压缩模型的办法，减小预测时的计算压力。[18]提出了 Fast Lane，可以快速地进行语义分割，检测不固定数量的车道。

1.3 研究内容

如前文所述，车道检测是自动驾驶系统中的关键技术。由于深度学习算法在车道检测任务中已经展现出巨大的优势，所以本文主要研究在单目视觉下，使用深度学习算法实现对车道线的检测。

车道线检测是一种语义分割任务，对于道路图片，通过网络模型计算得出图片中属于车道线部分的像素。然后将这些像素使用曲线拟合，在原图片上标注为车道线。网络最多输出四条车道线，即车辆左侧、右侧和车辆所在车道这三条车道的四条车道线。

为了解决城市环境中车道线检测的难点和问题，本文从以下几点入手，来着手解决。

（1）使用 U-Net 作为语义分割基础网络，时序神经网络（TCN）联系多张图片，实现了在时间维度上联系多帧图片的语义分割网络。

（2）为了解决车道线检测的误检、漏检问题，本文单独训练了一个子网络，称其为车道线存在性判断网络，来判断每条车道的存在性。

（3）针对车道线检测这种类别不均衡的问题，对比使用了 Focal Loss 和 Dice Loss 两种损失函数来消除类别不均衡对网络训练的不利影响。

（4）复现了通过 LSTM 方法联系多帧图片的算法，在同一数据集上训练，证明了本文提出方法的有效性。

本文的所有网络模型均使用 CULane 数据集进行训练、验证和测试。

1.4 本文组织结构

本文一共有五个章节，

第一章为绪论，介绍在当前世界背景下，车道线检测技术的重要意义和国内外发展现状。阐述本文的研究内容与难点。

第二章介绍车道线检测的流程，从输入图片的预处理、网络的构建与后处理三方面入手，详细介绍了使用深度学习方法检测车道线的流程。

第三章为本文构建的车道线检测算法模型和用于对比的网络模型，包括网络

结构、损失函数的选择和网络输出的后处理过程。

第四章为文章的实验部分，介绍了使用的数据集、评价指标和实验的软、硬件环境，列出了实验结果，并对实验结果进行分析、评价和总结。

第五章为总结与展望，对本文的方法进行总结，详细阐明方法的先进性与不足之处，阐述下一步可能的研究方向。

2 车道线检测流程

基于深度学习的车道线检测需要进行输入图片预处理、网络模型计算、后处理三个步骤，而网络模型需要训练后才能使用，流程如图 2.1 所示。

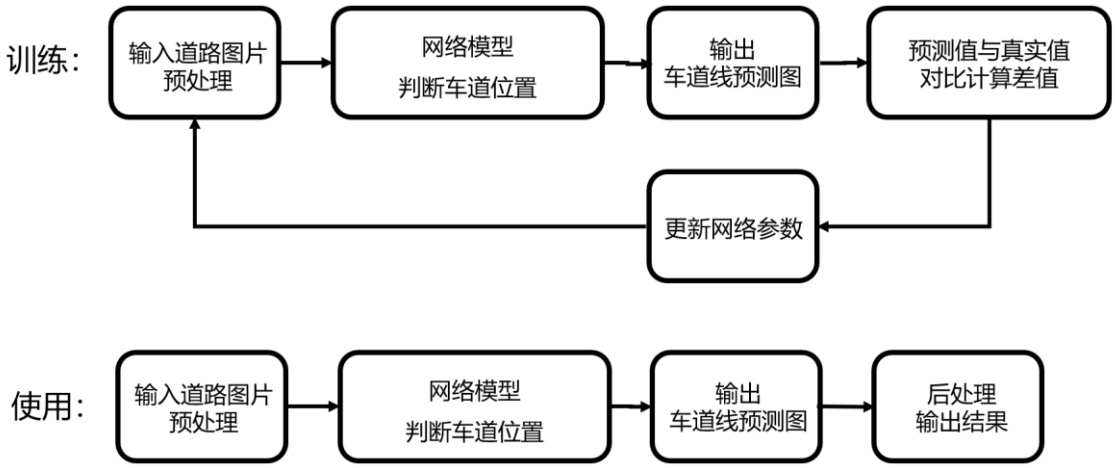

图 2.1 车道检测流程图

2.1 道路图片预处理

道路图片一般是由安装在车辆前方的车载摄像机采集到的，这些图片一般为彩色 RGB 三通道图像，本质是三个同样尺寸的矩阵，取值范围为[0,255]。为了能在之后的网络优化中让网络更快更好地拟合数据，需要对图片数据进行归一化处理。

归一化是训练神经网络中对输入数据的一个重要处理方法。通常，归一化主要有两个步骤，零均值和归一化方差。

$$\begin{gathered}\mu=\frac{1}{m}\sum_{i=1}^{m}x^{(i)}\ ,\ \sigma^2=\frac{1}{m}\sum_{i=1}^{m}(x^{(i)})^2 \\ x'=x-\mu\ ,\ x'=x\div\sigma^2\end{gathered} \tag{2.1}$$

式中，x' 与 x 分别为归一化之后和之前的输入值，m 为输入数据的特征数量。对于 RGB 图片来说，m 等于像素值乘通道数。但是在车道线检测任务中，不希望输入数据是零均值的，所以在本文中，直接对输入图片的像素值除以 225，使像素值的取值范围变为[0,1]即可。

2.2 网络模型构建

网络模型的构建是深度学习任务中最为重要的工作，其中的工作包括网络模型的结构构建和参数构建。结构构建通常是根据自己的需要，对已有的网络模型进行修改，以适应具体任务。而参数的构建需要使用基于梯度下降法的优化方法不断迭代来优化参数，这一过程被称为训练。

2.2.1 深度学习网络模型概述

普通神经网络：

在机器学习领域，神经网络可以看作是一种特殊的函数。神经网络由输入层、隐藏层和输出层组成，输入层负责输入数据，输出层则输出最后的预测值，如图 2.2。对于隐藏层，当在使用监督学习训练时，里面的数据是不会直接表现出来的，当然也不必要知道的。

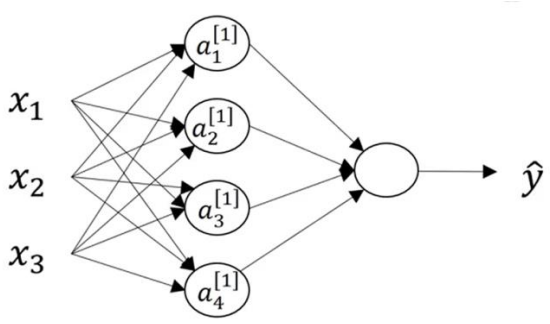

图 2.2 神经网络图

$$\begin{aligned} z_1^{[1]} &= w_1^{[1]T}x + b_1^{[1]} \quad , \quad a_1^{[1]} = \sigma(z_1^{[1]}) \\ z_2^{[1]} &= w_2^{[1]T}x + b_2^{[1]} \quad , \quad a_2^{[1]} = \sigma(z_2^{[1]}) \\ z_3^{[1]} &= w_3^{[1]T}x + b_3^{[1]} \quad , \quad a_3^{[1]} = \sigma(z_3^{[1]}) \\ z_4^{[1]} &= w_4^{[1]T}x + b_4^{[1]} \quad , \quad a_4^{[1]} = \sigma(z_4^{[1]}) \end{aligned} \tag{2.2}$$

$$\sigma = \frac{1}{1+e^{-x}} \tag{2.3}$$

$$\hat{y} = \sigma(w^{[2]T}a^{[1]} + b^{[2]}) \tag{2.4}$$

网络中的参数 w 开始是随机的，b 一般为 0，网络根据公式(2.2)、(2.3)、(2.4)计算后，得到一个输出 $\hat{y}$，这一过程称为正向传播。

$\hat{y}$ 是一个预测值，为了计算预测值和实际值之间的差距，我们可以设计一个

简单的损失函数：

$$L=(\hat{y}-y) \tag{2.5}$$

其中 y 为真实值。

然后由损失函数，计算损失 L 对上述公式中每个参数 w，b 的偏导数，

$$dW=\frac{\partial L}{\partial w}\ ,\ db=\frac{\partial L}{\partial b} \tag{2.6}$$

然后使用梯度下降法，得到新的参数，这一过程称为反向传播。

$$\begin{aligned} W' &= W-\alpha dW \\ b' &= b-\alpha db \end{aligned} \tag{2.7}$$

α 为梯度下降的步长。

经过多次的训练后，网络的输出 $\hat{y}$ 与真实值 y 的差距逐渐缩小，当缩小到一定程度之后，就要停止训练，否则就会出现过拟合的情况。训练完成的网络，就可以拿来预测新的值，达到了建模的目的。

经过研究发现，增加隐藏层层数，网络的效果会更好。所以出现了深度学习一词，所谓深度学习，就是通过有很多层隐藏层的神经网络构建出的算法，不断进行正向传播、反向传播来拟合数据集数据的过程。

卷积神经网络（CNN）：

自 2012 年 AlexNet[19]在 ImageNet 图像分类大赛上以大幅领先第二名的成绩获得冠军以来，在处理图像相关的任务上，卷积神经网络表现出了势不可当的劲头。近年来的车道线检测算法都是基于卷积神经网络来构建的。

通常，卷积神经网络包含卷积层和池化层。d

卷积层进行的是卷积操作，如图 2.3 所示。例如一张单通道的灰度图片，其本质是长度和宽度均为 5 的矩阵，设计一个卷积核，卷积核的通道数与输入图片的通道数必须相同，卷积核的长和宽均为 3，把卷积核放在输入图片上，从左到右，从上到下，将输入图片和卷积核中对应位置的数值相乘，再将这九个数相加，得到一个数值，将这个数值按照卷积核滑动的顺序填在一个新的矩阵当中，这个新的矩阵就是输出。这一过程类似于普通神经网络的求解过程，卷积核中的参数相当于对输入矩阵对应位置的数值进行了一次线性变换后再求和。所以在反向传播的过程中，需要求解出损失函数对卷积核中每一个参数的导数，然后同样采用梯度下降法进行参数更新。

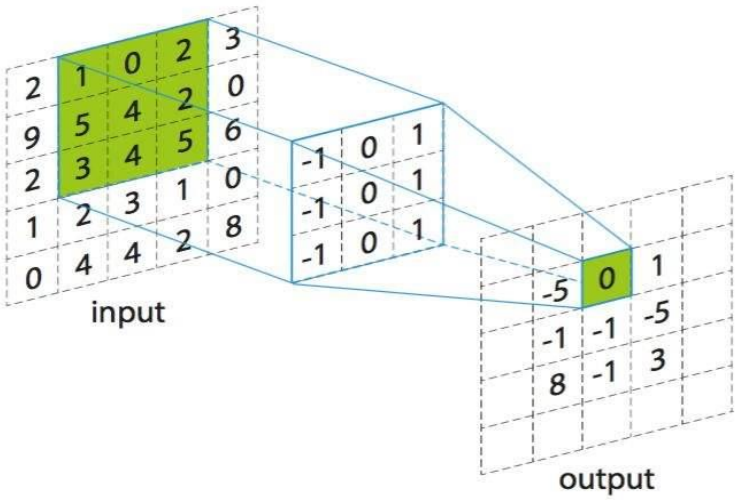

图 2.3 卷积神经网络中的卷积操作

卷积神经网络在卷积操作的过程中，涉及到卷积核大小的设定，通常卷积核设置为 3×3×D，D 为输入图片的通道数。除此之外，还需要设定卷积核的滑动步长，通常步长设置为 1，即每一次向右或者向下滑动均为滑动一格。这样一来就造成了输出矩阵的尺寸一定要比输入矩阵的尺寸小的情况，因此，我们还需要在原图周围设置空白填充，这样就可以做到输出矩阵和输入矩阵的尺寸相等。

池化层进行池化操作，池化层主要是为了进行下采样，可以增大更深层网络单个像素点的感受野，同时还有避免过拟合的作用[20]。

在池化层中，同样需要设置核的宽度，通常设置为 2×2。池化有平均池化和最大化池化，平均池化就是将核内的数值取平均数，填在新的矩阵当中，最大化池化则是直接取核中最大的数值，如图 2.4 所示。可以看到，经过池化层之后，输出矩阵的长宽尺寸会变为输入矩阵长宽尺寸的一半。

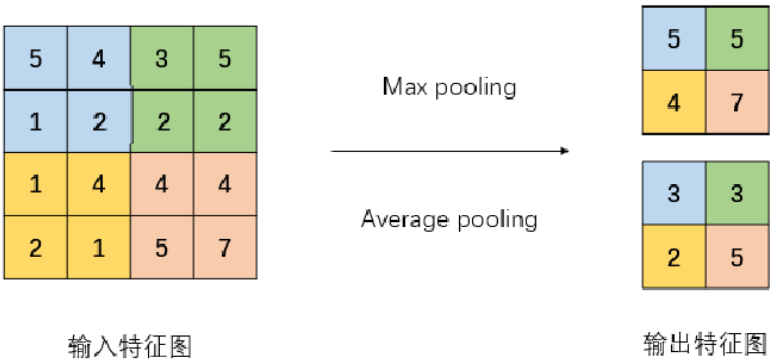

图 2.4 卷积神经网络中的池化操作

循环神经网络(RNN)：

循环神经网络通常被应用在音频识别、自动翻译等自然语言处理任务。在这些任务中，每一个输出不仅要受到当前输入的影响，还有受到之前或之后输出的影响。例如在自动翻译项目上，当前单词翻译出来的内容不仅仅要考虑单词本身的含义，还要参照前文和后文的内容。

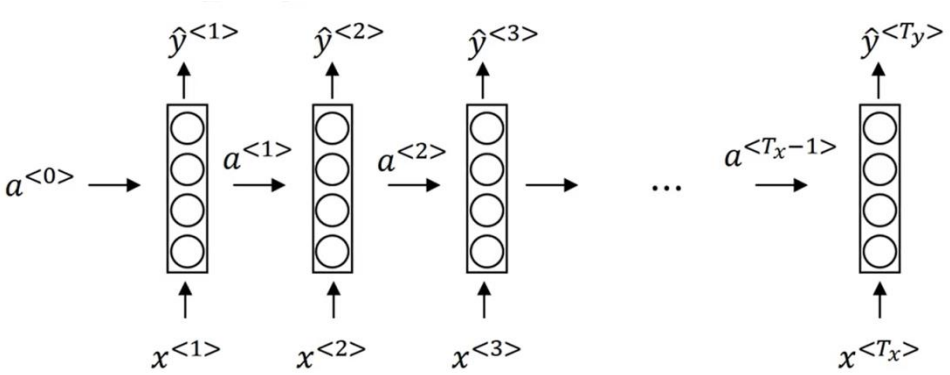

图 2.5 循环神经网络（RNN）

如图 2.5 所示，后一层网络输出不仅受当前输入的影响，还会受到之前层网络输出的影响。

在 t 时刻，

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{2.8}$$

$$\hat{y}^{<t>} = g_2(W_{ya}a^{<t>} + b_y) \tag{2.9}$$

其中，g 为激活函数，W 与 b 为网络参数。

在循环神经网络中，出现的一个问题是经过很多层的参数相乘与激活后，很可能会出现梯度消失的问题，即经过多层的传播之后，后面网络层很难有效的接收到更靠前的层传来的信息，所以有人设计了长短期记忆（LSTM）[21]的方法来解决这一问题。已经有相当多的研究表明了这种方法的有效性。

对于涉及时间序列的问题，通常采用循环神经网络来实现，因为循环神经网络的网络结构更适合沟通时序信息，而卷积神经网络受限于网络结构，很难传递时间上的信息。为了解决这一问题，时序卷积神经网络（TCN）[22]应运而生，时序卷积神经网络的本质还是一种卷积神经网络，如图 2.6 所示，但是其通过对多个输入进行卷积的操作实现了在时间序列上联系多个输入的效果，解决了传统卷

积神经网络只能处理单帧图片，无法联系前后文信息的问题，在很多问题上，这种网络结构的表现甚至超过了循环神经网络。

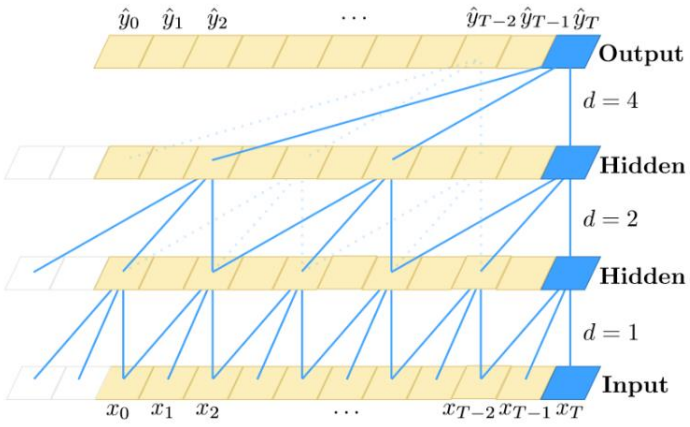

图 2.6 时序卷积神经网络

为了能让卷积神经网络能联系时间信息，时序卷积神经网络采用了一种膨胀卷积的方式，这样一来一个输出就可以受到多个输入的共同影响。

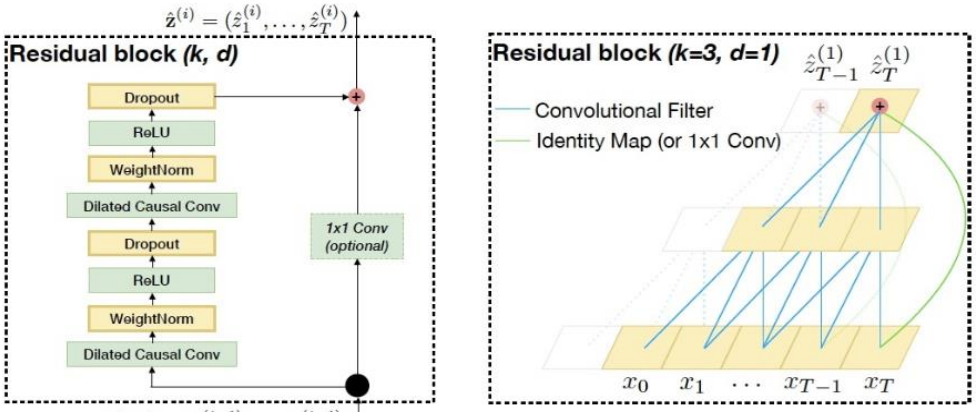

图 2.7 TCN 的残差结构

同样为了解决梯度消失和梯度爆炸的问题，TCN 在跨层网络之间添加了残差层，如图 2.7 所示，避免长距离传递信息造成的问题。

2.2.2 车道线检测网络模型构建

车道线检测问题可以看作是一种语义分割的问题，图像识别与语义分割是计算机视觉最有难度也是最基础的任务之一。语义分割就是将图片的每个像素划分为某一类的过程。在车道线检测任务中，就是要让每一个像素点判断它属于车道还是背景。

FCN[23]是语义分割领域的开山之作，而 U-Net[24]则是在 FCN 上的进一步发展。U-Net 在 2015 年发表于 MICCAI 的会议上，起初是作为医疗图像分割网络被提出的，但是由于其强大的能力，U-Net 已经成为了跟多图像语义分割网络的基础网络。它的 U 型结构也给后续的网络研究带来的很大的启发。

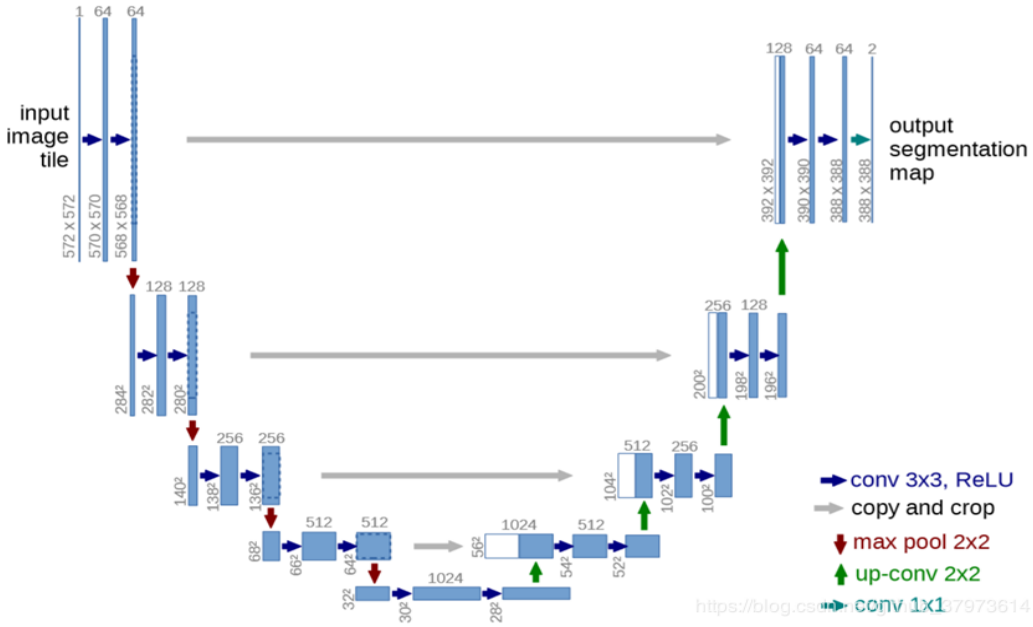

图 2.8 U-Net 网络结构图

U-Net 最大的特点就是 U 型网络结构和跳层链接。网络结构如图 2.8 所示，主要由下采样和上采样两部分组成。

下采样计算过程如下：

（1）输入的图片为 572×572 的单通道图片，通过两次填充为 0，64 个尺寸为 3×3 的卷积核卷积操作得到 568×568×64 的特征图。

（2）通过 2×2 的池化层池化变为 284×284×64 的特征图。

（3）又进行两次填充为 0，128 个尺寸为 3×3 的卷积核卷积操作得到 280×280×128 的特征图。

（4）通过 2×2 的池化层池化变为 140×140×128 的特征图。

（5）通过两次填充为 0，256 个尺寸为 3×3 的卷积核卷积操作得到 136×136×256 的特征图。

（6）通过 2×2 的池化层池化变为 68×68×256 的特征图.

（7）通过两次填充为 0，512 个尺寸为 3×3 的卷积核卷积操作得到 64×64×512 的特征图。

（8）通过 2×2 的池化层池化变为 32×32×512 的特征图。

（9）通过两次填充为 0，1024 个尺寸为 3×3 的卷积核卷积操作得到 28×28×1024 的特征图。

在上采样的过程中，需要进行双线性插值计算和特征图的叠加。

为了填充特征图中未知的空白像素，需要进行双线性插值计算。

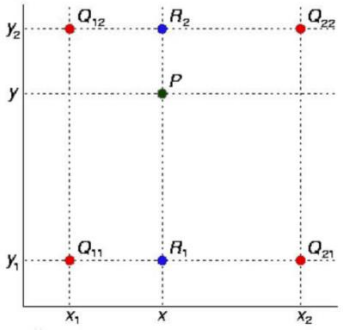

图 2.9 双线性插值

如图 2.9 所示，即已知$Q_{11},Q_{12},Q_{21},Q_{22}$四点像素值，求 P 点像素值，有，

$$R_1 = \frac{x_2 - x}{x_2 - x_1} Q_{11} + \frac{x - x_1}{x_2 - x_1} Q_{21}$$

$$R_2 = \frac{x_2 - x}{x_2 - x_1} Q_{21} + \frac{x - x_1}{x_2 - x_1} Q_{22} \qquad (2.10)$$

$$P = \frac{y_2 - y}{y_2 - y_1} R_2 + \frac{y - y_1}{y_2 - y_1} R_1$$

式中的 P 为 P 点的像素值。

上采样计算过程如下：

（1）进行上采样，尺寸变为 56×56×512。

（2）与下采样对应特征图进行层维度的叠加，变为 56×56×1024，再进行两次填充为 0，512 个尺寸为 3×3 的卷积核卷积操作得到 52×52×512 的特征图。

（3）进行上采样，尺寸变为 104×104×256。

（4）与下采样对应特征图进行层维度的叠加，变为 104×104×512，再进行两次填充为 0，256 个尺寸为 3×3 的卷积核卷积操作得到 100×100×256 的特征图。

（5）进行上采样，尺寸变为 200×200×128。

（6）与下采样对应特征图进行层维度的叠加，变为 200×200×256，再进行两次填充为 0，128 个尺寸为 3×3 的卷积核卷积操作得到 196×196×128 的特征

图。

（7）进行上采样，尺寸变为 392×392×64。

（8）与下采样对应特征图进行层维度的叠加，变为 392×392×128，再进行两次填充为 0，64 个尺寸为 3×3 的卷积核卷积操作得到 388×388×64 的特征图。

（9）最后经过一次填充为 0，2 个尺寸为 1×1 的卷积核卷积操作得到 388×388×2 的分割图。

U-Net 的下采样加上采样的结构非常适合在此基础上做一些变体。在经过了计算之后，得到 388×388×2 的预测图，一个通道为车道线，一个通道为背景。然后计算损失函数，再使用基于梯度下降法的各种优化方法对网络参数进行不断优化，让预测值和真实值的差距越来越小，然后网络模型就可以拿来使用。

2.3 车道线预测及后处理

使用训练好的模型来预测车道，同样需要先将收集到的图片进行预处理，然后输入到网络中进行计算，计算得到的输出就是车道线的预测结果，例如图 2.10。

图 2.10 输入图片与网络输出

一张预测图虽然在图片中标记出了车道线的位置，但是无法直接使用，需要进行一些处理。通常，会对车道线使用曲线进行拟合。如果能够确定相机与道路的角度，还可以计算得到车道线的曲率，从而得到车道的弯曲程度。

2.4 城市环境中车道线特点

（1）交通环境复杂。城市环境中车道线经常会被遮挡，尤其是在交通环境较为拥堵时，车道线往往全部不可见。

（2）光照情况复杂。道路两旁的建筑、树木枝叶投下的阴影会对检测造成不利影响。夜间和不良天气情况下，路灯车灯的光投射在路面上也会干扰检测。

（3）交叉口较多。城市环境中的交叉路口较多，容易产生误检。

（4）城市道路往往没有中央隔离带，所以车辆上的相机会拍到对向车道，可能会导致错误地检测出对向车道的车道线。

（5）车道线是细长物体，在图片的所有像素中只占很少的一部分，在语义分割任务中是一种典型的类别不均衡的任务，给网络的训练带来了一些不利影响。

2.5 本章小结

本章介绍了基于深度学习算法的车道线检测流程，对深度学习中神经网络、卷积神经网路、循环神经网络的计算过程、适用范围和利弊进行了阐述。

3 基于时序神经网络构建的车道线检测算法(UTCN）

为了能够将时序信息添加到车道线检测算法中，充分发挥卷积神经网络的优势，故构建了由主网络、子网络两个网络组成的大网络模型来检测车道线，为了方便，称这个大网络模型为 UTCN。主网络为基于时序神经网络和 U-Net 构建的网络，用来判断车道线位置，子网络为车道线存在性判断网络。算法整体流程图如图 3.1 车道线检测算法流程图所示。

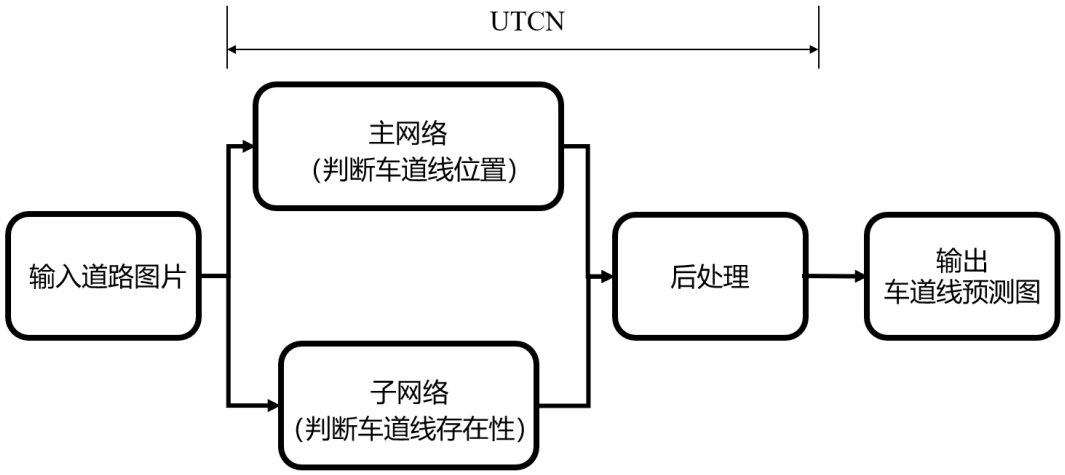

图 3.1 车道线检测算法流程图

3.1 主网络整体结构

网络整体结构采用语义分割网络典型的编解码结构，依次由下采样、时序融合和上采样三部分组成。如图 3.2 所示。

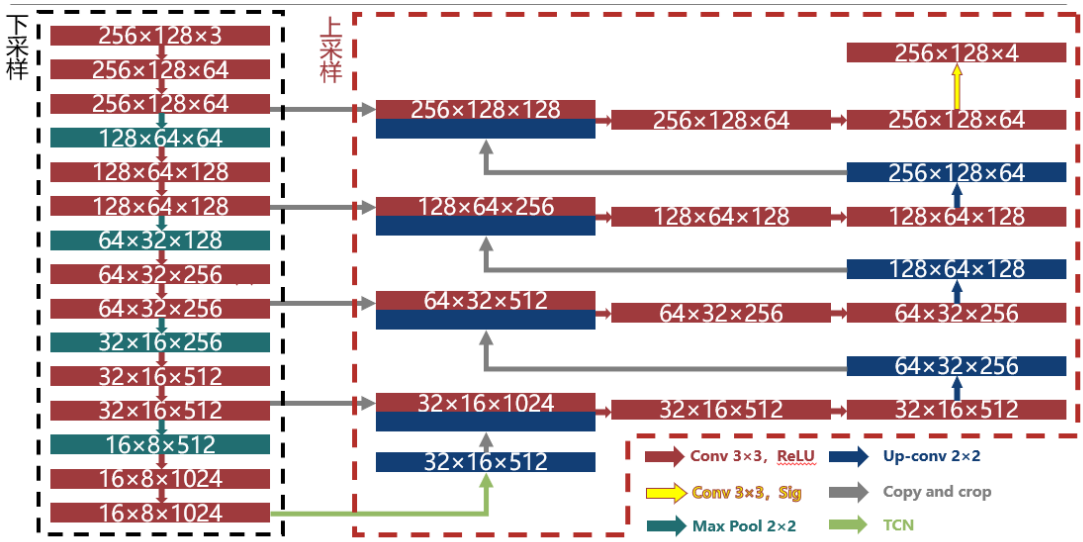

图 3.2 UTCN 网络图

为了提高网络得到训练速度，除最后一层外，均采用线性整流函数（ReLU）作为激活函数，

$$R(x)=\max(0,x) \tag{3.1}$$

$R(x)$ 为 ReLU 激活函数，x 为输入。

最后一层的输出中，为了使值限定在（0,1）的范围内，使用 sigmoid 激活函数，

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{3.2}$$

$\sigma(x)$ 为 Sigmoid 激活函数，x 为输入

下采样与上采样的计算步骤如下，

（1）输入图片为彩色三通道 RGB 图像，三个通道分别为蓝色、绿色红色三个通道。

（2）为了让输出与出入有同样的尺寸，设置填充为 1。通过两次 64 个尺寸为 3×3 的卷积核卷积操作得到 256×128×64 的特征图。

（3）通过 2×2 的池化层池化变为 128×64×64 的特征图。

（4）又进行两次填充为 1，128 个尺寸为 3×3 的卷积核卷积操作得到 128×64×128 的特征图。

（5）通过 2×2 的池化层池化变为 64×32×128 的特征图。

（6）通过两次填充为 1，256 个尺寸为 3×3 的卷积核卷积操作得到 64×32

×256 的特征图。

（7）通过 2×2 的池化层池化变为 32×16×256 的特征图.

（8）通过两次填充为 1，512 个尺寸为 3×3 的卷积核卷积操作得到 32×16×512 的特征图。

（9）通过 2×2 的池化层池化变为 16×8×512 的特征图。

（10）通过两次填充为 1，1024 个尺寸为 3×3 的卷积核卷积操作得到 16×8×1024 的特征图。

（11）下采样得到的特征图进行时序处理，处理完成后，尺寸仍然是 16×8×1024

（12）进行上采样，尺寸变为 32×16×512。

（13）与下采样对应特征图进行层维度的叠加，变为 32×16×1024，再进行两次填充为 1，512 个尺寸为 3×3 的卷积核卷积操作得到 32×16×512 的特征图。

（14）进行上采样，尺寸变为 64×32×256。

（15）与下采样对应特征图进行层维度的叠加，变为 64×32×512，再进行两次填充为 1，256 个尺寸为 3×3 的卷积核卷积操作得到 164×32×256 的特征图。

（16）进行上采样，尺寸变为 128×64×128。

（17）与下采样对应特征图进行层维度的叠加，变为 128×64×256，再进行两次填充为 1，128 个尺寸为 3×3 的卷积核卷积操作得到 128×64×128 的特征图。

（18）进行上采样，尺寸变为 256×128×64。

（19）与下采样对应特征图进行层维度的叠加，变为 256×128×128，再进行两次填充为 1，64 个尺寸为 3×3 的卷积核卷积操作得到 256×128×64 的特征图。

（20）最后经过一次填充为 1，4 个尺寸为 3×3 的卷积核卷积操作得到 256×128×4 的分割图。

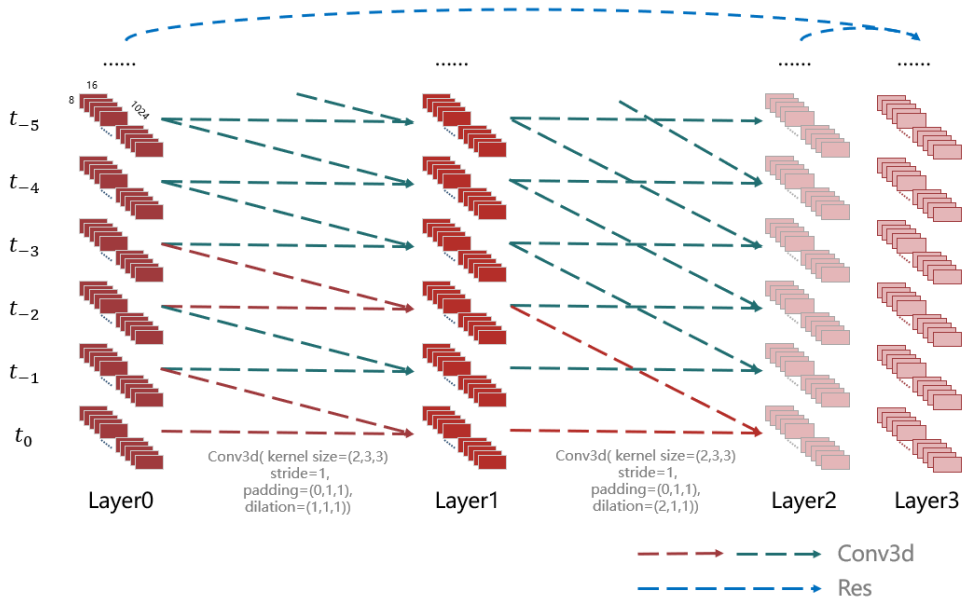

图 3.3 时序神经网络模块网络图

时序融合的过程就是采用 TCN 的思想，将多个下采样得到的特征图按照时间顺序排列起来，然后进行特殊的卷积操作，从而使其能够融合时间信息。具体过程如图 3.3 所示。第零层的每一个特征图分别代表一张图片经过下采样后得到的尺寸为 16×8×1024 的特征图。当前时刻采集到的图片经下采样后得到的特征图为第零层的t_0，上一时刻采集到的图片经过下采样后得到的特征图为t_{-1}，以此类推。对第 0 层的特征图，使用卷积操作。卷积核尺寸由传统的 C×W×H 变为 D×C×W×H，其中，D 为深度，C 为通道数，W 与 H 为核宽度和高度，深度 D 即每次卷积涉及到的特征图个数，对于第零层，D=2，即每次卷积同时联系两个相邻特征图。卷积核的通道数必须与输入特征图通道数相等，不能也不需要人为设定。这里取卷积核宽与高尺寸为 3×3。经过第一次卷积和激活之后，得到了第一层输出，第一层的特征图个数与第零层完全相同，每个特征图的尺寸也完全相同。

接下来进行第二次卷积，第二次卷积操作与第一次基本一致，不同的是，在第二次卷积中，卷积核不再联系相邻的两个特征图，而是联系相间隔的两个特征图。同样的，完成第二次卷积操作后，特征图的个数、每个特征图的尺寸与第一层和第零层均相等。

经过这样的两次卷积后，可以很清楚的看到，第二层的每一个特征图都是由第零层的四个特征图计算得到的。如第二层t_0时刻计算出的特征图，是由第零层的t_0,t_{-1},t_{-2},t_{-3}四个时刻的特征图计算得到的。这就让卷积神经网络拥有了类似于

循环神经网络可以联系时间信息的特性。

为了避免信息在两次传递中出现损耗，在第零层和第二层之间加入了残差操作，即，

$$X_{l3}^{t0} = X_{l2}^{t0} + X_{l0}^{t0} \tag{3.3}$$

最后，第三层 t_0 时刻的特征图将会继续进行上采样的计算。

3.2 车道线存在性判断网络

为了更好地判断车道线是否存在，减少误检，漏检的情况，本文单独设计了一个子网络来判断车道线是否存在。

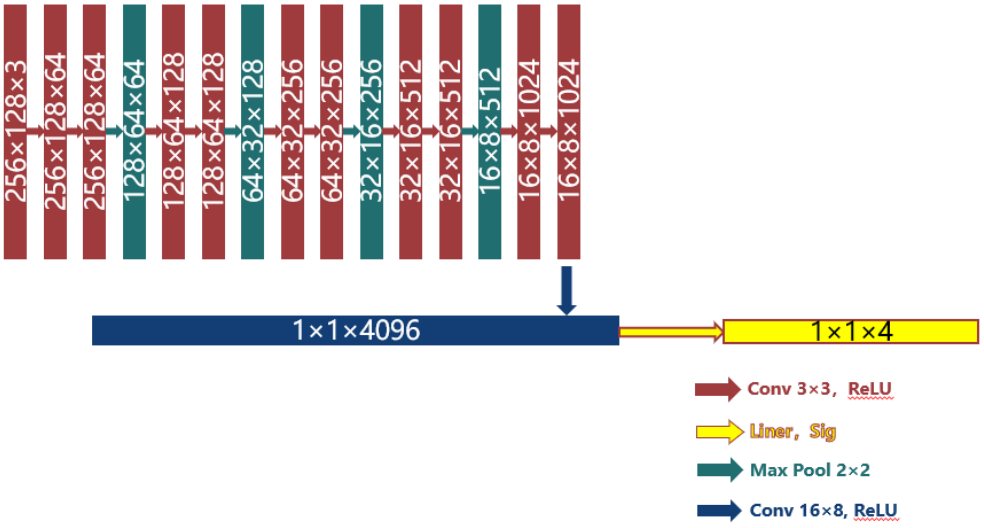

图 3.4 车道存在性判断网络图

网络结构如图所示，特征提取部分与前文主网络的下采样部分相同，

（1）输入图片为彩色三通道 RGB 图像，三个通道按顺序分别为蓝色、绿色、红色三个通道。

（2）为了让输出与出入有同样的尺寸，设置填充为 1。通过两次 64 个尺寸为 3×3 的卷积核卷积操作得到 256×128×64 的特征图。

（3）通过 2×2 的池化层池化变为 128×64×64 的特征图。

（4）又进行两次填充为 1，128 个尺寸为 3×3 的卷积核卷积操作得到 128×64×128 的特征图。

（5）通过 2×2 的池化层池化变为 64×32×128 的特征图。

（6）通过两次填充为 1，256 个尺寸为 3×3 的卷积核卷积操作得到 64×32×256 的特征图。

（7）通过 2×2 的池化层池化变为 32×16×256 的特征图.

（8）通过两次填充为 1，512 个尺寸为 3×3 的卷积核卷积操作得到 32×16×512 的特征图。

（9）通过 2×2 的池化层池化变为 16×8×512 的特征图。

（10）通过两次填充为 1，1024 个尺寸为 3×3 的卷积核卷积操作得到 16×8×1024 的特征图。

（11）当特征图变为 16×8×1024 之后，使用 4096 个大小为 16×8 的卷积核进行卷积，得到了尺寸为 1×1×4096 的特征图，然后再进行一次全连接操作，得到 1×1×4 的输出，即 4 个数字。因为采用了 Sigmoid 作为全连接层的激活函数，所以，网络输出的四个数的值在(0,1)内分布。这四个数分别代表着四条车道的存在概率，当存在概率大于 0.8 时，即可视为该车道存在，否则不存在。

3.3 对比网络

为了验证本文网络的有效性，实现了两个用于对比的网络。

3.3.1 基于 LSTM 构建的车道检测算法(ULSTM)

2019 年，Qin Zou 等人 42[13]同样在 U-Net 的基础上设计了车道线检测网络，使用 LSTM 联系多帧图片，实现了联系时间上的信息来检测车道线，结构如图 3.5 所示为了方便，称其为 ULSTM。

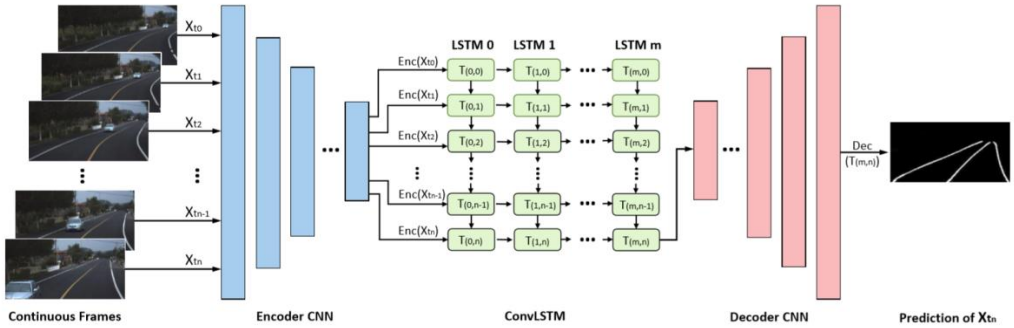

图 3.5ULSTM 网络图

网络结构除 LSTM 部分之外，与本文的网络结构相同，所以可以用来对比使用 TCN 融合时间信息的有效性。

3.3.2 不融合时间信息的 U-Net

为了验证融合时间信息对网络输出造成的影响，使用基础的 U-Net 进行对比。网络结构与本文主网络的编解码网络部分相同。

3.4 损失函数

车道线检测是一种典型的类别不均衡任务，车道线的像素只占图片所有像素的一小部分。如果简单的使用普通的交叉熵损失函数，在这种情况下，即便网络输出全为背景，也能得到很高的准确率，损失函数也会很小，但是很显然，这样的网络是没有意义的。为了解决类别不均衡带来的不利影响，就必须要使用适当的损失函数。

3.4.3 Focal Loss 损失函数

Focal Loss[25]这一损失函数是由何凯明团队为了解决目标检测任务中正负样本比例严重失衡的问题而提出的，这一损失函数降低了大量的负样本在训练中占的权重，在车道线检测任务中，就是降低了大量背景像素所占的权重，同时也会增加难训练样本的权重，也可以看作是一种困难样本挖掘的函数。

普通的交叉熵损失函数如下，

$$L=\begin{cases}-\log\hat{y} & , \quad y=1\\ -\log(1-\hat{y}) & , \quad y=0\end{cases} \tag{3.4}$$

式中 L 为交叉熵损失函数，$\hat{y}$ 为网络输出的预测值，y 为真实值。

可以看到，在普通的交叉熵函数中，对正负样本和简单、难分类样本的权重是相同的，不适用于类别不均衡问题。

Focal Loss 的公式如下，

$$L=\begin{cases}-\alpha(1-y')^{\gamma}\log\hat{y} & , \quad y=1\\ -(1-\alpha)y'^{\gamma}\log(1-\hat{y}) & , \quad y=0\end{cases} \tag{3.5}$$

在 Focal Loss 中，先加入了一个因子 $\gamma>0$，这样就可以使得损失函数更加关注于分类困难的、错误的样本。例如，当真实值为 1，而预测值为 0.9 时，预测结果和真实值比较接近，属于容易分类的样本，$(1-0.9)^{\gamma}$ 就是一个比较小的数值，这时损失函数就会比较小。当真实值为 1，而预测值为 0.1 时，预测结果和真实值差距较大，属于难分类的样本，这时 $(1-0.1)^{\gamma}$ 就是一个比较大的值，所以此时的损

失函数也就比较大，即增大了难分类样本在损失函数中占的权重。

此外 Focal Loss 还加入了平衡因子α，用来平衡正负样本本身的不均衡问题。通过作者研究，发现α取 0.25，γ取 2 时效果最优。

3.4.4 Dice Loss 损失函数

Dice Loss 损失函数[26]是一种基于交并比概念的损失函数，在 V-Net 中被提出，后被广泛应用在了医学影像分割当中。由于其良好的表现，同样适用于其它像素级分类任务。

$$L=\frac{2\left|\mathrm{X}\cap\mathrm{Y}\right|+1}{\left|\mathrm{X}\right|+\left|\mathrm{Y}\right|+1} \tag{3.6}$$

式中 X 为预测值，Y 为真实值。分子分母都加 1 可以避免分母为零的问题，同时减小过拟合。$\left|\mathrm{X}\cap\mathrm{Y}\right|$的含义是预测值和真实值同样为正的像素数量。但是由于网络的输出层是 Sigmoid 激活函数，所以输出值不可能非正即负。所以具体的计算过程为两个矩阵相点乘，即矩阵对应位置的像素值相乘，之后再求和。例如，

$$\begin{aligned}\left|\mathrm{X}\cap\mathrm{Y}\right| &= \begin{bmatrix}0.01 & 0.03 & 0.02 & 0.02\\ 0.05 & 0.12 & 0.09 & 0.07\\ 0.89 & 0.85 & 0.88 & 0.91\\ 0.99 & 0.97 & 0.95 & 0.97\end{bmatrix}\bullet\begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1\end{bmatrix}\\ &= \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0.89 & 0.85 & 0.88 & 0.91\\ 0.99 & 0.97 & 0.95 & 0.97\end{bmatrix}\\ &= 7.41\end{aligned}$$

即可表达预测值与真实值之间的差别情况。

由于 X 与 Y 均为一个或多个矩阵，所以通过交并比的方式计算出来的损失函数本身就受样本不均衡的影响较小，比较适合车道线检测任务。

3.5 后处理

输入图片通过网络的计算后，得到的特征图还需要进行后处理才能更好的表现车道。所以需要进行后处理过程。

首先，输入图片经过网络的计算，得到与输入图片尺寸相同的四通道图片，也可以看作是四张尺寸相同的单通道图片，网络的最后一层使用的激活函数为

Sigmoid，所以像素值取值在 0 到 1 之间。像素值为 1 代表该像素属于车道线，而像素值取 0 代表该像素为不属于车道线的背景部分。

对输出的特征图，将像素值大于 0.9 的像素点，将其像素值改为 1。对于像素值小于等于 0.9 的像素点，将其值改为 0。这样就可以得到四张四条车道线的单通道预测图。

对每张单车道预测图，用三次曲线进行拟合，即可得到车道线的拟合函数。

3.6 本章小结

本章中对本文构建的车道线检测主网络、车道存在性判断网络的结构和计算过程进行了详细介绍。在此基础上详细阐释了损失函数对样本不均衡问题的重要意义。

4 车道线检测方法实验分析

4.1 车道线数据集

对于所有的监督学习任务来说，数据集的作用都是至关重要的。神经网络中的各项参数，都是根据数据集来不断调整，拟合。数据集的质量甚至可以直接决定网络模型的表现。对于车道线检测任务来说，从无到有创建一份数据集是一份极度费时、费力、消耗资源巨大的工作。目前已经公开的大部分有关于车道检测的数据集要么数据量太小，要么质量不高。所以本文采用目前最大的车道检测公开数据集 CULane 来进行网络模型的训练、验证和测试。

CULane 是一个在北京城市环境中收集到的数据集，相较于其他数据集，更符合国内的道路环境，一定程度上避免了数据集与实际采集到的数据出现不同分布的情况。该数据集中用于语义分割的部分总共有 88054 张图片原图和标签，原图的尺寸为 1640×590×3，标签的尺寸为 1640×590×1，标签最多标注了四条车道线，当路面标线不清、标线被遮挡时依然进行了标注。

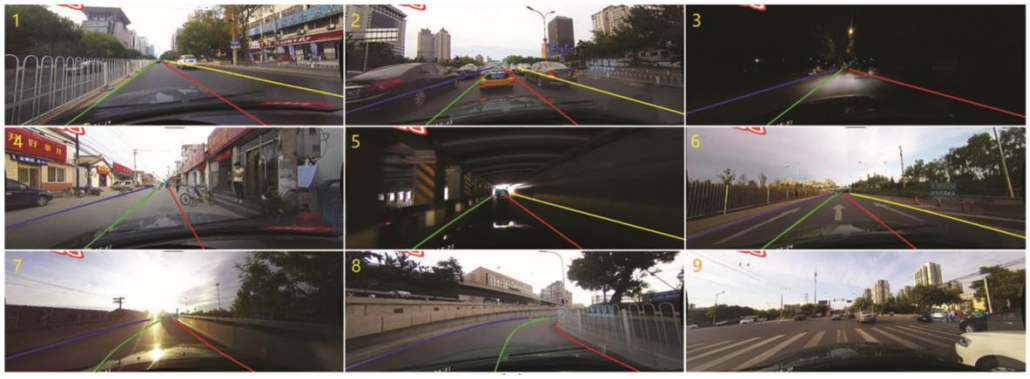

图 4.1 CULane 数据集示例

将这些图片按照 7：1.5：1.5 的比例，分割为训练集、验证集和测试集。所以本文使用的数据集共包含训练集图片 61911 张，验证集图片 13063 张，测试集图片 13080 张。

4.2 评价指标

对车道检测任务的评价，如果简单的计算准确率并不能准确全面的反映网络模型的效果，所以本文采用了评价语义分割任务常用的 F1-Measure 评价方法。

混淆矩阵是 F1-Measure 评价的基础

表 4.1 混淆矩阵

混淆矩阵		预测值	
		正	负
真实值	正	True Positive(TP)	False Negative(FN)
	负	False Positive(FP)	True Negative(TN)

在本文的车道线检测任务中，

True Positive(TP)表示网络输出认为存在车道线，且车道线预测正确。

False Positive(FP)表示网络输出认为存在车道线，但是车道线预测错误。

False Negative(FN)表示网络输出认为不存在车道线，但实际存在车道线。

True Negative(TN)表示网络输出认为不存在车道线，且实际也不存在车道线。

网络模型进行测试时，需要将以上四个指标一直累加，直至结束。

评价时，主要用到四个指标，

（1）精准率（Precision），表示预测准确的车道线数量与预测总数量的比值，

$$\text{Precision} = \frac{TP + TN}{TP + FN + FP + TN} \tag{4.1}$$

（2）召回率（Recall），表示模型的查全率，即被检测出的车道线数量与全部存在的车道线数量的比值，

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

（3）F1-Measure 是精准率和召回率的调和平均值，

$$\text{F1-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} \times \text{Re call}} \tag{4.3}$$

（4）为了判断一条车道线是否被准确的检测到，需要计算预测到的车道线与标签中标注车道线的交并比（Intersection-over-Union，IoU）。

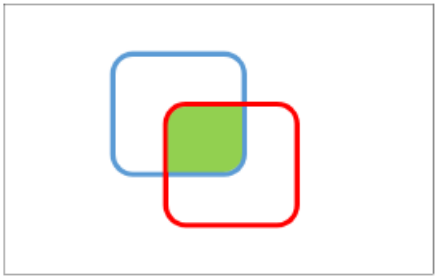

图 4.2 交并比

如图 4.2 所示，蓝框为网络模型预测出的车道线像素区域，用 SR（Segmentation）

表示，红色框表示标签中车道线像素区域，用 GT（Grand Truth）表示，绿色部分为二者的交集，则交并比的计算公式如下，

$$\mathrm{IoU}=\frac{\mathrm{SR}\cap\mathrm{GT}}{\mathrm{SR}\cup\mathrm{GT}} \tag{4.4}$$

车道线检测网络模型的评价流程如下：

（1）输入图片，得到主网络输出的四张单通道预测图，同时得到车道存在性判断网络输出的车道存在性概率。如果概率大于 80%，则判断车道存在，否则不存在。

（2）将车道线使用曲线进行拟合，并绘制成一张新的图片。

（3）计算拟合曲线图片与标签图片中车道线像素的交并比（IoU），当交并比大于某一阈值时，则认为车道线被准确的检测到。

（4）不断累加 TP，FP，FN，TN，在测试集全部计算结束后，计算各项评价指标。

4.3 实验环境及训练参数

本文的硬件部分，图形处理器（GPU）是一张 NVIDIA GeForce RTX 2080 Ti 显示卡，显存大小为 11G，CPU 为 Intel E5-2678 v3 ，主频 2.50GHz。软件部分的环境为，Linux 操作系统，Pytorch 深度学习框架，图像处理部分使用了 OpenCV 开源库。

图片在输入网络时，全部调整为 256×128 的分辨率，输入图片为 RGB 三通道彩色图片，经过归一化处理。

由于 UTCN 和 ULSTM 两个网络需要输入连续的四张图片，而数据集中的很多图片并不是连续采集到的，所以要将这些图片进行分组，把连续采集到的图片分为四张一组，如有 5 张连续采集到的图片，那么就可以分为两组图片，分别为第一张到第四张组成的第一组和第二张到第五章组成的第二组。这样一来，训练集的图片组数一共有 59807 组，验证集有 10970 组，测试集有 10950 组。对于处理单帧图片的 U-Net 而言，因为每次仅需要输入一张图片，所以可以把一张图片看作一组，所以该网络的训练集、验证集和测试集图片组数依然为 61911 组、13063 组和 13080 组。

对于 UTCN 和 ULSTM 的主网络，训练时的批量大小（Batch Size）为 20，对于车道存在性判断网络，训练时的批量大小设置为 30。测试时，因为要将这两个网

络同时运行，所以批量大小设置为 10。

优化方法采用 Adam 优化方法，学习率设为 0.001，两个 β 值分别设为 0.9 和 0.999，e 的值设置为1×10^{-8}。

4.4 实验结果及分析

本节的全部实验基于 CULane 数据集和上一节所述的软、硬件环境。对本文的网络模型和 3.3 节中的两个模型进行训练，经过初期的测试，发现使用 Dice Loss 作为损失函数的效果更好，所以损失函数使用 Dice Loss。

4.4.1 实验结果

所有模型经过了 20 次循环的训练，训练集和验证集的损失函数值几乎不再下降，测试集的损失函数值大于训练集和验证集，开始有回升的趋势，所以停止训练。把训练好的模型放在测试集上测试，对测试集计算精准率、召回率和 F1-Measure，计算得到的结果如表 4.2 测试集测试结果所示。

表 4.2 测试集测试结果

网络模型	是否判断车道存在性	精准率（%）	召回率（%）	F1-Measure（%）	每次输出耗时（秒）
UTCN	是	67.496	82.930	74.421	0.0163
	否	56.629	99.617	71.405	
ULSTM	是	65.600	83.666	73.540	0.0202
	否	55.196	99.905	71.107	
U-Net	是	58.712	82.806	68.707	0.0071
	否	47.865	99.271	64.580	

4.4.2 结果评价

将三种网络的精准率、召回率和 F1-Measure 三项准确性评价指标和运行时间这一运行速度评价指标计算得到后，绘制成柱状图，如图 4.3 结果对比所示。

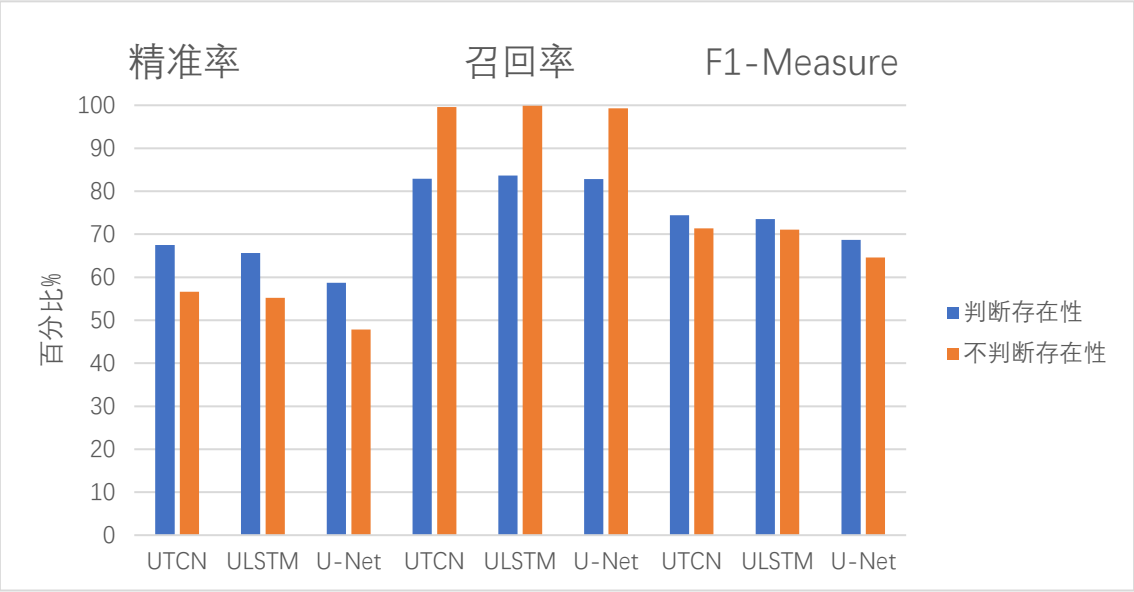

图 4.3 结果对比

在检测车道线的准确度上，可以很明显地看出，UTCN 和 ULSTM 的 F1-Measure 基本相等，UTCN 的精准率略高于 ULSTM，ULSTM 的召回率略高于 UTCN。只处理单帧图片的 U-Net 精准率最低，在判断车道线存在性的情况下，精准率比 UTCN 和 ULSTM 分别低了 8.784%和 6.888%，由此可以证明融合时间特征在车道检测任务中可以提高检测的准确性，同时，还证明了使用 TCN 和 LSTM 两种方法融合时间特征有相近的效果。

为了检验车道线存在性判断网络这一子网络存在的必要性，还测试了在不进行车道线存在性判断时网络的表现。实验结果表明，在不进行判断时，网络的精准率分别降低了 10.867%，10.404%和 10.847%。当然，判断网络的作用就是隐藏掉一些其认为不存在的车道的预测图，所以在不进行判断时，主网络几乎是输出了所有可能的车道线，也因此在没有判断时，三个网络的召回率都接近了 100%，判断网络的存在让召回率降低了 16.687%、16.239%和 16.465%。但是综合起来看，判断网络依然让三个网络的 F1-Measure 值提高了 3.016%，2.433%和 4.127%，而且在实际应用中，宁愿让网络模型漏检了实际存在的车道，也不愿让网络模型检测到不存在的车道，所以就证明了车道线存在性网络存在的有效性和必要性。

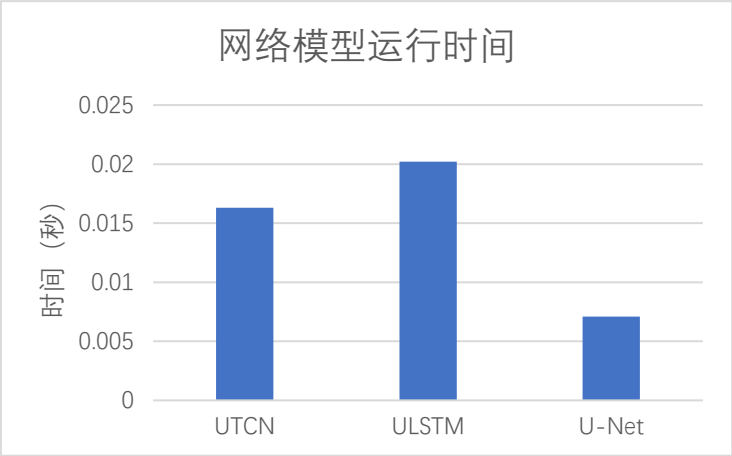

图 4.4 运行时间对比

在网络的运行时间上，处理单帧图片的 U-Net 占有巨大优势，网络模型每次输出的运行时间不到 UTCN 和 ULSTM 的一半。ULSTM 由于其复杂的计算机制，导致运行时间最长。而 UTCN 虽然运行时间要比 U-Net 要高，但是考虑到要同时处理多帧图片，在效果和 ULSTM 相近的情况下，耗时要更短，运行速度快了 19.3%，也证明了在本文的车道线检测任务中，TCN 比 LSTM 更具有时间效率优势。

为了直观对比，这里选择几张典型场景的图片，让三种网络分别输出语义分割图。

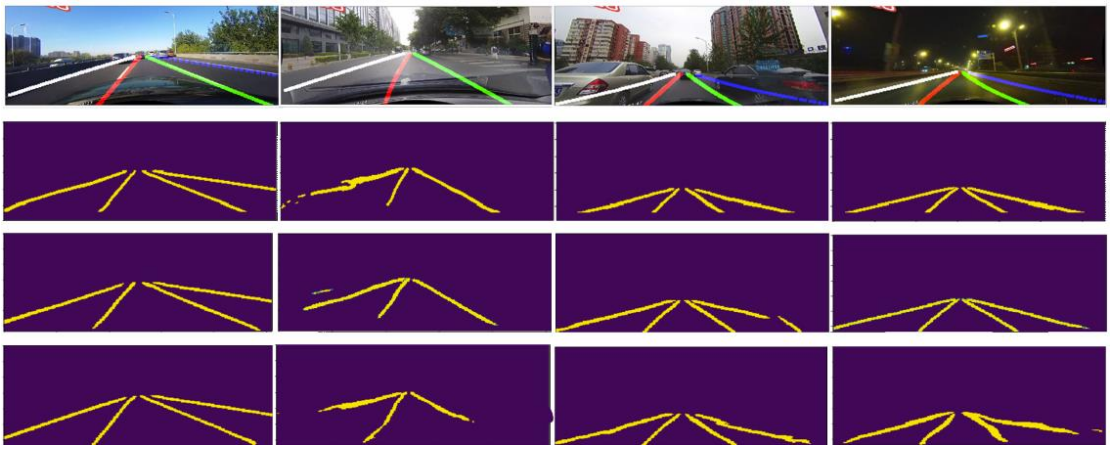

图 4.5 网络输出示例

从左到右的场景实例分别为简单、标线模糊、拥堵、夜晚；从上至下分别为曲线拟合后的 UTCN 网络输出，UTCN 网络输出分割图，ULSTM 网络输出分割图，U-Net 网络输出分割图。

由上述示例可以很清楚的看出，三种网络在路面条件良好，无遮挡、阴影干

扰，地面标线清楚的情况下，都很好的输出了结果，分割线即便不经过曲线拟合也比较平滑，很好的检测出了四条车道线。

在标线模糊的情况下，三种网络结构都能够预测出车道线的位置，但是使用单帧图片预测的 U-Net 在检测模糊的车道线时，出现了检测不全的情况。在原图中，车辆正在通过路口，所以靠近车辆端是没有车道线的，但是我们依然希望网络能够根据现场的环境预测出车道线的延长线，由于是处理的单帧图片，网络没有融合时间的特征，所以没有达到预期目标。而两个融合了时间信息的网络，可以很好的预测出车道的延长线。

在拥堵的环境中，两侧的车道线被严重遮挡，三个网络均对四条车道线做出了较为准确的预测。而没有融合时间信息下 U-Net 网络，虽然能够较为准确地预测出车道线位置，但是相对于两个融合了时间信息的网络，两侧被遮挡的车道线的置信度不高，预测图出现了波动和噪点等不稳定的现象。在夜晚条件下，道路标线在道路上各种光照的干扰下变得模糊不清，U-Net 网络出现了很不稳定的情况，而 UTCN 和 ULSTM 两个网络则可以很好的预测出车道线的位置。

综上所示,本文提出的 UTCN 相较于传统的处理单帧图片的算法，具有更高的准确度。相较于使用 LSTM 处理多帧图片的算法，准确率相近的情况下，本文的算法更具有时间效率优势，运行速度快了 19.3%。设计的车道线存在性判断网络也显著提高了三种网络的准确性和结果的可靠性。

5 总结与展望

针对当前亟待解决与优化的车道线检测问题，本文首先介绍了国内外车道线检测领域的发展现状和发展趋势，把传统的通过图像处理的检测方法和通过深度学习的检测方法进行了对比。为了更好地优化城市环境中车道线检测的速度、准确率，消除阴影、高光、遮挡、眩光和黑夜等不良条件对检测的影响，本文提出了一种通过时序神经网络联系多帧图片的车道线检测算法，训练子网络来判断车道的存在性。使用 Dice Loss 损失函数消除样本不均衡带来的不利影响，并且在同一数据集上实现了单帧检测和基于长短期记忆的两种对比网络。

实验证明，本文提出的网络算法能够很好的联系多帧连续图片的信息，相较于处理单帧图片的算法，准确率有较大的提升。相较于其他融合时间的方法，准确率基本一致，运行速度有显著提升，证明了使用时序神经网络融合时间信息的有效性。同时也证明了车道线存在性判断网络存在的有效性和必要性，实验证明，该网络能够显著的提升检测的准确性和检测结果的可靠性。

当然，本文的车道检测方法依旧有不足之处，首先，由于只使用了一种结构的时序神经网络来融合时间信息，所以相当于仅用了固定张数的图片，还没有研究不同张数的图片对网络的影响。第二，由于数据集的限制，本文中无法控制采集到图片的时间间隔，所以也没有研究采集图片的帧率（没两张图片之间的时间间隔）对网络的影响。第三，在本文提出的网络模型实现上，每次都是输入固定张数的图片进行计算，然而根据网络的结构特性，可以将之前的计算结果保存在内存中，每次计算仅输入一张图片，之后把这张图片计算得到的特征图与内存中之前的计算结果一同输入到时序神经网络中。虽然本文的算法实现方法不会对检测的精准度有任何影响，但是却会影响网络运行的速度，不适合实际应用。在未来，还需要对算法进行进一步的研究和改进。

参考文献

[1] Kang D J, Jung M H. Road lane segmentation using dynamic programming for active safety vehicles[J]. Pattern Recognition Letters, 2003, 24(16): 3177-3185.

[2] Collado J M, Hilario C, De La Escalera A, et al. Adaptative road lanes detection and classification[C]//International Conference on Advanced Concepts for Intelligent Vision Systems. Springer, Berlin, Heidelberg, 2006: 1151-1162.

[3] Cela A F, Bergasa L M, Sanchez F L, et al. Lanes detection based on unsupervised and adaptive classifier[C]//2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks. IEEE, 2013: 228-233.

[4] Wang Y, Shen D, Teoh E K. Lane detection using spline model[J]. Pattern Recognition Letters, 2000, 21(8): 677-689.

[5] Wang Y, Teoh E K, Shen D. Lane detection and tracking using B-Snake[J]. Image and Vision computing, 2004, 22(4): 269-280.

[6] Lim K H, Seng K P, Ang L M. River flow lane detection and Kalman filtering-based B-spline lane tracking[J]. International Journal of Vehicular Technology, 2012, 2012.

[7] Li J, Mei X, Prokhorov D, et al. Deep neural network for structural prediction and lane detection in traffic scene[J]. IEEE transactions on neural networks and learning systems, 2016, 28(3): 690-703.

[8] Gurghian A, Koduri T, Bailur S V, et al. Deeplanes: End-to-end lane position estimation using deep neural networksa[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016: 38-45.

[9] Neven D, De Brabandere B, Georgoulis S, et al. Towards end-to-end lane detection: an instance segmentation approach[C]//2018 IEEE intelligent vehicles symposium (IV). IEEE, 2018: 286-291.

[10] Pan X, Shi J, Luo P, et al. Spatial as deep: Spatial cnn for traffic scene understanding[C]//Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

[11] Hou Y, Ma Z, Liu C, et al. Learning lightweight lane detection cnns by self attention distillation[C]//Proceedings of the IEEE International Conference on Computer Vision. 2019: 1013-1021.

[12] Visin F, Kastner K, Cho K, et al. Renet: A recurrent neural network based alternative to convolutional networks[J]. arXiv preprint arXiv:1505.00393, 2015.

[13] Zou Q, Jiang H, Dai Q, et al. Robust lane detection from continuous driving scenes

using deep neural networks[J]. IEEE Transactions on Vehicular Technology, 2019.

[14] Wang Z, Ren W, Qiu Q. LaneNet: Real-time lane detection networks for autonomous driving[J]. arXiv preprint arXiv:1807.01726, 2018.

[15] Bell S, Lawrence Zitnick C, Bala K, et al. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2874-2883.

[16]杨威. 基于卷积神经网络的高效语义分割方法研究[D]. 中国科学院大学（中国科学院光电技术研究所）, 2019.

[17]李皈颖. 深度模型简化：存储压缩和计算加速[D]. 中国科学技术大学, 2018.

[18]侯长征. 基于视觉的车道线检测技术研究[D]. 西南交通大学, 2017.

[19] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[20] Boureau Y L, Ponce J, LeCun Y. A theoretical analysis of feature pooling in visual recognition[C]//Proceedings of the 27th international conference on machine learning (ICML-10). 2010: 111-118.

[21] Xingjian S H I, Chen Z, Wang H, et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting[C]//Advances in neural information processing systems. 2015: 802-810.

[22] Bai S, Kolter J Z, Koltun V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling[J]. arXiv preprint arXiv:1803.01271, 2018.

[23] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3431-3440.

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

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

[26] Milletari F, Navab N, Ahmadi S A. V-net: Fully convolutional neural networks for volumetric medical image segmentation[C]//2016 Fourth International Conference on

3D Vision (3DV). IEEE, 2016: 565-571.

致　谢

刚步入大学校门时，总是认为四年很漫长，等到真要结束时，才发现再漫长的时光都可以戛然而止。大学，是授之以渔，培养终身学习的能力的地方。所以，首先感谢合肥工业大学对我的培养，让我有了一生中最重要的能力，求知的能力。

在本文中应用的机器学习相关内容是我一直想要去了解，却又没有入口和动力的领域，正因如此，本文不仅是对这四年的一个总结，更是给自己的一个礼物。所以，要感谢我的指导老师陈佳佳老师，在选题之初带我入了门，在撰写文章的过程中给予了我巨大的帮助。

为了完成毕业设计，在两到三个月的时间里我全身心投入到相关基础理论的学习中，然后又用了一到两个月的时间完成模型的构想和实现。在这期间，遇到大大小小无数的问题，模型也经历了几次大改，很多数据都被废弃。但随着问题被一一解决，我对很多理论有了更深层次的认识，最后的结果已经不再那么重要，因为求知的过程就是最好的果实。在这四年里，这样的果实让我收获颇丰，所以，感谢所有同学、老师和所有帮助过我的人，我们每走一步，都是站在前人的肩膀上，这个时代是无数人奋斗出来的时代，所以更要感谢创造这个时代的所有人。

最后，要毫不吝啬的地感谢自己，不管身在何处，境遇如何，永远有一颗好奇和求知的心。

作者：周正

2020 年 5 月 25 日

附　录

UTCN 的部分网络结构代码如下：

```
class UTCN(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=True):
        super(UTCN, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = DoubleConv(n_channels, 64)
        self.down1 = Down(64, 128)
        self.down2 = Down(128, 256)
        self.down3 = Down(256, 512)
        self.down4 = Down(512, 512)
        self.up1 = Up(1024, 256, bilinear)
        self.up2 = Up(512, 128, bilinear)
        self.up3 = Up(256, 64, bilinear)
        self.up4 = Up(128, 64, bilinear)
        self.outc = OutConv(64, n_classes)

        self.tcn = TemporalBlock(512,512) #TCN

    def forward(self, x):
        xTCN = []
        for i in range(0,4):
            if i==3:
                x1 = self.inc(x[i])
                x2 = self.down1(x1)
                x3 = self.down2(x2)
```

```
                x4 = self.down3(x3)
                x5 = self.down4(x4)
                xTCN.append(x5)
        else:
                xd = self.inc(x[i])
                xd = self.down1(xd)
                xd = self.down2(xd)
                xd = self.down3(xd)
                xd = self.down4(xd)
                xTCN.append(xd)
xTCN=torch.stack((xTCN[0],xTCN[1],xTCN[2],xTCN[3]))
xTCN=xTCN.permute(1,2,0,3,4)
xTCN_output = self.tcn(xTCN)
xTCN_output = xTCN_output.permute(2,0,1,3,4)
xTCN_output = xTCN_output[3]

x = self.up1(xTCN_output, x4)
x = self.up2(x, x3)
x = self.up3(x, x2)
x = self.up4(x, x1)
logits = self.outc(x)
logits = torch.sigmoid(logits)
#logits = F.softmax(logits,1)
return logits
```