基于语义分割和分类网络的模拟自动驾驶应用

12 组　　徐昕翊 曹展翔 明陈林

摘要

结合视觉的自主避障在自动驾驶领域中有着广泛的应用。为了模拟纯视觉的自动驾驶，本文介绍一种基于语义分割网络和分类网络的深度学习模型，实现在模拟游戏平台《极品飞车 8》的自动驾驶。主要思路是利用实时采样的驾驶画面，通过语义分割网络划分出道路和车辆等障碍信息，再经过端到端的分类网络，及时准确地预测出当前车辆的运动命令。该模型综合考虑轻量化、实时性和准确性，在仿真自动驾驶实验中有不错的表现。

1. 介绍

基于视觉的自主避障在自动驾驶领域中占据着非常重要的地位，美国特斯拉公司甚至放弃激光雷达，尝试采用纯视觉的方式解决自动驾驶的难题。而其他的自动驾驶公司也普遍倾向于采用视觉与激光雷达的组合方案，来满足自动驾驶任务的要求。

由于自动驾驶的实时性和安全性的要求，一个具有实际意义的良好自主避障方法必须满足实时性、准确性和轻量化的要求。但是直接利用实时采样的驾驶画面进行端到端分类的运动命令预测，往往因为画面中大量的冗余信息导致预测准确率低，预测效果差的问题。

本文提出一种基于语义分割和分类网络的深度学习模型，先使用时效性较高的语义分割网络，处理原始的采样图片。将其中的冗余信息剔除，突出标注出关键数据，如：道路，车辆等障碍信息。语义分割网络的输出图片，将通过端到端的分类网络，输出及时预测的运动命令。运动命令空间选取前进、左转、右转、前进并左转、前进并右转、无操作六类，基本囊括载具前进的全部动作。端到端的分类网络输出预测的动作命令，最后通过模拟驾驶的方式，将命令作用到载具，实现沿道路的自动驾驶。

我们首先收集了人类玩家在《极品飞车 8》中的游戏数据，将路况画面和按键作为数据集，训练我们的分割网络。在保证分割网络具有较好的效果后，利用成熟的分割网络，生成更多的分类网络的数据集，训练分类网络以达到一个较好的预测水平。为了检验我们模型的性能，我们自主开发了适配《极品飞车 8》的接口，程序实时截取路况图片，经过整个网络的处理，将输出的运动命令通过模拟键盘的方式，作用到游戏中，模拟整个自动驾驶的过程。整体的大致架构可以由图 1 表示：

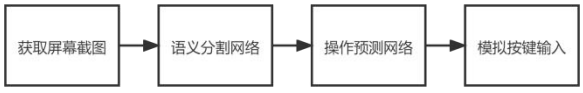


图 1 整体模型结构

2. 相关工作

自从 2012 年 Alexnet 问世以来，深度神经网络在计算机图像处理方面大放异彩。图像处理的基本任务，如分类，检测和分割等等，在近年来都得到了长足的发展，并且在未来还会更进一步。就本课题而言，我们主要调研了分类和分割任务。

对于分类任务，近年来涌现出很多经典的网络结构，如VGGNet[1]，GoogLenet（或称 Inception）[2]，Resnet[3]和Densenet[4]等，以及它们的变体；这些网络结构在分类任务上取得了非常好的结果，同时页经常称为其他任务（如检测和分割）的backbone。VGGNet一直使用 3×3 的卷积核，这样可以使网络结构简单，而且能够减少参数。

GoogLenet 的结构是相较于之前的卷积神经网络有很大不同，其采用多种卷积核（包括池化）并行的运算操作，然后将结果拼接起来，这种方法既能够保持网络结构的稀疏性，同时也能够利用密集矩阵的高计算性能。相较于 GoogLenet 的结构，Resnet则是直接通过残差结构完成主分支和侧面分支的连接，这种简单的方式却能够高效地堆叠网络，能够将网络的层数堆得很深，解决之前网络过深导致的损失上升，正确率下降的问题。Densenet 则是在 Resnet 的基础上增加了更多的连接，使各个特征图之间的连接更加稠密，从而达到更好的分类效果。

对于分割任务，我们选择了使用实时语义分割的方法对输入图片进行处理，因为我们的任务只是为了识别道路和避开障碍，所以不需要使用实例分割。语义分割在理解图像内容和寻找目标图像方面具有重要的意义，这种方法也理所当然地应用到了自动驾驶上，作为载具采取决策的重要依据。与此同时，为了实现我们的目标——达到实时分割的效果，我们必须要选择轻量级的分割网络，在效果和速度上做出合理的权衡。

为了实现实时语义分割，我们不能使用像FCN[5]那样模型很大，运算次数很多的网络。SegNet[6]引入了 encoder − decoder 结构，但是仍然不能够满足实时分割的要求（其运行的速率大约为 2.5 fps）。最终我们选择了使用 ENet 这个轻量级实时语义分割网络。它是在 FCN和 SegNet的基础上改进的网络，其结构相对简单，而且能够在算力有限的GPU上运行出不错的实时分割效果。

3. 实验方法

我们使用 Pytorch 搭建整体的网络结构并进行相应的实验。在 3.1 节中，我们将详细介绍用于语义分割的分割网络ENet的参数和调用设计。在 3.2 节中，我们将详细介绍用于预测运动命令的分类网络。在 3.3 节中，我们将介绍用于检测模型的仿真平台上的相关工作，由于缺少专业的自动驾驶仿真平台，我们采用《极品飞车 8》作为我们的仿真平台，将游戏画面处理之后输入模型，将模型输出结果通过模拟按键作用到游戏中，仿真整个自动驾驶的过程。3.4 节，我们将介绍整个训练过程中用到的技巧，用于加速运行速度和提高准确率的一些方法。

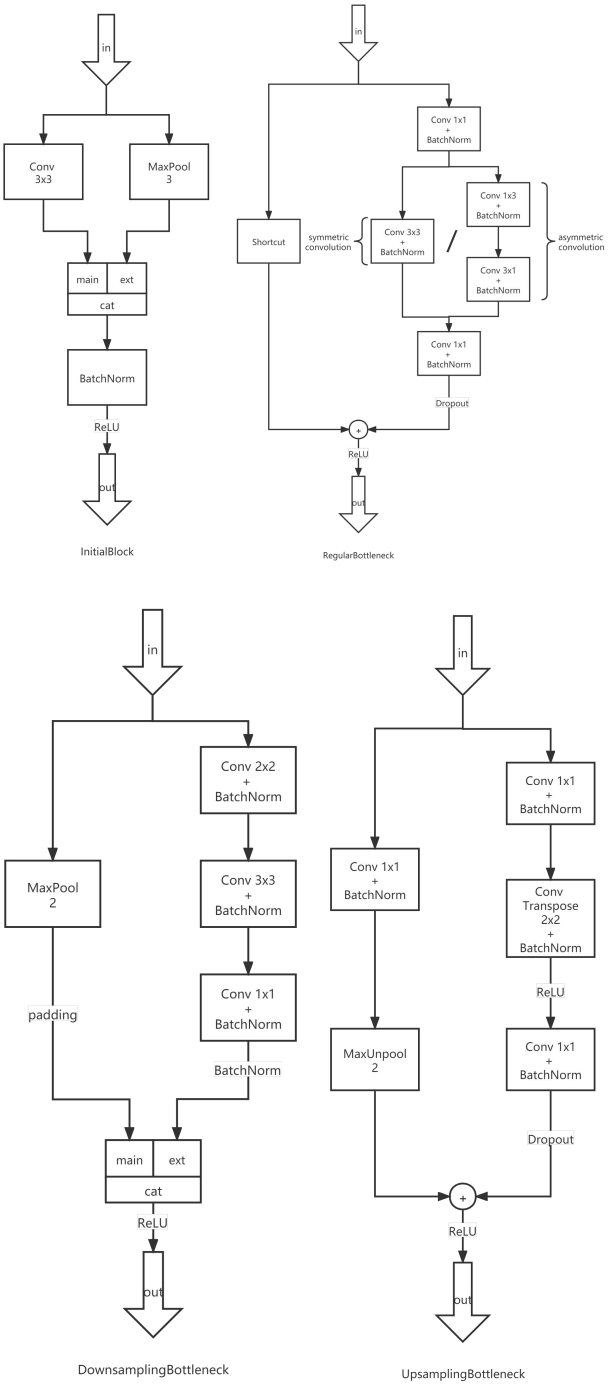
3.1. 语义分割网络：$ENet$

如上述所说，我们使用 ENet 的网络结构处理语义分割。

ENet 是一个实时的全卷积语义分割网络，其网络结构具有 4 个关键部分：首先是初始化模块，通过步长为 2 的卷积和池化分别将图片的长宽分割成原图的一半，然后将得到的特征图进行拼接。这种方式称为“提前降采样”，因为原作者认为直接处理大图片的代价十分昂贵。在此之后，作者采用了不对称的网络结构：4 层的编码器和 2 层的解码器，因为其认为编码器更加值得重视。编码器和解码器都采用了 resnet [3]的结构，编码器由下采样模块和常规模块构成，解码器由上采样模块和常规模块构成。如果是常规的卷积部分，则主分支不会有任何处理，直接叠加到经过卷积的侧面分支上。如果是上采样或者是下采样模块，则主分支会进行最大池化或者反池化以保证维度和侧面分支的相同；如果通道数和旁路不符合，那么还要对主分支进行 padding，然后按元素相加。

除此之外，在网络的训练过程中，同时还使用 SpatialDropout [9]和Batc*h*Normalization[8]等方法以防止过拟合。作者通过实验表明，在下采样时采用 PReLU [7] 作为激活函数，上采样时则使用 ReLU 作为激活函数能够取得更好的效果。

ENet作为一个实时且轻量级的语义分割网络，为了减少参数和运算次数，所有的卷积都不使用偏置。在旁路的卷积操作都是先通过 1×1 卷积来降低通道数，在选择采用空洞卷积以增加感受野，然后再通过 1×1 的卷积卷回正常的通道数。在高语义的特征图部分，还采用非对称卷积[10]进一步减少参数，增加网络的运算速度。图 2 展示了 ENet 上采样和下采样的网络结构。图 3 展示了ENet的模块结构和整体的网络架构。



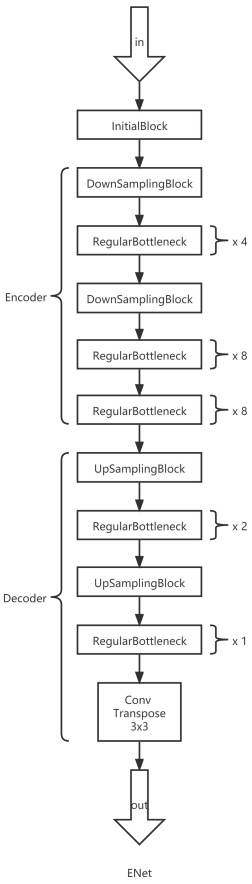


图 2 Enet网络结构

3.2. 分类网络：改进 VGG

在图像分割完成后，得到的特征图是一维的类灰度图，但是尺寸和分割前的图片没有变化。为了预测输出的操作，我们进一步采用了 VGG 的网络结构进行完成按键的分类任务。VGG 网络的最主要的特征就是使用很多的 3×3 的卷积核进行处理，这样的网络结构简单清晰。而且由于前一步的分割网络相当于对图片特征的提取，所以这里我们也不打算采用复杂的分类网络结构或者额外的数据增强。

除此之外，我们对于 VGG 的结构进行了改进，以此更好的切合我们的要求。首先尽管输入的图片是 480×640 的矩形，我们还是直接进行了卷积而不将图片 resize 或 padding 成正方形。我们最初采用通道数一直增加的卷积然后进行全连接，尽管卷积和池化的步长设置的很大以此快速降低维度，但是最终由于通道数为 1024，导致全连接层的节点非常多，模型的尺寸非常大，约为 29 M。后来为了进一步压缩网络模型的大小和其进行推理的时间，保证更好的时效性，我们将卷积的通道数先增加后减小，全连接层前的通道数为 32，这样模型的尺寸变为了接近 3 M，直接缩小了 10 倍。图 3 展示我们使用的分类网络结构。

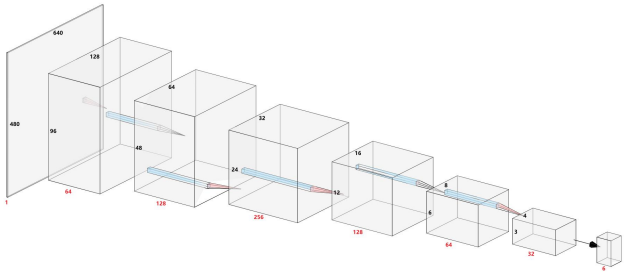


图 3 分类网络结构

3.3. 仿真实验接口

由于缺少成熟的仿真平台接口，我们决定利用《极品飞车 8》作为自动驾驶的仿真平台，并基于此自主开发了对接接口。主要包括了路况画面的实时获取和运动指令的模拟实现。

路况画面获取我们采用实时截取游戏画面的方式实现。截屏通过调研，最后决定使用pyautogui库中的接口，返回较为方便操作的PIL Image对象，同时花费时间较短，约为 8ms。可以满足模型实时性的要求。

运动指令的模拟实现我们采用模拟按键的方式输给游戏对象。按键模拟起初只是简单找了pyuserinput库进行按键的模拟，后来发现实际对游戏并不起作用，检索信息后发现，对于一些安全性较好的全屏游戏，游戏不会接收SendInput方式的通过函数发送的键盘信号，只会接收DirectInput方式的触发键符对应硬件编码产生的信号，所幸pywinio库能够实现键盘的DirectInput功能，最终成功触发键盘事件，在游戏中实现自动化按键。

3.4. 其他设计

为了提高AI操作的精细程度，以及提高分类网络的容错能力，需要保证网络以及模拟按键的处理速度至少保持在 0.1s之内，而在测试的过程中，我们发现如果将网络放在非主线程，则网络的处理速度会大大减慢。故我们采用将网络处理放在主线程，而单独增加了一个新的线程，用于模拟按键。最终，在测试平台《极品飞车 8》上，我们在显卡为 GTX1650 的笔记本上的运行速度为 20fps，超过人类极限反应速度。

由于无操作的运动命令实际上取决于驾驶员的主观性较多，所以在测试的过程中，载具大部分不会选择采取无操作的运动命令，这就导致了其会因为前进命令的累加导致速度过快。为了改善这个问题，我们受 dropout 层的启发，在最终模拟按键部分增加了前进命令的 p 执行策略，即以预先设定的 p 概率执行此次的前进命令，否则无操作，利用摩擦力控制载具速度，这样能够防止因为车速过快而导致的时空。采用这种方法的重要性还体现在本次项目中无法使用 PID，所以这种方法能够协助我们对车辆进行更有效的控制。

4. 数据集

由于需要适配仿真平台，而且并没有《极品飞车 8》的相关数据集，我们自己制作了数据集，通过获取一些水平较高玩家的对局视频，将其以固定频率采样得到的图片有序输入进模型中，将预测运动命令与实际玩家的运动命令做监督学习，训练模型对于各个情况下运动命令预测的准确性。

分割网络的数据集，我们获取人类高水平玩家的对局视频，并主动挑选了一些有代表性的游戏截屏。使用 labelme 手动标注了 528 张分割效果图片。其中优先选取车辆和道路并存的画面帧进行标注。为了提高网络的分类效果，我们采取多边形的标注策略，即用多边形标注出路面和车辆，并且在交集处做到相切、不重合。为了提高载具行驶时的规范性，我们没有将左右空旷的人行道区域划分为路面（但实际上在游戏中载具是可以通过的），这样也预留了一定的空间，提高了整体的鲁棒性。

分类网络的数据集的制作流程可分为以下步骤：首先对游戏过程进行截图并记录当前玩家的操作作为图片名称。然后经过人工预处理，对数据进行清洗，剔除玩家失误的片段。接着使用训练好的分割网络生成分割结果，保留玩家操作的标签，做监督学习。最终保留的分类网络的数据集数量为 7373 张分割结果及标签，尽量保持各个运动命令出现频率的一致。

为了能够尽量找出在测试集上效果最好的模型，我们将数据集都分为训练集、测试集和验证集。通过验证集的指标，预测其在测试集上的指标好坏，从而尽可能地保留最好的模型参数。

5. 实验过程和结果

ENet 训练过程在 google colab 上完成，使用了 Adam 优化器，batch size 为 8，训练了 300 个 epoch，初始学习率为 5×10^{-4}，每 100 个 epoch 时间学习率减小 10 倍。每 10 次训练后进行验证，每次保留验证正确率最高的模型，最终保留的模型是训练 280 次时的模型。

在训练过程中，我们使用了自己通过 labelme 标注的数据集。我们选择仅仅将图片分成 3 类，即背景，道路和车，因为针对本次任务，主要起作用的就是车和路两类。由于语义分割训练的数据较少，总共只有 528 个样本，所以这里采用了额外的数据增强进行训练。原作者的分割网络没有使用数据增强，仅仅只有一个归一化。但是当我们也不采用数据增强时，最终在测试集上模型的表现不佳。所以在思考之后，我们使用了较多的数据增强，如随机上下或左右翻转，随机修改图片的亮度，饱和度等属性，随机旋转，随机裁剪等。在测试集上得到如下的结果，可见增加了数据增强后预测的效果显著提升。表 1 展示有无数据增强的对比结果。

	无数据增强	数据增强
测试$mIoU(\%)$	73	89.8

表 1 有无数据增强结果对比

分割部分结果展示如下，可见效果还是较好的。

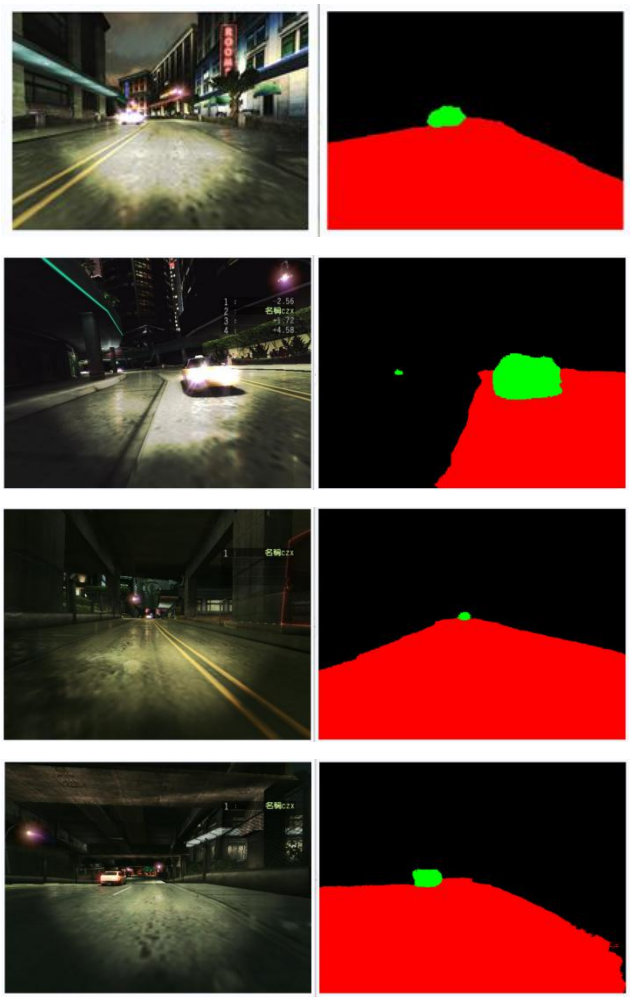


图 4 分割展示结果

分类网络则主要负责将分割图片映射到分类网络的预测空间中去。分类网络的预测空间具体分布关系为：{'A': 0, 'D': 1, 'N': 2, 'W': 3, 'WA': 4, 'WD': 5}。我们利用分割网络分割出的结果进行处理，最终分类测试集正确率 0.7206。但其实在某些画面中，玩家可能做出的”W”动作和“WA”或“WD”动作都是合法的，并不是仅有唯一解。这在后续的仿真实验中也证实了，实际的测试运行游戏时具有一定的容错能力，不会因为预测的键稍有不同就导致整体出差错，车辆完全失控。我们认为这里不能仅仅依靠正确率

进行评判，主要体现模型性能的不是正确率，而是最终运行的结果。

6. 总结

本文提出了一种基于纯视觉的模拟自动驾驶应用，根据实时获取的路况画面，经过语义分割网络提取图片中的路和车等特征，然后通过分类网络处理预测出下一步的运动命令，调整载具的运动姿态。模型在实时性和准确性都达到了一定的标准，并且在《极品飞车 8》游戏中的自动驾驶仿真中可以有一些不俗的表现。未来，一方面我们将扩大数据集，通过更大量的数据集的训练提高整体网络的性能。另一方面，我们将结合速度、加速度等信息，综合视觉和控制技术，实现更完备的自动驾驶应用架构。

7. 致谢

感谢华为公司提供的免费华为云计算服务。

8. 参考文献

[1] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[2] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1-9.

[3] 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.

[4] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 4700-4708.

[5] 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.

[6] Badrinarayanan V, Kendall A, Cipolla R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(12): 2481-2495.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," pp. 1026–1034, 2015.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[9] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 648–656.

[10] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818-2826.