arXiv:2302.06572v1 [eess.SY] 13 Feb 2023

EnergyShield：面向能效的神经网络控制器可证明安全的卸载

Mohanad Odema、James Ferlez、Goli Vaisi、Yasser Shoukry、Mohammad Abdullah Al Faruque 加州大学欧文分校电气工程与计算机科学系，美国加利福尼亚州欧文

摘要

为缓解基于神经网络（NN）的自动驾驶系统（ADSs）对能量的高需求，我们研究了将神经网络控制器从自动驾驶系统卸载到附近边缘计算基础设施的问题，同时确保形式化的车辆安全属性。具体而言，我们提出了 EnergyShield 框架，将控制器“防护机制”重新用于自动驾驶系统车辆的低功耗运行时安全监视器。具体来说，EnergyShield 中的防护机制不仅提供安全干预，还对车辆安全受到威胁之前可容忍的边缘端响应时间进行基于状态的形式化量化。借助 EnergyShield，自动驾驶系统可以通过无线方式将神经网络计算卸载到边缘计算机，从而节省能量，同时在收到响应之前仍维持形式化的安全保证（车载硬件提供即时故障安全机制）。为验证 EnergyShield 的优势，我们在 Carla 仿真环境中实现并测试了该框架。结果表明，与车载神经网络评估相比，EnergyShield 在确保车辆安全运行的同时实现了显著节能：在各种无线条件和边缘延迟下，能耗降低了 24% 至 54%。

关键词

形式化方法，车载，边缘计算，自动驾驶车辆，可证明安全性，卸载，自动驾驶系统

1 引言

神经网络（NN），尤其是深度神经网络（DNN）的理论与应用不断进步，推动了多项人工智能任务取得革命性进展，包括感知、运动规划和控制。因此，深度神经网络为取代以往只能由人类完成的任务提供了可行的工程解决方案，其中最具雄心的应用当属自动驾驶系统。然而，最先进的自动驾驶系统需要采用规模非常庞大的深度神经网络架构来完成关键的感知与控制任务，而这些任务通常涉及处理数十个摄像头、激光雷达及其他传感器的输出。因此，当代自动驾驶系统只有在车辆本身部署大量计算资源的情况下才得以实现，因为其深度神经网络必须在闭环中实时处理这类高带宽传感器数据，

本研究部分得到了美国国家科学基金会（NSF）通过 CCF-2140154、CNS-2002405 和 ECCS-2139781 项目资助，以及 C3.ai 数字化转型研究所的支持。

允许免费为个人或课堂使用而制作本作品的部分或全部内容的数字或纸质副本，但前提是这些副本不得用于营利或商业目的，且不得为此目的而制作或分发，并且副本首页须附有本声明和完整引文。本作品中第三方组件的版权必须予以尊重。其他所有用途，请联系版权所有者/作者。*ICCPS '23，2023年5月9日至12日*，德克萨斯州圣安东尼奥© 2023 版权所有者/作者保留版权。ACM ISBN 978-x-xxxx-xxxx-x/YY/MM。https://doi.org/10.1145/nnnnnnn.nnnnnnn

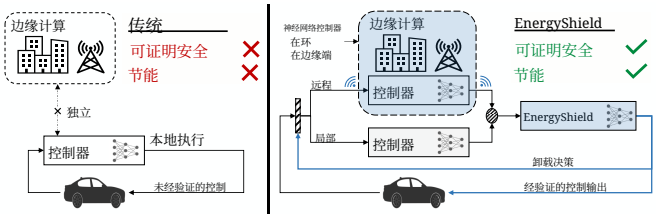

图 1：能效导向的神经网络控制器可证明安全卸载示意图。

实时高性能车载计算对能耗的实际影响尚未得到充分研究，但现有研究表明其影响十分显著：例如，车辆续航里程最多可减少 15% [22, 27]。

与此同时，半导体设计和封装技术的进步使低成本、低功耗的硅芯片成为可能；无线网络技术的进步则让高带宽、低延迟的无线链路即使在复杂的多用户环境中也成为可能。这些进步共同促成了日益普及、成本低廉且可通过无线方式访问的计算资源，使其遍布传统有线基础设施的边缘端 附近。尤其是，如今已能在联网自动驾驶系统与附近的边缘计算设施之间实现可靠的毫秒级延迟无线连接 [3, 24, 26]。

因此，边缘计算的普及为降低自动驾驶系统车辆的局部能耗提供了一种自然而然的方法：通过无线方式将繁重的感知和控制深度神经网络计算卸载 到附近充足的边缘计算基础设施中。然而，即使是现代无线网络和适合卸载的深度神经网络架构（例如编码器/解码器），也无法提供形式化保证，证明让边缘计算“进入控制回路”后，其性能能够与车载硬件相当（甚至达到可接受的水平）。在人命攸关时，这种情况不可接受：即使是在获取控制动作或感知分类结果时相对罕见且短暂的延迟，也可能造成致命后果。

本文提出了 EnergyShield 框架，作为一种以形式化且可证明安全的方式对自动驾驶系统控制器执行深度神经网络到边缘的卸载的机制。因此，据我们所知，EnergyShield 是首个能够通过以可证明安全的方式将大型深度神经网络的计算智能地卸载至边缘计算，从而显著降低车载能量消耗的框架；见图 1。EnergyShield 的核心思想是在始终具备足够车载计算能力作为安全备用方案的前提下，将深度神经网络计算以安全性感知的（状态）上下文方式卸载至边缘端。该过程通过使用控制器“防护机制”实现，该机制既用于强制实施安全性，又作为一种新型低功耗运行时安全监视器。具体而言，这种基于防护机制的安全监视器能够提供可证明安全的边缘计算响应时间：也就是说，在每个时刻，EnergyShield 都会提供一个时间区间，在此区间内边缘计算必须响应卸载请求，才能保证车辆在此期间的安全。

如果边缘资源未能在该时间内提供响应，车载本地计算将继续评估相关深度神经网络，以免车辆安全不再得到保障。通过加入估计器来预测边缘计算负载和无线网络吞吐量，还可以进一步节省能量；将该估计值与运行时安全监视器提供的可容忍边缘计算延迟进行比较，从而作出更智能的卸载决策，避免执行可能失败的卸载。

EnergyShield 的主要技术创新在于上述基于屏蔽器的运行时安全监视器。尽管控制器"安全屏蔽"是一种众所周知的、用于确保通用控制器安全的方法，但 EnergyShield 的安全屏蔽方面本身包含两项重要的新贡献：第一，不仅使用屏蔽器来强制实施安全性，还将其作为运行时安全监视器，以量化系统距离不安全状态的剩余时间；第二，针对实现复杂度和能量因素，对该运行时监视器进行了专门设计。对于第一项贡献，EnergyShield 扩展了现有方法，即使用（零化）障碍函数（ZBF）的当前值作为运行时监视器，以量化智能体的安全性：具体而言，EnergyShield 的创新之处在于，转而利用 ZBF 的当前值推导出智能体变得不安全之前时间的可靠量化结果。此外，EnergyShield 以极高的能效实现了这种量化：即通过一个小型查找表，仅需少量浮点运算次数即可获得有保证的"距离不安全状态的时间"。运行时安全监视器的这一特性还得益于 EnergyShield 中所采用的一个特定但已知的 ZBF 和屏蔽器 [15]：这些组件都极其简单，因此可以使用小型、节能的神经网络来实现 [15]。这些设计选择共同确保，通过卸载节省的任何能量都不会随后消耗在 EnergyShield 自身的实现上。

我们通过一系列重要的实验结束本文，以验证 EnergyShield 框架所提供的安全性和能量节省效果。具体而言，我们在 Carla 仿真环境 [11] 中使用多个强化学习学习（RL）训练的智能体。我们的实验表明，EnergyShield 完全消除了所考察的强化学习智能体与障碍物发生碰撞的情况——也就是说，使它们变得安全——同时将神经网络能耗最多降低了 54%。此外，我们还表明，EnergyShield 具有直观且注重安全的卸载行为：当自动驾驶系统靠近障碍物、因而安全性较低时，EnergyShield 的运行时安全监视器会有效地强制仅在车载设备上执行神经网络推理；当自动驾驶系统离障碍物较远、因而更安全时，EnergyShield 的运行时安全监视器则允许更多卸载，从而实现更多节能。**相关工作：** 数据训练控制器的形式化方法。 目前已有多种方法，可通过形式化保证来验证数据训练控制器的安全性；其中大多数方法会以某种方式运用控制理论的思想，对训练好的控制器进行增强，以实现这一目标。这些方法的优秀综述见 [10]。此类方法的例子包括李雅普诺夫方法 [4, 7, 8]、安全模型预测控制 [20]、可达性分析 [1, 16, 17]、屏障证书 [25, 29, 32, 35, 36, 38]，以及不确定性的在线学习 [31]。控制器"安全屏蔽" [2] 是另一种技术，通常属于障碍函数这一类 [6]。另一种方法则尝试使用形式化验证技术（例如模型检验）来验证学习型控制器的形式化安全性质：例如，

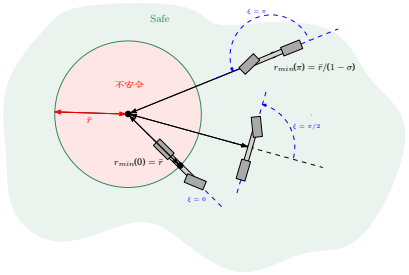

图 2：障碍物规范和最小屏障函数距离随车辆相对朝向的变化关系， ξ。

通过使用类 SMT 求解器 [12, 23, 33] 或混合系统验证 [13, 19, 37]。然而，这些技术只能评估给定控制器的安全性，而无法设计或训练安全的智能体。

自主系统边缘计算。 针对自动驾驶系统，已有许多不同的边缘/云卸载方案被提出，但迄今为止没有任何方案提供形式化保证。其中一些方案侧重于调度技术和网络拓扑，以实现高效卸载 [9, 14, 30, 34, 40]。另一些方案则侧重于拆分以及其他神经网络架构，使卸载更加高效 [5, 26, 28]。

2 预备知识

2.1 符号说明

令 $\mathbb{R}$ 表示实数；$\mathbb{R}^+$ 表示非负实数；$\mathbb{N}$ 表示自然数；$\mathbb{Z}$ 表示整数。对于连续时间信号 $x(t)$ $t \geq 0$，将其离散时间采样版本记为 $x[n]$，其中采样周期 T（单位：秒）为固定值；即令 $x[n] \triangleq x(n \cdot T)$，其中 $n \in \mathbb{Z}$。令 $\mathbf{1}_a \colon \mathbb{R} \to \{a\}$ 表示取值为 a 的常值函数；即对所有 $x \in \mathbb{R}$（必要时将其解释为序列），均有 $\mathbf{1}_a(x) = a$。最后，令 $\dot{x} = f(x,u)$ 表示一个一般的控制系统，其具有 $x \in \mathbb{R}^n$ 和 $u \in \mathbb{R}^m$；令 $\pi \colon \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^m$ 表示一个可能时变的控制器。对于该系统和控制器，考虑时刻 $\tau \geq 0$ 和状态 x_0，并将该系统在控制器 π 作用下、假设 $x(t_0) = x_0$ 时的 t_0 移位状态演化记为 $\zeta^{t_0,\pi}_{x_0} \colon \mathbb{R}^+ \to \mathbb{R}^n$。令 $\zeta^{n_0,\pi}_{x_0}[n]$ 表示相同的量，但采用离散时间，并对 π 进行零阶保持施加。$\|\|$ 和 $\|\|_2$ 分别表示 $\mathbb{R}^n$ 上的最大范数和二范数。

2.2 运动学自行车模型

在本文中，我们将使用运动学自行车模型（KBM）作为自动驾驶车辆 [21] 的形式化动力学模型。不过，我们考虑的是相对于平面上某个固定点定义的状态变量的 KBM模型，而非绝对笛卡尔坐标。也就是说，状态是车辆相对于该固定点的距离 $\|\vec{r}\|$ 和朝向角 ξ。这些量描述了一个状态空间模型：

$$\begin{pmatrix} \dot{r} \\ \dot{\xi} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} v\cos(\xi-\beta) \\ -\frac{1}{r}v\sin(\xi-\beta) - \frac{v}{\ell_r}\sin(\beta) \\ a \end{pmatrix}; \quad \beta \triangleq \tan^{-1}\left(\frac{\ell_r}{\ell_f+\ell_r}\tan(\delta_f)\right) \quad (1)$$

其中 $r(t)$是到原点的距离；a是线加速度输入；δ_f是前轮转向角输入；而 $\psi + \xi = \tan^{-1}(y/x)$。为方便起见，我们指出几个特殊状态：在 $\xi = \pm\pi/2$处，

也就是说，转向角可以瞬时设定，因此可以忽略转向齿条动力学。

车辆与障碍物相切，在 $\xi = \pi$ 或 0 处，车辆分别直接朝向或背向原点（图 2）。β 是转向控制 δ_f 的一个可逆函数。

我们还假设 KBM 具有转向约束，使得 $\delta_f \in [-\delta_f \max\ \delta_f \max]$。为简单起见，我们将直接把 β 视为控制变量；鉴于上文所述原因，这不会损失一般性。因此，β 也受到约束：$\beta \in [-\beta \max\ \beta \max]$。最后，我们将 KBM 的状态向量和控制向量定义为：$\chi \triangleq (\xi,\ r,\ v)$ 和 $\omega \triangleq (a,\ \beta)$，其中 $\omega \in \Omega$ 允许控制。$\triangleq \mathbb{R} \times [-\beta \max\ \beta \max]$，为允许控制集合。因此，KBM 的动力学由 $\dot{\chi} = f$ KBM$(\chi,\ \omega)$ 给出，其中 fKBM 根据（1）定义。

2.3 屏障函数与安全屏蔽

接下来，我们将使用控制器“屏蔽器”，这是一种可即时修正闭环中控制器输出的方法；其目标是进行修正，使原控制器无论如何设计或实现，都能变得安全——这正是“屏蔽器”这一名称的由来。具体而言，控制器屏蔽器围绕状态空间中一个实值函数进行设计，该函数称为（零化）屏障函数（ZBF）。ZBF 通过其符号直接指定一组安全状态：ZBF 非负的状态被视为安全。随后，ZBF 以间接方式（作为状态的函数）指定安全控制，使得沿动力学轨迹，ZBF 的符号保持不变。

形式上，考虑一个与状态反馈控制器 $\pi\colon\ x \mapsto u$ 构成闭环的控制系统 $\dot{x} = f(x,u)$。在这种情况下，闭环中的反馈控制器会将控制系统转化为自治系统——自治向量场 $f(\cdot\ \pi(\cdot))$。因此，在自治系统的背景下，我们回顾以下零化屏障函数（ZBF）的定义：

定义 1（零化屏障函数（ZBF） [39, 定义 2]）。令 $\dot{x} = f(x, \pi(x))$ 为上述闭环自治系统，其中 $x(t) \in \mathbb{R}^n$。此外，令 $h\colon\ \mathbb{R}^n \to \mathbb{R}$，并定义 $C \triangleq \{x \in \mathbb{R}^n\colon\ h(x) \geq 0\}$。若存在一个局部利普希茨的扩展类$\mathcal{K}$函数，$\alpha$ 使得：

$$\nabla_x h(x) \cdot f(x, \pi(x)) \geq -\alpha(h(x)) \text{ for all } x \in C \tag{2}$$

那么 h 称为零化障碍函数（***ZBF***）。

此外，上述障碍函数的条件可以转化为关于此类反馈控制器输出的集合成员问题。以下推论对此作出解释。

命题 1。令 $\dot{x} = f(x,u)$ 为一个控制系统，在集合 $\mathcal{D} \times \Omega admis.$ 上关于两个自变量均满足利普希茨连续；此外，令 $h\colon\ \mathbb{R}^n \to \mathbb{R}$，其中 $C_h \triangleq \{x \in \mathbb{R}^n | h(x) \geq 0\} \subseteq \mathcal{D}$，并令 α 为一个$\mathcal{K}$类函数。若集合

$$R_{h,\alpha}(x) \triangleq \{u \in \Omega_{admis.} | \nabla_x^T h(x) \cdot f(x,u) + \alpha(h(x)) \geq 0\} \tag{3}$$

对每个 $x \in \mathcal{D}$都非空，且利普希茨连续的反馈控制器 $\pi\colon\ x \mapsto u$ 满足

$$\pi(x) \in R_{h,\alpha}(x) \quad \forall x \in \mathcal{D} \tag{4}$$

则C_h对于闭环动力学 $f(\cdot\ \pi(\cdot))$是正向不变的。特别地，若 π 4 x t $\dot{x}$ =满足()，且()是(、())的一 f x π x h x 0 ≥0 h x t ≥0 $t \geq 0$ 条轨迹，其中(())，那么对所有，都有(())。

证明。由零化屏障函数 [39, 定理 1]的应用可直接得出。□

命题 1 是控制器屏蔽的基础：(3) 和 (4) 确立了 h（以及相关的 α）是闭环自治动力学 $f(\cdot\ \pi(\cdot))$ 的零化屏障函数。还要注意，无需区分闭环反馈控制器 π 与由 π 和一个基于当前状态对其输出进行安全屏蔽（或过滤）的函数构成的复合体。因此，定义如下：

定义 2（控制器屏蔽器）。令 $\dot{x} = f(x、\ u)$、h、C_h、α和$\mathcal{D} \times \Omega admis.$ 如命题 *1* 所述。则控制器屏蔽器是一个利普希茨连续函数 $\mathfrak{S}\colon \mathcal{D} \times \Omega admis. \to \Omega admis.$，满足

$$\forall (x,u) \in \mathcal{D} \times \Omega_{admis.}.\mathfrak{S}(x,u) \in R_{h,\alpha}(x). \tag{5}$$

2.4 KBM 的控制器屏蔽器

本文将使用已有的、为 KBM 设计的零化屏障函数和控制器屏蔽器 [15]。二者协同工作，为图 2 中所示的安全属性提供控制器安全屏蔽：即防止 KBM 进入以原点为中心、半径为 $\bar{r}$ 的圆盘。

具体而言，[15]为 KBM 提出了以下一类候选障碍函数：

$$h_{\bar{r},\sigma}(\chi) = h_{\bar{r},\sigma}(\xi, r, v) = \frac{\sigma\cos(\xi/2)+1-\sigma}{\bar{r}} - \frac{1}{r} \tag{6}$$

其中，$\sigma \in (0,\ 1)$ 是用于表征该类别的参数；此选择与一个 K 类函数结合：

$$\alpha_{v_{\max}}(x) = K \cdot v_{\max} \cdot x \tag{7}$$

从而形成一类参数化的零化屏障函数（ZBF）。还要注意，这类 ZBF 忽略了状态变量 v；[15] 中的一项结果表明，只要车辆速度保持（受控）在范围 $(0,\ v\max]$ 内，这类函数就可用作障碍函数。还要注意，方程 $h_{\bar{r},\ \sigma}(\chi) = 0$ 有一个便于求解的解，我们将其记为 $r\min$，以供后续参考：

$$r_{\min}(\xi) = \bar{r}/(\sigma\cos(\xi/2) + 1 - \sigma). \tag{8}$$

[15] 中的一项主要结果是提供了一种机制，可根据 KBM 参数（例如 ℓ_r）和安全参数 $\bar{r}$ 来选择参数 σ，从而确保得到的具体函数确实是所需的 ZBF。

最后，我们指出，[15] 还提出了一种基于 (6) 的极轻量级障碍函数实现方案。也就是说，它包含一个“屏障合成器”，通过 ReLU 神经网络 [15, pp 6] 对一个简单的单输入单输出凹函数进行近似，从而实现控制器屏蔽器。这种构造在后文中也会证明很有优势。我们将由此得到的控制器屏蔽器记为 $\mathfrak{S}$KBM，其相关的障碍函数、KBM 参数和安全参数均根据上下文推断得出。

3 框架

注意：在本节中，我们分别用 x、y 和 u 表示自动驾驶系统的状态、传感器和控制变量；这种抽象符号将展示 *EnergyShield* 框架，而不涉及具体的建模细节。关于 *EnergyShield* 的详细讨论，包括形式化证明和假设，见第 *4* 节。

3.1 EnergyShield 动机与…… text

EnergyShield 框架的基本动机如下。假设某个自动驾驶系统包含一个大型神经网络 $\mathcal{NN}_c$，负责根据传感器

信号 y生成控制动作 $\mathcal{NN}_c$。进一步假设，由于神经网络规模庞大，使用车载硬件计算 $\mathcal{NN}_c$的输出会消耗大量能量。因此，从能量角度来看，将对 $\mathcal{NN}_c$ 的计算卸载 到边缘计算基础设施会更有利：换言之，将特定的传感器测量值 y无线传输至车外的边缘计算机，由其计算输出 $u = \mathcal{NN}_c(y)$ 并将结果返回给发起请求的自动驾驶系统。

这种方法的问题主要在于安全性。具体而言，控制器 $\mathcal{NN}_c$ 旨在实时且以闭环方式运行：也就是说，离散时间采样点 n处的控制动作应根据相同时间采样点的传感器测量值计算得出。用离散时间信号的符号说明（见第2.1节）表示，即：$u[n] = \mathcal{NN}_c(y[n])$。然而，将传感器测量值 $y[n]$卸载到边缘端意味着，$\mathcal{NN}_c(y[n])$的正确输出在经过某个非零数量的采样点（设为 Δ）之前无法返回车辆。因此，在时间 n，无法获得 $\mathcal{NN}_c(y[n])$来按预期设置 $u[n] = \mathcal{NN}_c(y[n])$；相反，最早能够使用输出 $\mathcal{NN}_c(y[n])$的时间将是 $n+\Delta$，即 $u[n+\Delta] = \mathcal{NN}_c(y[n])$。这种延迟显然会造成安全问题，因为车辆的状态——以及因此应采取的正确控制动作——将在这期间的 Δ 个时间采样点内发生变化。更重要的是，即使在 $n+\Delta$ 处施加"正确"的控制动作，也可能不足以确保安全：例如，经过 Δ 个采样点后，可能已经来不及采取充分的避让转向。

3.2 EnergyShield结构

如果我们假设 ADS 本身具备足够的车载计算能力，能够实时获得输出 $\mathcal{NN}_c(y[n])$，那么上述安全问题就转化为做出卸载决策：理想情况下，既能节省能量，又不损害安全性。也就是说，是否应将 $\mathcal{NN}_c$ 的某项计算卸载到边缘端？ADS 应等待响应多久，才能确保情况仍可纠正？

卸载决策的性质意味着，为确保卸载期间 ADS 车辆的安全，EnergyShield 必须处理两个相互交织的关键问题。一方面，EnergyShield 必须能够纠正 在做出卸载决策后由 $\mathcal{NN}_c$ 提供的控制动作（见上文说明）。另一方面，EnergyShield 必须限制对每项卸载请求的等待时长，以便首先确保可以 纠正 $\mathcal{NN}_c$ 提供的动作；也就是说，在所有可能的卸载延迟 Δ 中，并不能立即判断哪些可以纠正、哪些无法纠正（粗略地说，$\Delta = \infty$ 是一个很可能无法纠正的延迟）。从这个意义上说，确定某个响应延迟 Δ，是否可以纠正，实质上决定了如何做出卸载决策，因为这给出了安全性的失效时间：即开始卸载，并等待响应，直到经过 Δ，个采样，此时恢复对 $\mathcal{NN}_c$ 的本地计算。

具体而言，EnergyShield 包含两个核心组件，以实现这些目标：

C1：控制器屏蔽器。EnergyShield 包含一个控制器屏蔽器（见第 2.3 节），无论卸载延迟的控制器输出如何，都能确保安全性得到维护；换言之，它会纠正 $\mathcal{NN}_c$ 因以下情况而产生的不安全行为：

在卸载延迟期间，车辆状态发生了（对 $\mathcal{NN}_c$ 而言）未预期的变化。

C2：运行时安全监视器。EnergyShield 包含一个运行时安全监视器，为自动驾驶系统提供一个上限 Δmax（以采样为单位），说明为维持安全性，它等待其某个卸载请求响应的最长时间；假定此期间控制动作不会更新；也就是说，若卸载延迟为 $\Delta \le \Delta$max，则**C1**（即控制器屏蔽器）可保证在卸载延迟期间保持最后一次控制信号更新后安全恢复（必要时，**C1** 可使用车载计算）。换言之，Δmax 规定了 **C1** 使用车载计算所提供的安全保证的失效时间。

自然，**C1** 和**C2** 需要协同设计，因为它们的目标彼此影响。事实上，在 EnergyShield 的具体设计中，这些组件基于同一个零化屏障函数（ZBF，定义见第 2.3 节）进行设计：形式化细节请参见第 4 节。

遗憾的是，组件**C1** 和**C2** 都无法有效地处理控制器所用的相同原始传感器测量值 $y[n]$；尤其是，鉴于我们打算通过零化屏障函数和控制器屏蔽器来实现它们，这一点更是如此。具体而言，两者都需要一些（有限的）自动驾驶系统状态信息才能执行各自的任务。因此，EnergyShield 需要一个感知/估计器组件，为 **C1** 和**C2**提供状态信息。请注意，为了提供更大的灵活性，我们有意将此类估计器的设计排除在 EnergyShield 框架之外：具体而言，由于控制器 $\mathcal{NN}_c$ 本身可能包含一个估计器，我们希望也允许将估计任务卸载——前提是它在本地即时执行，并在通知**C1** 和 **C2** 之前完成（见第 3.3 节）。不过，EnergyShield 仍然需要此类估计器，因此我们将其纳入为一个组件：

C3：状态估计器。EnergyShield 要求向**C1** 和**C2**提供（最小化的）状态估计值作为输入。按照惯例，该估计器将是一个神经网络，记为 $\mathcal{NN}_p$： $y \mapsto x$。我们假设 $\mathcal{NN}_p$ 可由车载硬件在一个采样周期内计算得出。

C3 与**C1** 和**C2** 的接口使后两个组件能够感知（状态）上下文。也就是说，*EnergyShield* 会基于当前车辆状态进行上下文感知卸载决策。此外，需要注意的是，由于卸载期间会保持先前的控制动作 Δmax，**C2** 的输出除了依赖状态外，还依赖控制动作：也就是说，对于任意状态 $\hat{x}$ 以及卸载前刚刚施加的控制动作 u，**C2** 实际产生的输出为 $\Delta\max(\hat{x}, u)$。

EnergyShield 还有一个组件，但它纯粹是为了节能而设计，不影响安全性。至关重要的是，**C2**所提供的安全性具有已知的失效时间，即 $\Delta\max(x)$，这使得我们有机会在作出卸载决策时利用额外信息。具体而言，可以利用对预期边缘端响应时间 $\hat{}$ 的估计值 Δ，来放弃那些不太可能在安全期限 $\Delta\max(x)$ 失效之前完成的卸载。因此，EnergyShield 包含一个边缘响应时间估计器，以便预先跳过可能会失败的卸载：

C4：边缘响应时间估算器。EnergyShield 规定，应提供当前边缘响应时间 $\hat{\Delta}$ 的估计值，以便作出卸载决策。

在形式化分析中，我们将对一次采样的计算延迟进行建模。

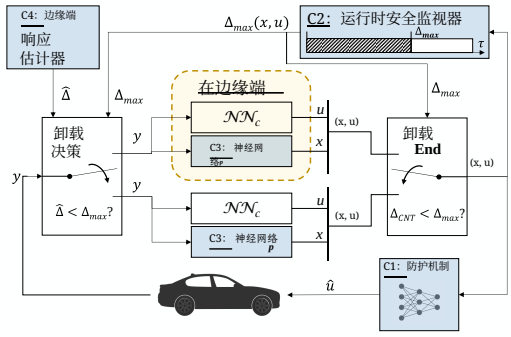

图3：EnergyShield 框架

不过，我们注意到，*EnergyShield* 并未规定该组件必须使用特定的估计器：许多不同的估计器都可能适用，并且每种估计器都可能因其有效性不同而产生不同的能耗曲线。此外，由于 $\hat{\Delta}$ 从未用于覆盖 $\Delta\max(x)$，因此无论使用何种具体估计器，安全性都能得到保留。

图3展示了 EnergyShield 中组件 **C1** 至**C4** 的互连关系。请注意，组件 **C3**（状态估计器）分别与组件 **C1** 和**C2**（控制器屏蔽器和运行时安全监视器）相连。此外请注意，**C2** 的输出向卸载决策开关提供信号 $\Delta\max(x)$；组件**C4**提供的即时边缘响应时间估计也会为该决策提供信息。

3.3 EnergyShield卸载决策的语义

在本小节中，我们考虑一个单独的假设性 EnergyShield 卸载决策的时间线，以说明第 3.2 节所述组件之间交互的细节。具体而言，假设一个卸载间隔刚刚完成，并且在时间索引 n_0 处需要做出新的卸载决策。

我们将卸载决策初始化与该卸载决策得到解决之间的时间称为一个卸载周期 （解决方式可能是收到来自边缘端的响应，或故障转移至车载计算）。

时间线：

[$n_0 - 1$] 上一个卸载周期的最后时间索引。

- 假设：$\hat{x}[n_0] \triangleq x[n_0-1] = \mathcal{NN}_p(y[n_0-1])$ 始终在上一个卸载周期的最后一次采样中由局部计算。[n_0] 新卸载周期的初始时间索引。
  - 新卸载周期的第一个采样继承自上一个卸载周期中局部计算的 $\hat{x}[n_0]$。
  - $\hat{x}[n_0]$ 被提供给**C1**，用于校正由上一次卸载计算得到的 $u[n_0]$。将此校正后的控制动作记为 $\hat{u}[n_0]$。在 n_0 处应用的控制动作：即 u_0-h $\hat{u}[n_0]$，其中 $\hat{u}[n_0]$ 如上所计算。
  - $\hat{x}[n_0]$ 和 u_0-h被提供给**C2**（运行时安全监视器），以生成 $\Delta\max(\hat{x}[n_0], u_0\text{-h})$
  - **C4**根据目前为止交换的所有数据包，生成边缘响应时间的估计值 $\hat{\Delta}$。
- 卸载决策：
  – 满足以下条件时继续卸载： $\hat{\Delta} \le \Delta\max(\hat{x}[n_0], u_0\text{-h})$ 且 $\Delta\max(\hat{x}[n_0], u_0\text{-h}) \ge 1$；即继续向 $y[n_0]$ 边缘端传输。初始化卸载时长计数器： $\Delta\text{cnt} = 1$ – 否则，终止卸载周期，并使用本地故障安全机制。跳转至带有 $\Delta\max(\hat{x}[n_0], u_0\text{-h})=1$ 的卸载失败。

[⋮]

[$n_0+\Delta_{\text{cnt}}$] *Offload in progress; no edge response and* $\Delta_{cnt} < \Delta_{max}(\hat{x}[n_0])$

- Maintain zero-order hold of $u[n_0 + \Delta_{\text{cnt}}] = u_0\text{-h}$.
- Increment Δ_{cnt}: $\Delta_{\text{cnt}} \leftarrow \Delta_{\text{cnt}} + 1$.

[⋮]

现在，当前卸载周期将以以下两种方式之一结束：*I)* 卸载成功： （从 $n_0 + \Delta cnt$ 恢复时间线）

[⋮]

[$n_0 + \Delta$] 收到边缘端响应； $\Delta = \Delta cnt \le \Delta max(\hat{x}[n_0], u_0\text{-}h)$

- 应用收到的控制动作 $u[n_0 + \Delta] = \mathcal{NN}_c(y[n_0])$。
- 启动对 $\mathcal{NN}_c$ 的本地计算，用于下一个时间区间。
- 启动对 $\mathcal{NN}_p$ 的本地计算，用于下一个时间区间。
- $n_0 - \Delta$ 成为下一卸载周期起始索引的时间 $n_1 - 1$。（参见上文的 $n_0 - 1$ 时间索引。）

II) 卸载失败： （从 $n_0 + \Delta cnt$ 恢复时间线）

[⋮]

[$n_0+\Delta_{\max}$] 未收到边缘端响应，且安全期限已过； $\Delta cnt = \Delta max(\hat{x}[n_0], u_0\text{-}h)$

- 维持原有的零阶保持控制 $u[n_0 + \Delta\max(\hat{x}[n_0], u_0\text{-h})] = u_0$-h（期间没有任何更新。）
- 启动对 $\mathcal{NN}_c$ 的本地计算，用于下一个时间区间。
- 启动 $\mathcal{NN}_p$ 的本地计算，以用于下一个时间间隔。
- $n_0 - \Delta\max(\hat{x}[n_0])$ 成为下一个卸载周期起始索引的时间 $n_1 - 1$。（参见上文的 $n_0 - 1$ 时间索引。）

具体而言，请注意两个关键事实。首先，如果**C2** 返回 $\Delta\max(\hat{x}, u) = 0$，那么它实际上会强制对 $\mathcal{NN}_c$ 和$\mathcal{NN}_p$进行纯车载评估。其次，我们确保在**C1** 和**C2** 采取行动之前，它们始终都能获得最新的状态估计。

4 ENERGYSHIELD：可证明安全的卸载

4.1 主要形式化结果

4.1.1 形式化假设。 本节首先列出关于自动驾驶系统的形式化假设。这些假设主要围绕第3节所述的 EnergyShield 结构展开。

假设 1（自动驾驶系统安全性）。考虑平面上的一个固定点，将其视为自动驾驶系统需要避开的静态障碍物；并将以原点为圆心、半径为 $\bar{r}$ 的圆盘视为不安全状态集合；参见图 *2*。

假设 2（ADS模型）。假设 *1* 成立。因此，假设 *ADS*车辆由 (1) 中的 *KBM*动力学建模。此外，假设与该模型的交互以离散时间进行，并采用零阶保持。令 T 为采样周期。

假设 3（ADS传感器）。假设假设 *1* 和 *2* 成立。设采用 *KBM* 建模的自动驾驶系统在每个离散时间步都能获取传感器信号 $s[n] \in \mathbb{R}^N$ 的采样值，并且存在一个感知神经网络 $\mathcal{NN}_p$： $s[n] \mapsto \chi[n]$，它将每个离散时刻的传感器信号映射到同一时刻的（精确）*KBM*状态 $\chi[n]$。

假设 4（ADS控制）。假设假设 *1–3* 成立。设这辆采用 *KBM* 建模的 *ADS*车辆配备了一个神经网络控制器 $\mathcal{NN}_c$： ω s ，该控制器在每次采样时都能获取传感器测量值。

假设 5（ADS本地计算）。假设假设 *1–4* 成立。设 *ADS* 的车载硬件能够在不到 T 秒的时间内计算出 $\mathcal{NN}_p$ 和 $\mathcal{NN}_c$ 的输出——也就是说，计算并非瞬时完成。因此，设使用车载硬件时，控制动作的获取存在一个采样周期的计算延迟：即在采样时刻 $n+1$ 应用的控制动作是 $\omega[n+1] = \mathcal{NN}_c(s[n])$。

4.1.2 组件设计问题。需要解决两个核心问题，分别对应 EnergyShield 两个主要组件的设计：**C1**和**C2**（参见第 3 节）。

这些问题的解分别留待第 4.2 节和第 4.3 节讨论。我们在此陈述这些问题，以便在下一小节陈述主要结果。

问题1（控制器屏蔽器设计 **(C1)**）。假设假设*1–5*成立。问题是进行如下设计：首先，设计函数 h 和 α，使其构成 *KBM* 的零化屏障函数（参见）；然后利用该零化屏障函数，为 *KBM* 模型设计一个控制器屏蔽器 $\mathfrak{S}$。对于采用零阶保持输入的离散时间版本的 ***KBM***，所得控制器屏蔽器还必须具备以下附加性质：

- 设 $\chi[n_0-1]$ 和 $\chi[n_0]$ 为 *KBM* 状态，满足 $h(\chi[n_0-1])\ h(\chi[n_0]) > 0$，并设 $\chi[n_0]$ 是由在状态 $\chi[n_0-1]$下施加可行输入$\hat{}\omega[n_0-1]$ 后得到的状态。那么控制动作

$$\hat{\omega}[n_0] = \mathfrak{S}(\chi[n_0-1], \omega[n_0]) \tag{9}$$

必须产生一个状态 $\chi[n_0+1]$，使得 $h(\chi[n_0+1]) > 0$；即，控制器屏蔽器在 *KBM* 离散化和单步估计延迟（与 $\mathcal{NN}_p$ 相关联）下仍能保持安全性，与不卸载任何计算时的情况相同。

问题 2（运行时安全监视器设计**(C2)**）。设假设 *1–5* 成立，并假设 h、α 和 $\mathfrak{S}$ 是问题*1*的解。接下来要设计一个运行时安全监视器：

$$\Delta_{max} : \mathbb{R}^3 \times \Omega_{admis.} \to \mathbb{N} \tag{10}$$

具有以下性质：

- 设 $\chi[n_0-1]$ 满足 $h(\chi[n_0-1]) > 0$。那么，对于从 $\chi[n_0-1]$ 开始、施加于离散化 *KBM* 的恒定控制 $\omega = \omega[n_0]$，以下结论成立：

$$\forall n = 0, \ldots, \Delta_{max}(\chi[n_0-1], \omega[n_0])\ .\ h(\chi[n_0-1+n]) > 0 \tag{11}$$

即，恒定控制 $\omega = \omega[n_0]$ 从状态 $\chi[n_0-1]$ 起至少在 $\Delta_{max}(\chi[n_0-1], \omega[n_0])$ 个采样内保持安全性。

（$\chi[n_0-1]$ 中的延迟计入了 $\mathcal{NN}_p$的计算时间 .）

4.1.3 主要结果。鉴于这些问题的解，可以直接得出我们的主要形式化结果。

定理 1. 设假设*1–5*成立，并假设存在一个用于 *KBM*动力学的零化屏障函数，利用该函数可以求解问题*1*和问题 *2*。

那么，第 *3.3* 节所述的卸载策略能够保证基于 *KBM* 建模的自动驾驶系统的安全性（假设*1*和*2*）。

证明。该证明主要由构造得出。每个卸载周期的持续时间都受运行时安全监视器限制；因此，能够求解问题 2 的安全监视器可以确保在卸载周期内执行指定恒定控制动作时的安全性。根据问题1中控制器屏蔽器的附加特性，卸载周期结束后也能维持安全性：也就是说，如果仍有充足的安全裕度，则可以决定进行新一轮卸载；如果没有卸载安全裕度，则以闭环方式执行防护机制。□

推论1. 设假设*1–5*成立，并考虑第*2.4*节中指定的 *KBM*动力学零化屏障函数。那么，第*4.2*节中的控制器屏蔽器使用该零化屏障函数并求解问题*1*；同样，第*4.3*节中的运行时监视器使用该零化屏障函数并求解问题 *2*。因此，我们对 *EnergyShield* 的实现是安全的。

4.2 KBM控制器屏蔽器

幸运的是，我们可以使用现有的、为 KBM 设计的零化屏障函数和控制器屏蔽器：见第 2.4 [15]节。也就是说，按照 [15] 中的设计方法选择参数 σ 后，即可获得零化屏障函数（见第 2.4 节）；为简便起见，我们不再进一步讨论这一设计过程。因此，在本节中，我们将已完整实现的控制器屏蔽器称为 $\mathfrak{S}$KBM，并假定它是针对相关的 KBM 模型和安全参数 $\bar{r}$设计的（见图 2）；即假设1和假设2。

因此，本节的主要要求是根据需要修改 $\mathfrak{S}$KBM，使其满足问题1所要求的附加性质。以下引理将对此进行说明，并定义相关修改。

引理1. 通常，设假设*1–5*成立，并设 $\mathfrak{S}$*KBM* 是根据第 *2.4* 节在这些假设下设计的控制器屏蔽器。则存在一个 $\rho > 0$，使得以下控制器屏蔽器：

$$\mathfrak{S}^{\rho}_{KBM} : ((r,\xi,v),\omega) \mapsto \begin{cases} \mathfrak{S}_{KBM}((r-\rho,\xi,v),\omega) & r-\rho \geq r_{min}(\xi) \\ \beta_{max} & r-\rho \geq r_{min}(\xi) \wedge \xi \geq 0 \\ -\beta_{max} & r-\rho \geq r_{min}(\xi) \wedge \xi < 0 \end{cases} \tag{12}$$

解决问题*1*。

该引理的证明推迟至附录A。

关于引理1，还需补充一点说明。请注意，修改后的控制器屏蔽器 $\mathfrak{S}^{\rho}$ KBM保留了 [15]中设计的控制器屏蔽器 $\mathfrak{S}$KBM 的能效实现；实际上只需额外增加两个ReLU单元，因此每次评估的能耗增幅微不足道。

4.3 KBM运行时安全监视器：生成安全到期时间

回顾一下，EnergyShield 的运行时安全监视器必须提供一段失效时间，在卸载周期内保障车辆的安全，

在这段时间内，仅施加一个固定的控制输入。该失效时间必须附带可证明的保证，确保车辆安全在此期间不会受到损害。在 EnergyShield 和问题 2 的表述中，这意味着只有 $h_{\bar{r},\sigma}$ 必须在运行时安全监视器所给截止时间失效之前始终保持非负：参见问题 2 的条件 (11)。

这一表述很方便，因为这意味着问题可以再次在连续时间中进行分析，这与我们上文对问题1的考量不同：转换回离散时间时需要进行向下取整操作；而补偿计算 $\mathcal{NN}_p$ 所引入的一个采样的状态延迟，则需要从结果中减去一个采样。也就是说，要解决问题 2 并设计一个 EnergyShield 安全监视器，只需提供一个（实数）时间 ν，使得：

$$\forall t \in [0, \nu]\ .\ h\big(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)\big) > 0. \tag{13}$$

换句话说，从 $\chi[n_0-1]$ 开始、使用恒定控制 $\omega[n_0]$ 的 f 流，在持续时间 $[0,\nu]$ 内保持 $h>0$。我们再次强调，这样的 ν 可以通过向下取整操作并减去 1，转换为输出 $\Delta\max(\chi[n_0-1], \omega[n_0])$ 所需的正确采样单位。基于此，我们给出以下引理，以解决问题 2。

引理2。 按照惯例，假设*1–5*成立。令

$$\Delta_{max}(\chi[n_0-1,\omega[n_0]) \triangleq \max(\lfloor \nu(\chi[n_0-1],\omega[n_0])\rfloor - 1, 0) \tag{14}$$

其中 $\nu = \nu(\chi[n_0-1], \omega[n_0])$ 是方程的解：

$$\sqrt{2}\cdot L_{h_{\bar{r},\sigma}}\cdot \|f_{KBM}(\chi[n_0-1],\omega[n_0])\|_2\cdot \nu\cdot e^{L_{f_{KBM}}\cdot\nu} = h(\chi[n_0-1]) \tag{15}$$

对于 $L_{h_{\bar{r},\sigma}}$ 和 $L_{f_{KBM}}$ 分别是 $h_{\bar{r},\sigma}$ 和 f_{KBM} 的利普希茨常数上界。

则如此定义的 $\Delta max(\chi[n_0-1,\omega[n_0])$ 可求解问题 *2*。

引理2的证明见附录B。

引理2给出了问题 2 的完整解，正如所述。然而，直接使用该形式时，每次计算 $\Delta\max(\chi[n_0-1],\omega[n_0])$ 都需要对式（15）进行数值求解；也就是说，每当运行时安全监视器请求安全到期时间时（若卸载周期在卸载前终止，则每次采样都需要请求）。式（15）的性质使得数值求解并不特别繁重，尤其是与其取代的神经网络计算相比；不过，在实际应用中，也可以将其实现为查找表，以提高能效。

5 实验与发现

本节旨在评估以下几个关于 EnergyShield：（*i*）与传统的连续本地执行模式相比，在各种网络条件和服务器延迟下可实现的节能幅度；（*ii*）通过障碍物碰撞规避确保安全的能力；（*iii*）边缘响应时间上界（Δ_{max}）在多大程度上能代表相应驾驶情境中固有的风险；以及（*iv*）与具有不同鲜明行为特征的控制器集成时的泛化性，尤其是在采用不同学习所得驾驶策略时的泛化性。

5.1 实验设置

5.1.1 运行策略：除了基线连续本地执行外，我们还定义了 EnergyShield 的两种卸载模式：

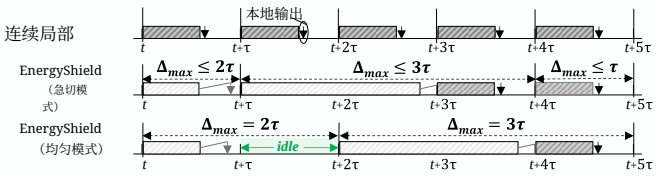

图4：在基准时间窗口 τ 下，我们实验中的运行策略。颜色越深表示本地执行。

- 急切模式：</style}若自动驾驶系统（ADS）已收到边缘响应，或 Δ_{max} 已过期，则立即开始新的卸载间隔。
- 均匀模式：无论是否已收到边缘响应，新卸载间隔的开始时间都始终延迟至 Δ_{max} 到期。

我们定义这两种模式，以体现 EnergyShield 在实现理想控制行为或最大化能效方面所能达到的行为权衡。图4展示了这一区别：在第一个卸载间隔中，均匀型 EnergyShield 收到边缘响应后会保持空闲，直到 Δ_{max} 到期；而急切模式下的 EnergyShield 则不会如此。

5.1.2 实验场景： 我们使用用于自动驾驶研究的 CARLA 开源模拟器 [11]开展实验。我们遵循 [15], 中提出的设置，并实现了类似的实验场景。具体而言，该场景包含一辆四轮车辆，沿着一条长 100 米的高速公路赛道从起点 A 驶向目的地 B，途中有 4 个行人障碍物。第一个障碍物在行驶 40 米后出现，其余障碍物的生成位置则在第一个障碍物的位置与最终目的地之间均匀分布，并可能沿纵轴发生 $\pm 10m$ 变化。

5.1.3 实验设置： 在本节中，我们的所有 实验均在以下两个二元配置参数的不同组合下进行：

- **S**：该二进制变量表示控制器屏蔽器组件是否处于活动状态（见第 3.2 节）。
- **N**：该二进制变量表示这是否为更具挑战性的“噪声”版本实验测试案例。

具体而言，噪声版本会在纵向和纬向上，分别为障碍物的生成位置添加从正态分布 $\mathbf{N}(\mathbf{0}, 1.5m)$ 中采样得到的额外偏移值。例如，配置设置 $(\mathbf{S}=1, \mathbf{N}=0)$ 表示实验是在控制器屏蔽器处于活动状态、且障碍物生成位置没有受到扰动的情况下进行的。

5.1.4 仿真设置：对于控制器模型，其第一阶段包含两个并行模块：一个作为自动驾驶系统大型神经网络模型的目标检测器，以及一个变分自编码器（β-VAE），用于提供驾驶场景的额外潜在特征表示。两个组件均处理来自车辆所安装的前向摄像头的 160×80 RGB图像。在后续阶段，强化学习（RL）智能体会汇总检测器的边界框预测、潜在特征和惯性测量（δ_f^c、 v和 a），以预测车辆控制动作（转向角和油门）。

https://github.com/shieldNN/shieldNN2020

惯性测量可以直接从CARLA获取，其位置和方向测量也用于直接计算相对于车辆当前最近障碍物的 r 和 ξ，以进行障碍物状态估计。我们使用奖励函数R训练强化学习控制器智能体，该函数旨在通过碰撞规避最大限度地提高赛道完成率，同时尽量减小车辆相对于主赛道的中心偏差。有关R的定义以及强化学习智能体的详细信息，请参阅附录C。

5.1.5 性能评估： 我们使用预训练的 ResNet-152 作为目标检测器，并在工业级 Nvidia Drive PX2 Autochauffer ADS 上部署该检测器，通过延迟和能耗来评估其性能。我们发现，ResNet-152 的单次推理过程耗时 ≈ 16ms，因此将 CARLA 中的时间步长设为 20 ms，因为检测器闭环是仿真中的计算瓶颈。为评估无线传输功率，我们采用 [18] 中的数据传输功耗模型，并假设使用 Wi-Fi 通信链路。

5.1.6 无线信道模型： 我们将自车与边缘服务器之间的通信开销建模如下：

$$L_{comm} = L_{Tx} + L_{que} \quad s.t., \; L_{Tx} = \frac{data_size}{\phi}, \tag{16}$$

其中，L_{que} 表示服务器端可能出现的排队延迟，而 L_{Tx} 是传输延迟，其定义为传输数据量 $data_size$ 除以实际信道吞吐量 ϕ。这里，我们假设 ϕ 是自车所体验到的"有效"信道吞吐量，并将可能发生的丢包、重传等情况考虑在内。我们采用 [28] 中的瑞利分布模型，对吞吐量值 $\phi \sim Rayleigh(0, \sigma_\phi)$ 进行采样，其均值为零，尺度参数为 σ_ϕ（默认值为 σ_ϕ=20 Mbps）。排队延迟 q 的建模以及服务器响应时间的估算（即 $\hat{\Delta}$）详见附录C。

5.2 EnergyShield评估

本实验旨在评估在加入 EnergyShield 后，控制器在能效和安全性方面的表现。对于 S 和 N 的每种配置，我们运行测试场景 35 个回合，并汇总其综合结果。

能效： 我们首先评估 EnergyShield 相较于基线连续本地方案所带来的能效提升
执行。如图5所示，左侧条形图表明，与本地执行相比，EnergyShield 的两种模式在所有 S 和 N 配置下都显著降低了神经网络的能耗。在默认配置（S = 0，N = 0）下，EnergyShield 的能耗分别降低了 20% 和 40.4%，对应急切模式和均匀模式。对于后续 N = 1 或 S = 1 的配置，这些数值还会进一步提升。经分析，我们发现，这是因为自车遇到了更多障碍物不在其朝向正前方视线内的情况。原因在于，当 N = 1 时，部分障碍物可能会偏离自车为完成赛道而行驶的主车道；而当 S = 1 时，此类情况则源于 ControllerShield 对神经网络预测的转向输出施加纠正行为，使车辆相对于障碍物呈现出更多切向朝向（即 $\xi \to \pm\pi/2$）。因此，较大的 Δ_{max} 值 —— 约为 4–5

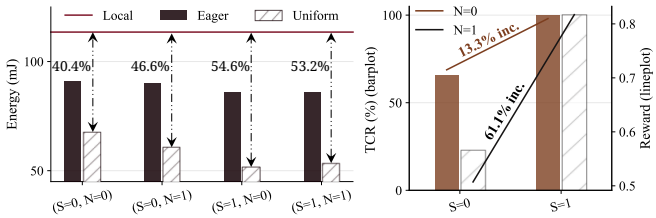

图5：EnergyShield相较于连续本地执行的能效提升（左）以及以$\mathcal{R}$ 评估和% TCR（右）表示的安全性分析。

时间采样点（相当于 80–100ms）会被越来越多地采样，这也会自动转化为更多的卸载决策。例如，在（S = 1，N = 0）下，能效提升分别达到 24.3% 和 54.6%，对应急切模式和均匀模式。

安全性评估： 为评估 EnergyShield 强制保障安全性的能力，我们将赛道完成率（TCR %）指定为比较指标，用以表示车辆能够无碰撞完成赛道的比例。以本地执行模式作为测试场景，图5右侧的条形图显示，在控制器屏蔽器未激活时（S = 0），与行人障碍物发生碰撞，导致 N = 0 22 9 = 1 时的 TCR% 为 65.7%，而在噪声测试用例中更低，为 . %（N ）。然而，激活控制器屏蔽器后（S = 1），碰撞完全避免，两种情况下的 TCR (%) 均升至 100%。这也体现在 R 的相应提升上，分别达到 13.3% 和 61.1%。为进一步展示此类情况，我们在图6中分析自车在 3 个具有不同 (S, N) 配置的回合中所选择的轨迹。如图所示，(S =0, N = 0) 实例与行人目标发生碰撞，未能完成赛道。然而，激活控制器屏蔽器（S = 1）后，会强制执行向左或向右的纠正机动动作，以避障并维持安全性。图中还提供了这些动作的示意图。

能耗与距离： 为了评估运行时安全监视器提供的 Δ_{max}上界在多大程度上能够代表相应的驾驶场景语境，我们考察了 EnergyShield 在距离最近障碍物（r）不同距离下的能耗。我们的假设是，较大的 r值意味着相对"安全"的驾驶情境，因此会产生较大的 Δ_{max} 采样值，进而产生更多卸载实例，提升神经网络的能效。如图7所示，我们绘制了 EnergyShield 的两种模式相对于本地执行的平均实际归一化能耗评分，数据涵盖每种配置下的 35 个回合，并按 r 进行统计。横轴上的每个刻度代表一个完整的 1 米距离区间，而非单个距离值；例如，横轴上的值 2 涵盖范围为 [2- 3）的所有距离。在距离较近时（r < 4米），我们发现 EnergyShield 的两种模式产生的能耗开销几乎与默认本地执行相同。这主要是因为运行时安全监视器识别出与物体距离过近所带来的较高风险，并相应地输出较小的 Δ_{max}值，而这些值只能通过本地执行来满足。随着与障碍物的距离增加， Δ_{max}值也随之增大，导致卸载实例数量逐渐增加，能耗继而持续降低。例如，对于 (S , N Δ_{max}) 配置，急切模式和均匀模式

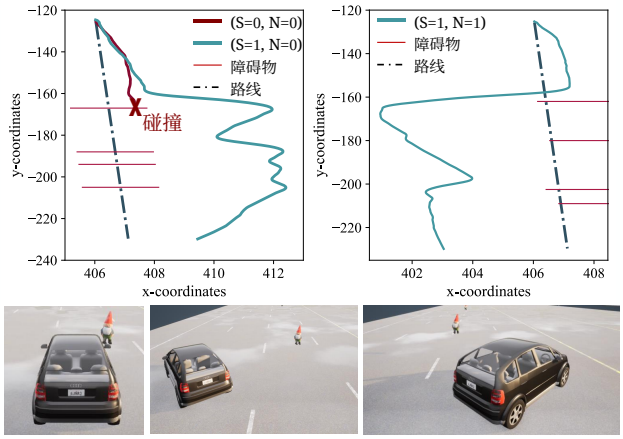

图6：上图：自车跟随的示例轨迹，起点位于顶部。下图：示意自车在上述运行模式下对首次遇到的障碍物所做出的反应。

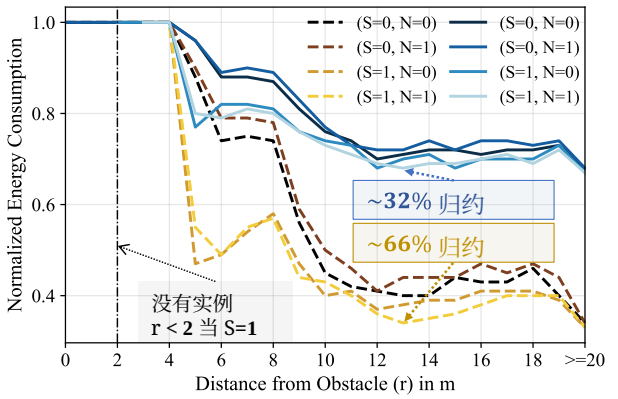

图7：急切模式（实线）和均匀模式（虚线）下 EnergyShield 的归一化能量增益，相对于与障碍物的距离（r）（单位：米）。

分别在 r = 13 米处实现了 32% 和 66% 的能耗归约。更进一步，在（ > 20 33 67 米）区间内，急切模式和均匀模式的所有配置分别实现了 % 和 % 的能量增益。

5.3 无线信道变化

在本实验中，我们评估无线信道条件变化对 EnergyShield 性能提升的影响。具体而言，针对信道吞吐量 ϕ 或排队延迟 q 可能发生的变化，我们研究 EnergyShield 带来的节能效果会在多大程度上有所变化。此外，对于每组实验运行，我们还考察完成单个卸载实例所需的额外转换窗口占总经过时间窗口的比例，并将其称为 % 额外转换窗口指标。接下来，我们首先在 q = 1 ms 固定时，分析 $\sigma_\phi \in$ {20，10、5} Mbps 变化的影响；然后在 σ_ϕ = 10 Mbps 固定时，分析 $q \in$ {10，20、50} ms 变化的影响。对于均匀型 EnergyShield，我们从图 8 中注意到，% 额外传输窗口在

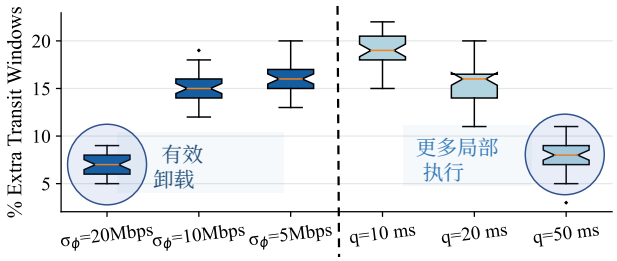

图 8：分析在不同 σ_ϕ 和 q 条件下，均匀型 EnergyShield 在 35 个回合中的额外中转窗口百分比。

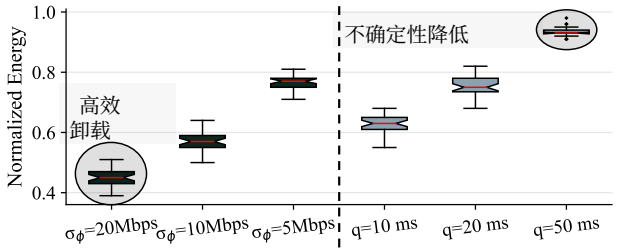

图 9：分析在不同 σ_ϕ 和 q 条件下，均匀型 EnergyShield 在 35 个回合中的归一化能耗。

高吞吐量（σ_{phi} = 20 Mbps）和高排队延迟（q = 50 ms）这两种对比条件下均有所下降，中位数分别达到 7% 和 8%。这是因为良好信道条件（σ_ϕ = 20 Mbps）意味着大多数卸载实例都能在一个时间窗口内完成，几乎不需要额外的传输窗口。而在不利的无线条件下（q = 50 ms），Δ 值通常会超过Δ_{max}，因此 EnergyShield 会更频繁地选择本地执行，以避免无线不确定性，从而减少传输窗口总数。李生图 9 也呈现了类似趋势：EnergyShield 的能耗在这些对比条件下有所变化，在 σ_ϕ=20 和 q=50时，分别达到本地执行能耗的 45% 和 93%（中位数）。

5.4 泛化性

为评估 EnergyShield 的泛化性，我们额外训练了 3 个强化学习控制器，以评估 EnergyShield 在维持安全保证方面的一致性，以及每个智能体具有不同驾驶行为时能效提升会如何变化。因此，我们对这些额外控制器重复了主要测试运行，计算每种可行的 (S, N) 配置下 35 个回合的平均能耗和 TCR。我们还报告了自车相对于主赛道车道的平均中心偏差（CD），以此作为表征各控制器不同驾驶行为的指标。在我们的实验测试案例中，CD 越大，r 值越大；也就是说，CD 值较低的控制器倾向于更靠近障碍物行驶，因此碰撞风险更高（通过最小化 CD 来最大化 R）；而 CD 值较大则表明智能体已学会驶入赛道上更靠外侧的车道，以优先考虑碰撞规避来最大化赛道完成次数（通过最大化 TCR 来最大化 R）。下文将重点介绍主要发现，详细结果请参阅附录C中的表 1。

我们的首项主要发现是，当 $S = 0$ 时，TCR (%) 取决于 CD 评分，因为只有 CD > 5m 的控制器能够持续达到 100% TCR。然而，在 $S = 1$ 时，这种依赖关系不再成立；当 $S = 1$ 时，控制器屏蔽器会强制执行障碍物规避安全保证，从而使所有控制器无论 CD 值如何都能达到 100% TCR。另一个有趣的发现是，在 EnergyShield 的两种模式下，CD 评分较大的控制器平均能耗都更低。例如，在 ($S = 1, N = 1$) 时，若将强化学习控制器按 CD 值从小到大排列为 $2.8 \rightarrow 3.6 \rightarrow 5.4 \rightarrow 5.7$ m，则其平均推理能耗依次递减为 $53.1 \rightarrow 45.7 \rightarrow 42.1 \rightarrow 39.8$ mJ。这确实凸显了 EnergyShield 执行安全且有效的上下文感知卸载的能力；尤其是运行时安全监视器能在更安全的情形下（例如与障碍物距离更远时）提供较大的 Δ_{max}，从而实现更高的能量增益；而控制器屏蔽器则始终能够维持安全保证，不受卸载决策影响。

参考文献

[1] Anayo K Akametalu、Jaime F Fisac、Jeremy H Gillula、Shahab Kaynama、Melanie N Zeilinger 和 Claire J Tomlin。2014。基于可达性的高斯过程安全学习。载于第 *53* 届 *IEEE* 决策与控制会议。IEEE。[2] Mohammed Alshiekh、Roderick Bloem、Ruediger Ehlers、Bettina Könighofer、Scott Niekum 和 Ufuk Topcu。2017。基于屏蔽的安全强化学习。（2017）。[3] Sabur Baidya、Yu-Jen Ku、Hengyu Zhao、Jishen Zhao 和 Sujit Dey。2020。面向新兴网联与自动驾驶汽车应用的车载与边缘计算。载于*ACM/IEEE* 设计自动化会议（*DAC'2020*）。[4] Felix Berkenkamp、Matteo Turchetta、Angela Schoellig 和 Andreas Krause。2017。具有稳定性保证的安全模型式强化学习。载于神经信息处理系统进展。[5] Luke Chen、Mohanad Odema 和 Mohammad Abdullah Al Faruque。2022。Romanus：模块化多传感器自动驾驶系统中的鲁棒任务卸载。载于第 *41* 届 *IEEE/ACM* 国际计算机辅助设计会议论文集。1–8。[6] Richard Cheng、Gábor Orosz、Richard M Murray 和 Joel W Burdick。2019。通过用于安全关键型连续控制任务的屏障函数实现端到端安全强化学习。（2019），3387–3395 页。[7] Yinlam Chow、Ofir Nachum、Edgar Duenez-Guzman 和 Mohammad Ghavamzadeh。2018。一种基于李雅普诺夫的安全强化学习方法。载于神经信息处理系统进展。8092–8101。[8] Yinlam Chow、Ofir Nachum、Aleksandra Faust、Edgar Duenez-Guzman 和 Mohammad Ghavamzadeh。2019。面向连续控制的基于李雅普诺夫的安全策略优化。*arXiv* 预印本 *arXiv:1901.10031*（2019）。[9] Mingyue Cui、Shipeng Zhong、Boyang Li、Xu Chen 和 Kai Huang。2020。通过边缘计算卸载自动驾驶服务。*IEEE* 物联网期刊7，10（2020），10535–10547。[10] Charles Dawson、Sicun Gao 和 Chuchu Fan。2022。利用学习得到的证书实现安全控制：神经李雅普诺夫函数、屏障函数和收缩方法综述。[11] Alexey Dosovitskiy、German Ros、Felipe Codevilla、Antonio Lopez 和 Vladlen Koltun。2017。CARLA：一个开放的城市驾驶模拟器。载于机器人学习会议。PMLR，1–16。[12] Souradeep Dutta、Susmit Jha、Sriram Sankaranarayanan 和 Ashish Tiwari。2018。深度前馈神经网络的输出范围分析。载于*NASA* 形式化方法研讨会。Springer。[13] Mahyar Fazlyab、Alexander Robey、Hamed Hassani、Manfred Morari 和 George Pappas。2019。深度神经网络利普希茨常数的高效精准估计。载于神经信息处理系统进展。[14] Jingyun Feng、Zhi Liu、Celimuge Wu 和 Yusheng Ji。2018。面向车联网的移动边缘计算：卸载框架与作业调度。*IEEE* 车辆技术杂志14，1（2018），28–36。[15] James Ferlez、Mahmoud Elnaggar、Yasser Shoukry 和 Cody Fleming。2020。ShieldNN：一种可证明安全的神经网络过滤器，用于过滤不安全的神经网络控制器。*arXiv* 预印本 *arXiv:2006.09564*（2020）。[16] Jaime F Fisac、Anayo K Akametalu、Melanie N Zeilinger、Shahab Kaynama、Jeremy Gillula 和 Claire J Tomlin。2018。一种用于不确定机器人系统中基于学习的控制的通用安全框架。*IEEE* 自动控制汇刊64，7（2018）。[17] Vijay Govindarajan、Katherine Driggs-Campbell 和 Ruzena Bajcsy。2017。人在回路系统的数据驱动可达性分析。载于*2017* 年 *IEEE* 第 *56* 届年度决策与控制会议（*CDC*）。IEEE，2617–2622。

[18] Junxian Huang、Feng Qian、Alexandre Gerber、Z Morley Mao、Subhabrata Sen 和 Oliver Spatscheck。2012。深入考察 4G LTE 网络的性能和功耗特性。载于《第十届移动系统、应用与服务国际会议论文集》。225–238。[19] Radoslav Ivanov、James Weimer、Rajeev Alur、George J Pappas 和 Insup Lee。2019。Verisig：使用神经网络控制器验证混合系统的安全属性。载于《第 22 届 ACM 混合系统：计算与控制国际会议论文集》。169–178。[20] Torsten Koller、Felix Berkenkamp、Matteo Turchetta 和 Andreas Krause。2018。用于安全探索的基于学习的模型预测控制。载于《2018 年 IEEE 决策与控制会议（CDC）》。IEEE。[21] Jason Kong、Mark Pfeiffer、Georg Schildbach 和 Francesco Borrelli。2015。面向自动驾驶控制设计的车辆运动学与动力学模型。载于《2015 年 IEEE 智能车辆研讨会（IV）》。IEEE，1094–1099。[22] Shih-Chieh Lin 等。[不详]。自动驾驶的架构影响：约束与加速。载于《编程语言与操作系统架构支持国际会议论文集（ASPLOS'18）》。[23] Changliu Liu、Tomer Arnon、Christopher Lazarus、Clark Barrett 和 Mykel J Kochenderfer。2019。深度神经网络验证算法。arXiv 预印本 arXiv:1903.06758（2019）。[24] Shaoshan Liu、Liangkai Liu、Jie Tang、Bo Yu、Yifan Wang 和 Weisong Shi。2019。面向自动驾驶的边缘计算：机遇与挑战。《IEEE 会议录》107，8（2019），1697–1716。[25] Christian Llanes、Matthew Abate 和 Samuel Coogan。2022。通过快速闭环可达性分析保障安全，并应用于无人机。载于《2022 年 ACM/IEEE 第 13 届网络物理系统国际会议（ICCPS）》。IEEE，127–136。[26] Arnav Malawade、Mohanad Odema、Sebastien Lajeunesse-DeGroot 和 Mohammad Abdullah Al Faruque。2021。SAGE：一种用于高效端到端自动驾驶车辆控制的分体式架构方法。《ACM 嵌入式计算系统汇刊》（TECS）20，5s（2021），1–22。[27] Aniruddh Mohan、Shashank Sripad、Parth Vaishnav 和 Venkatasubramanian Viswanathan。2020。自动化与轻型车辆电气化之间的权衡。《Nature Energy》5，7（2020），543–549。[28] Mohanad Odema、Luke Chen、Marco Levorato 和 Mohammad Abdullah Al Faruque。2022。Testudo：面向延迟关键型自治系统的协同智能。《IEEE 集成电路与系统计算机辅助设计汇刊》（2022）。[29] Alexander Robey、Haimin Hu、Lars Lindemann、Hanwen Zhang、Dimos V Dimarogonas、Stephen Tu 和 Nikolai Matni。2020。从专家演示中学习控制屏障函数。arXiv 预印本 arXiv:2004.03315（2020）。[30] Kengo Sasaki、Naoya Suzuki、Satoshi Makido 和 Akihiro Nakao。2016。云与移动边缘计算协同的车辆控制系统。载于《2016 年日本计测自动制御学会第 55 届年会》。IEEE，1122–1127。[31] Guanya Shi、Xichen Shi、Michael O'Connell、Rose Yu、Kamyar Azizzadenesheli、Animashree Anandkumar、Yisong Yue 和 Soon-Jo Chung。2019。Neural lander：利用学习得到的动力学实现稳定的无人机着陆控制。载于《2019 年国际机器人与自动化会议（ICRA）》。IEEE，9784–9790。[32] Mohit Srinivasan、Amogh Dabholkar、Samuel Coogan 和 Patricio Vela。2020。采用监督式机器学习方法综合控制屏障函数。arXiv 预印本 arXiv:2003.04950（2020）。[33] Xiaowu Sun、Haitham Khedr 和 Yasser Shoukry。2019。神经网络控制的自治系统的形式化验证。载于《第 22 届 ACM 混合系统：计算与控制国际会议论文集》。147–156。[34] Sihai Tang、Bruce Chen、Harold Iwen、Jason Hirsch、Song Fu、Qing Yang、Paparao Palacharla、Nannan Wang、Xi Wang 和 Weisong Shi。2021。VECFrame：面向网联自动驾驶车辆的车载边缘计算框架。载于《2021 年 IEEE 边缘计算国际会议（EDGE）》。IEEE，68–77。[35] Andrew J Taylor、Andrew Singletary、Yisong Yue 和 Aaron D Ames。2020。通过投影到状态安全实现情景式学习的控制屏障视角。arXiv 预印本 arXiv:2003.08028（2020）。[36] Li Wang、Evangelos A Theodorou 和 Magnus Egerstedt。2018。利用屏障证书安全地学习四旋翼动力学。载于《2018 年 IEEE 国际机器人与自动化会议（ICRA）》。IEEE，2460–2465。[37] Weiming Xiang、Diego Manzanas Lopez、Patrick Musau 和 Taylor T Johnson。2019。非线性动态系统神经网络模型的可达集估计与验证。载于《安全、自治与智能车辆》。[38] Wei Xiao、Noushin Mehdipour、Anne Collin、Amitai Y Bin-Nun、Emilio Frazzoli、Radboud Duintjer Tebbens 和 Calin Belta。2021。自动驾驶的基于规则的最优控制。载于《ACM/IEEE 第 12 届网络物理系统国际会议论文集》。143–154。[39] Xiangru Xu、Paulo Tabuada、Jessy W. Grizzle 和 Aaron D. Ames。2015。控制屏障函数在安全关键控制中的鲁棒性。《IFAC-PapersOnLine》48，27（2015），54–61。[40] Xumiao Zhang、Anlan Zhang、Jiachen Sun、Xiao Zhu、Y Ethan Guo、Feng Qian 和 Z Morley Mao。2021。Emp：边缘辅助的多车辆感知。载于《第 27 届移动计算与网络年度国际会议论文集》。545–558。

引理1的证明

证明。在此证明中，令 $\chi[n\prime] = (r[n\prime],\ \xi[n\prime],\ v[n\prime]),\ \ n\prime \in \mathbb{Z}_{\circ}$

为此，我们首先假设车辆最大速度 $v\max$ 受到上限约束（可控）。因此，令 $\gamma = 2\cdot v\max\cdot T$，其中 T为采样周期（参见第2.1节）。因此，还需注意，对于 $r[n_0-1] > r\min(\xi[n_0-1])$:

$$r[n_0+1] > r[n_0-1] - 2\cdot v_{\max}\cdot T > \bar{r} - \gamma \tag{17}$$

因此，在任意两个采样周期内

$$|\dot{\xi}| \le v_{\max}\cdot\left(\frac{1}{\bar{r}-\gamma} + \frac{1}{\ell_r}\right) \tag{18}$$

接着注意到，函数 $r\min$ 的利普希茨常数有界于 $L_{r_{\min}} \le \frac{\bar{r}\cdot\sigma}{2\sqrt{1-2\cdot(\cdot)\sigma)^2}}$。最后，得出

$$\begin{aligned}&|r_{\min}(\xi[n_0-1]) - r_{\min}(\xi[n_0+1])|\\ &\qquad \le \frac{\bar{r}\cdot\sigma}{2\cdot(1-2\cdot\sigma)^2}\cdot v_{\max}\cdot\left(\frac{1}{\bar{r}-\gamma}+\frac{1}{\ell_r}\right)\cdot 2\cdot T \triangleq \eta\end{aligned} \tag{19}$$

然后选择 $\rho \triangleq \eta + \gamma$.

现在，根据式 (12) 中修订后的安全屏蔽器的结构，问题1结论的证明可分为以下三种情况:

(i) $r[n_0] \ge r\min(\xi[n_0]) + \rho$; （无论 $r[n_0-1]$ 的位置如何）

(ii) $r[n_0] < r\min(\xi[n_0]) + \rho$ 且 $r[n_0-1] \ge r\min(\xi[n_0-1]) + \rho$; *(iii)* $r[n_0] < r\min(\xi[n_0]) + \rho$ 且 $r[n_0-1] < r\min(\xi[n_0-1]) + \rho$。在这三种情况下，我们都需要证明，对于下一状态，$h(\chi[n_0+1]) > 0$，或者等价地，$r[n_0+1] > r\min(\xi[n_0+1])$。

情形 (*i*) 和情形 (*ii*) 。对于这些情形，结论基本可由上文对 ρ 的选择直接推出。在 情形 (*ii*) 中，我们有

$$\begin{aligned}&r[n_0+1] - r_{\min}(\xi[n_0+1]) \ge \big(r[n_0-1] - r_{\min}(\xi[n_0-1])\big)\\ &-\big(r[n_0+1] - r_{\min}(\xi[n_0+1]) - r[n_0-1] - r_{\min}(\xi[n_0-1])\big).\end{aligned} \tag{20}$$

由以上计算可知，式（20）右侧的第二项有下界 $-\rho$ （利用三角不等式）。因此

$$r[n_0+1] - r_{\min}(\xi[n_0+1]) \ge \big(r[n_0-1] - r_{\min}(\xi[n_0-1])\big) - \rho \tag{21}$$

由于根据假设 $r[n_0-1] - r\ \ \xi[n_0-1] \ge \min(\rho)$，所需结论即可推出。类似的方法可证明情形 (*i*) ：只需重复 ρ 定义中的计算，但仅考虑一个采样。

因此，接下来只需考虑情形 (*iii*) 。这种情形要简单一些，因为控制信号正被覆盖，所以状态延迟不会带来上述技术难题。因此，利用 [15] 中设计的控制器屏蔽器的性质，结论几乎可以直接得出。具体而言，ShieldNN 验证器证明，“安全”控制与“不安全”控制之间的边界是关于 ξ 的凹函数（分别关于 $\xi \in [0,\pi]$）/凸函数（分别关于 $\xi \in [-\pi,0]$）。因此，根据 [15, 定理 1]，，从状态 $\xi \in [0,\pi]$ $\xi \in [-\pi,0]$ □（分别为 ）出发时，恒定控制 $\beta\max$（分别为 $-\beta\max$）在任意时长内始终能够保持安全。

B 引理2的证明

证明。根据上述论证，只要（15）蕴含（13），（14）中的 $\Delta\max$ 形式就能解决问题 2。因此，我们只需证明这一点。

首先，观察到:

$$\begin{aligned}h_{\bar{r},\sigma}(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)) \ge \Big|&\big|h_{\bar{r},\sigma}(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)) - h_{\bar{r},\sigma}(\chi[n_0-1])\big|\\ &- \big|h_{\bar{r},\sigma}(\chi[n_0-1])\big|\Big|\end{aligned} \tag{22}$$

根据三角不等式。因此:

$$\begin{aligned}&\big|h_{\bar{r},\sigma}(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)) - h_{\bar{r},\sigma}(\chi[n_0-1])\big| \le \big|h_{\bar{r},\sigma}(\chi[n_0-1])\big|\\ &\qquad\qquad \Longrightarrow\ h_{\bar{r},\sigma}(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)) \ge 0\end{aligned} \tag{23}$$

因此定义 $z(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)) \triangleq \big|h_{\bar{r},\sigma}(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)) - h_{\bar{r},\sigma}(\chi[n_0-1])\big|$。根据格朗沃尔不等式，我们进一步得到:

$$z(\zeta^{0,\chi[n_0-1]}_{1_{\omega[n_0]}}(t)) \le \sqrt{2}\cdot L_{h_{\bar{r},\sigma}}\cdot\|f_{\mathrm{KBM}}(\chi[n_0-1],\omega[n_0])\|_2\cdot t\cdot e^{L_{f_{\mathrm{KBM}}}\cdot t} \tag{24}$$

其中$L_{h_{\bar{r},\sigma}}$和 $L_{f_{\mathrm{KBM}}}$与引理陈述中的相同。注意，(24)右侧的函数关于 t是单调的。因此，我们可以利用(23)断言：若 ν 满足(15)（由(24)和(23)立即推得），则引理的结论成立。□

C 补充实验细节

C.1 训练细节

主要的强化学习智能体训练采用（S = 0，N= 0）配置，使用近端策略优化（PPO）算法，共进行了 1800 个回合。在最后 400 个训练回合中，我们随机调整自车的横向生成位置和朝向，以帮助智能体学习如何从机动动作中恢复。对于 β-VAE，我们使用了来自 [15] 的预训练模型，该模型经过训练，可从 CARLA 驾驶场景生成 64 维潜在特征向量。

C.2 奖励函数

对于奖励函数 R，我们定义如下:

$$\mathcal{R} = \begin{cases} -P, & \text{collision or } CD > CD_{th}\\ +P, & \text{track completed successfully}\\ f_{\mathcal{R}}(v, CD, \vartheta, r), & \text{otherwise}\end{cases} \tag{25}$$

其中，P是较大的正数，v是车辆的速度，CD是车辆相对于赛道中心的中心偏差，CD_{th}是预设阈值，ϑ表示车辆朝向与路段曲率切线之间的夹角，r是与最近障碍物的距离。如图所示，R 的取值可以是：(*i*) (+P)，表示车辆成功完成赛道（较大的正奖励）；(*ii*) (–P)，表示车辆发生碰撞或偏离道路中心超过 CD_{th}；或者 (*iii*) 由上述变量构成的函数 $f_{\mathcal{R}}(\cdot)$，其定义如下:

$$f_{\mathcal{R}}(v, CD, \vartheta, r) = \omega_1\cdot f_1(v) + \omega_2\cdot f_2(CD) + \omega_3\cdot f_3(\vartheta) + \omega_4\cdot f_4(r)$$

表 1：EnergyShield 在 4 种不同强化学习控制器上的性能。每个强化学习智能体都学习了一种独特的策略来沿路线行进，该策略以其相对于主赛道的中心偏差（CD）表示。强化学习控制器按 CD 值递增顺序排列，其中控制器1是所有评估中使用的主要强化学习控制器。

策略	（S, N）	控制器1			控制器2			控制器3			控制器4		
		CD(m)	TCR(%)	E(mJ)	CD（米）	TCR（%）	E（毫焦）	CD（米）	TCR（%）	E（毫焦）	CD(m)	TCR(%)	E(mJ)
局部	(0, 0)	0.92	65.7	113.5	2.3	100	113.5	5.5	100	113.5	5.8	100	113.5
	(0, 1)	0.8	22.9		2.3	97.1		5.5			5.8		
	(1, 0)	2.87	100		3.5	100		5.5			5.9		
	(1, 1)	2.91	100		3.6	100		5.4			5.7		
EnergyShield (急切模式)	(0, 0)	0.8	68.6	90.8	2.3	100	90	5.5	100	79.7	5.8	100	77.6
	(0, 1)	0.8	34.3	90	2.3	100	89			80	5.7		78.9
	(1, 0)	2.8	100	85.8	3.5	100	81.6			79.6	5.7		77.5
	(1, 1)	2.8	100	85.9	3.6	100	81.9			78.5	5.8		77.9
EnergyShield (均匀模式)	(0, 0)	0.9	74.3	67.7	2.3	100	63.5	5.5	100	43.7	5.8	100	39.7
	(0, 1)	0.7	22.9	60.6	2.3	97.1	63.1	5.5		44.4	5.7		40.8
	(1, 0)	2.9	100	51.5	3.5	100	44.5	5.5		43.6	5.8		40.1
	(1, 1)	2.8	100	53.1	3.6	100	45.7	5.4		42.1	5.7		39.8

$$\text{s.t.,}\quad f_1(v) = \begin{cases} \frac{v}{v_{min}}, & v < v_{min} \\ 1 - \frac{v - v_{target}}{v_{max} - v_{target}} & v > v_{target} \\ 1, & otherwise \end{cases}$$

$$f_2(CD) = \max(1 - \frac{l_{center}}{l_{max}}, 0)$$

$$f_3(\vartheta) = \max(1 - |\frac{\vartheta}{\pi/9}|, 0)$$

$$f_4(dist) = \max(\min(\frac{||r||}{r_{max}}, 1), 0)$$

其中，v_{min}、v_{max}、v_{target}分别是最小速度、最大速度和目标速度。l_{center}是车辆中心到指定赛道的横向距离。ϑ是车辆朝向与赛道切线之间的夹角。在我们的实验中，我们设定

v_{min}=35 v_{target}=40 v_{max}=45 r_{max} = 千米/小时、千米/小时、千米/小时、米、米，以及。 20 l_{max} = 10 P= 100

C.3 性能评估

我们使用标准的 TensorRT 库，在 Nvidia Drive PX2 ADS 平台上将模型编译为优化后的推理引擎，并测量其执行延迟。为评估本地执行功耗，我们测量 Nvidia Drive PX2 在处理任务和空闲时的平均功耗差值。

C.4 排队延迟

我们采用 M/M/1/k 模型来描述排队延迟，L_{que}，其中 $q_c = (1-\rho)\rho^c / (1-\rho^{C+1})$表示卸载~~任务到达~~时，在服务器容量为$C$ 的缓冲区中已有 c 个任务的概率，ρ 为平均服务器负载。我们假设每个任务会额外增加 1 毫秒延迟，因此，q_c 个位置可直接换算为以毫秒为单位的 L_{que}。除非另有说明，排队延迟的默认设置为 $C = 4000$ 和 $\rho = 0.97$。

C.5 边缘响应估计

由于卸载决策是根据先前边缘响应时间的估计值作出的，因此估计的通信延迟，L_{comm}，在时间 n 可定义为有效吞吐量 ϕ和排队延迟 q的 k 个先前值的函数，具体如下：$\hat{L}_{comm}(n) = \Phi(\phi_{n-k:\,n-1},\ q_{n-k:\,n-1})$。在我们的实验中，我们设定$k = 5$，并采用移动平均函数来评估 Φ。