# 基于谷歌地图的自驾驶系统路径规划强化学习

## 摘要

自动驾驶系统是人工智能的一个热门而重要的应用。我们在路径规划系统中实现了DoubleDQN算法，并介绍了学习代理与Google map交互的方法。与谷歌地图提供的路线相比，能耗减少了10%，但持续时间增加了两倍。通过应用更复杂的神经网络结构、更高分辨率的导航和对谷歌地图数据库的无限制访问，可以改善结果。

## Introduction

**Background knowledge**

强化学习是一种涉及马尔可夫决策过程(MDP)的规划算法。我们可以把它看作是对人类决策的模仿。我们称人类为**代理**，遇到一种情况，比如在餐馆里看菜单。我们称餐馆为**环境**，称当前的位置和感觉为**状态**。在餐馆里，我们做了一个决定，点了一顿饭，而“点”这个词就是**行动**。有可能你尝试了多次的同一种食物会变得比以前更辣。我们称这个几率为**状态转移概率**。这意味着，即使我们在相同的状态下做出相同的动作，结果也可能与之前的(正常味道)不同(辛辣)。餐后，我们会根据味道或用餐体验来评价这顿饭或餐厅。我们说味觉或体验是一种奖励。奖励可以是积极的，意味着我们真的喜欢食物或服务。这种餐饮体验会影响我们下一次在菜单上的几个餐馆或餐点中做出选择的决定。这将逐渐形成或改变我们制定决策的常规，我们称之为**策略**。

整个过程可以简化为图1和下面的几个字，在一个环境下，代理遇到一个状态并根据策略做出一个动作。环境会把我们带到下一个状态，并根据行动给予我们奖励。我们会做出另一个动作，获得另一个奖励，进入下一个状态，以此类推。我们将学习在一系列国家-行动-奖励程序的基础上形成和修改政策。

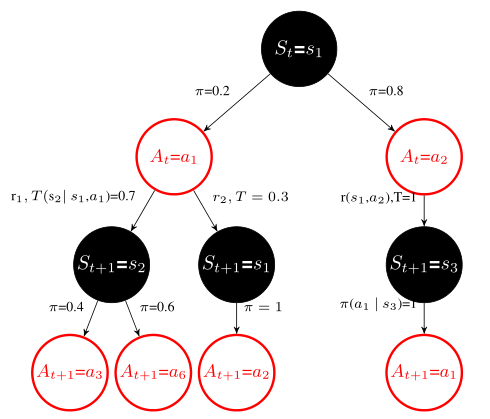
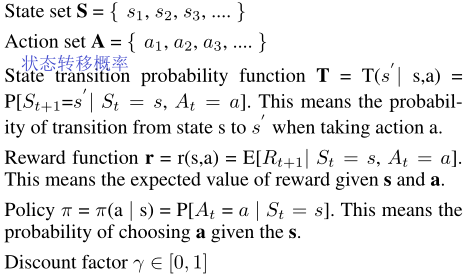


图1 状态、行动和回报之间的迭代过程。在状态St下，根据策略π，我们有0.2的机会选择行动a1，0.8的机会选择行动a2。回报R1和R2可以不同。

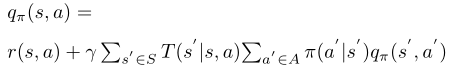
我们将在图1和以下段落中使用的一些重要术语和符号在这里指定。



### 学习过程

当我们做出一些决定和行动后，人们会从反馈中学习。我们希望最终目标的结果尽可能好。例如，一家五星级饭店位于山顶。在我们通往顶峰的路上有坏人和凶猛的动物。我们需要学习一个好方法来避开他们，到达山顶，而不是停留在半山腰的二星级餐厅。

我们的代理从每次行动的奖励中学习。我们给每个动作分配一个值，称这个动作值函数为q(s，a)。如果代理人处于状态s，有4个行动选择，那么我们得到这个状态s的四个q值。q值最初被设置为0，可以通过我们刚刚遇到的奖励来更新，此外，我们可以考虑未来的奖励，以防我们可能被当前的奖励所愚弄。这里是贴现因子γ的功能，它对未来回报的重要性进行加权。如果γ接近1，这意味着未来的回报几乎和当前的回报一样重要。我们可以用下面的等式来表示当前奖励和未来奖励的组合：



状态s’代表相对于当前状态s的下一个状态。在代理真正处于状态s’之前，我们不确定a’将被选择哪个动作，所以我们需要得到qπ(s’，a’)的期望值，这就是求和与π起作用的地方。同样，我们不确定代理在状态s采取行动a后会进入什么状态s’，所以我们需要T(s’|s，a)来表示可能性并考虑所有的可能性。在代理进入大多数状态并尝试每个状态中的大多数动作后，我们可以构建一个指令图来演示在给定特定状态下采取特定动作的质量。最后，代理人可以选择每个状态下最有可能引导我们达到最优结果的行动的最高值，我们称之为贪婪策略。我们可以通过非策略或策略来更新q(s，a ),更新算法将在下一节中指定。区别这两种策略的是我们如何更新q(s，a)。对于偏离策略，我们使用maxq(s’，a’)和r(s，a)来更新q(s，a)。对于on策略，我们使用q(s’，a’)和r(s，a)来更新q(s，a)。

值函数vπ(s)是通过考虑特定状态下的所有q值来对该状态进行分级的函数，如下式所示：

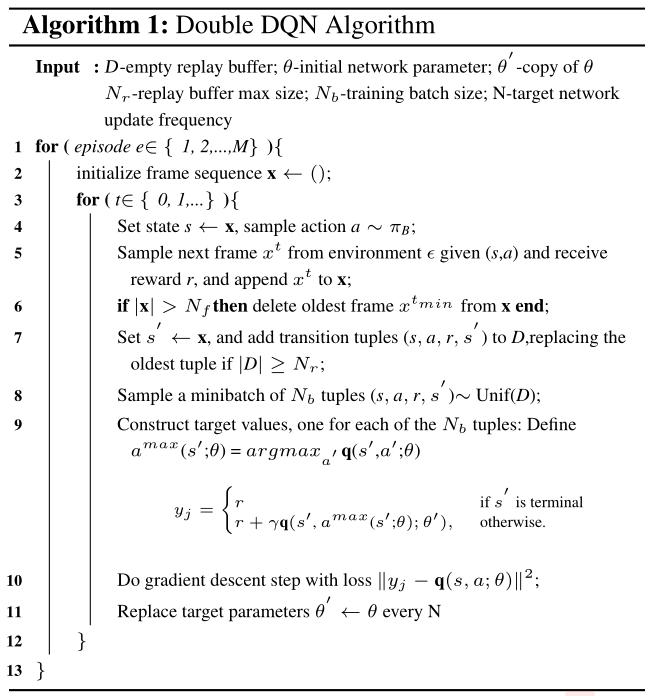


## 论文研究

### 基于Double Q-Learning的深度强化学习

在这项工作中，作者将神经网络应用于强化学习，并将其命名为Double DQN(Van Hasselt，Guez，和Silver 2016)。神经网络的功能是将状态s映射到q(s，·)。状态s的输入可以是n维向量，例如图像。输出是一个m维向量，其中每个元素可以解释为q值对应每个动作。简而言之，神经网络是从IRn到I Rm的映射函数。我们用θ来表示神经网络中的参数。

他们设立一个目标或指标来评估由θ计算出的q值。目标由另一个称为目标网络的由θ’控制的神经网络生成，该网络包含与由θ控制的神经网络相同的架构。现在我们得到两个结构相同的神经网络，分别由参数θ和θ’控制。θ’最初是从θ复制而来的。在一级的训练中(代理从开始到结束)，我们将每N步复制θ到θ’。换句话说，我们不会在这N步内更新θ’。目标的方程是Yt≡ rt+1+ γq(St+1，argmaxq(St+1，a；θt)，θ’t)。我们首先将St+1输入到θ网络中，并选择动作a对应于argmaxq(St+1，a；θt)。同时，我们将St+1输入到θ’网络中，得到另一个输出向量。我们通过结合当前报酬rt+1和来自θ’网络q(St+1，a；θ’t)。Zyiu Wang 2016年论文提供的学习算法如算法1所示。



体验重放是一种受生物启发的技术，用于消除数据序列中的相关性。我们随机选择均匀存储在重放缓存中的数据来计算损失，并在学习过程中更新权重。

### 价值迭代网络

值迭代网络(VIN)是一种无模型规划算法(Tamar et al. 2016)。我们可以使用具有标准反向传播和RL算法的VIN来处理需要视觉感知、连续控制和基于自然语言的决策的问题。我们的目标是学习一个端到端的策略，该策略可以概括解决不同的、未知的领域。

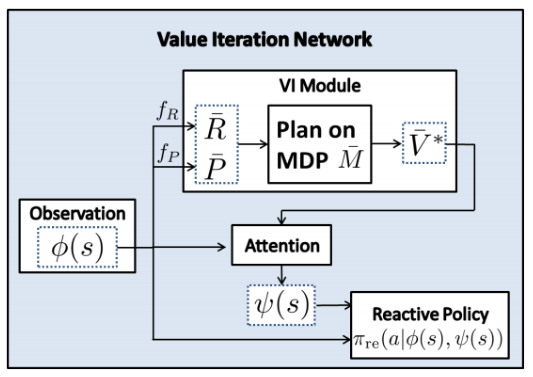


图2 价值迭代网络

在图2中，输入φ(s)是一幅图像(如地形图像)和当前状态。输出πre(a|φ(s)，ψ(s))是动作的概率向量。fR基本上是一个卷积神经网络(CNN )，它将输入图像转换为奖励图像(每个像素可以代表一个奖励值)。fp是一个状态转移函数。是一个与r大小相同的值函数。由于状态s的最优策略只能依赖于的子集附近的状态，作者使用注意力来表示这种逻辑，并输出一个向量ψ(s ),它表示我们真正关心的状态集中的值。

在图3中，它解释了VIN的核心，VI模块，一种实现方程的逻辑的机制。VI模块的每次迭代可以被视为将奖励图像R和先前的值函数传递到卷积层和最大池层，然后输出新的值函数V。卷积层中的每个通道对应于特定动作Q(s，a1)，Q(s，a2)，...Q(s，am)其中m是动作集的长度。卷积核权重对应于折扣转移概率γP(s’|s，a)。然后，该层沿着动作通道被最大化，以产生值函数V的下一次迭代，其中。然后，我们将附加到，作为它的第二个通道，并将它们馈送到卷积层和最大池层K次，以执行K次迭代，其中K是将奖励信息从目标传递到状态s的最小值。在K次迭代之后，VI模块将输出供代理做出决策。然后，我们可以应用DQN或其他RL方法来训练图2和图3中的参数。

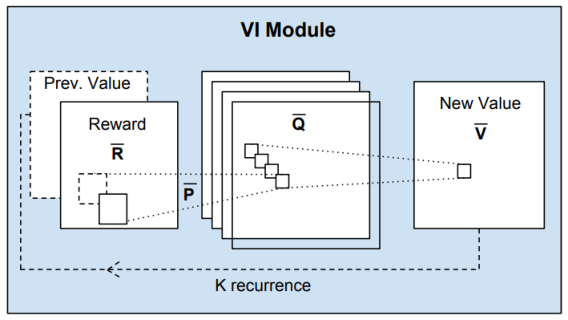


图3 值迭代模块

### 使用Style和Structure对立网络的生成图像建模(S2-GAN)

生成对抗网络包含两个模型:生成器G和鉴别器d。生成器试图生成看起来像真实图像的图像，而鉴别器试图区分真实图像和生成的图像

结构(场景的几何形状)和风格(纹理和照明)是图像形成的两个关键因素，但被最近的生成模型所忽略。本文(Wang和Gupta 2016)将生成过程分解为两个过程：(I)使用Structure-GAN生成表面法线图，以及(ii)使用Style-GAN以表面法线图作为输入并生成图像。这两个gan被独立训练，并通过加入学习合并在一起。

**Structure-GAN**：第一个生成网络的输入是(从均匀分布中采样的100维向量)，然后生成曲面法线贴图(G():72×72×3)。第一个鉴别器网络将图像(72×72×3)作为输入，并输出单个值[0，1]，该值告知表面法线贴图是真实的(接近1)还是生成的(接近0)。

**Style-GAN**：第二生成网络的输入是(从均匀分布采样的100-d向量)和地面真实表面法线，然后网络生成具有纹理和照明的图像(G(Ci，))。第二鉴别器网络的输入是地面真实表面法线映射，G(Ci，)，实像和实像的表面法线映射。本文还包括全卷积网络(FCN )，它以G(Ci，)为输入，估计它的表面法线映射，以使第二个生成网络输出与生成的表面法线映射对齐的更好的图像。

**Joint learning for S2-GAN**：在独立地训练了结构GAN和风格GAN之后，我们将一起训练两个网络，如图4所示，但是首先我们去除FCN部分。首先，我们输入并得到生成的表面法线G()，并通过将G(z)馈入结构GAN中的鉴别器网络来接收第一个损耗。其次，将G()和输入到StyleGAN的生成网络中，得到生成的图像(G(G()，))。现在，我们通过将G(G()，)和G()馈入样式为GAN的鉴频器网络来接收第二个损耗。我们将结合第一个损失和第二个损失(按0.1的比例)来训练Structure-GAN中的生成器网络，以产生更好的表面法线贴图。

我们可以应用这种技术，通过输入一些参数，如电机输出、电池输出、环境温度等，为车辆生成热图。如果热图足够真实，我们可以设计一个更好的冷却管理系统，并尽量减少热传感器的数量。自动驾驶系统可以根据车辆的状况规划更好的路线。

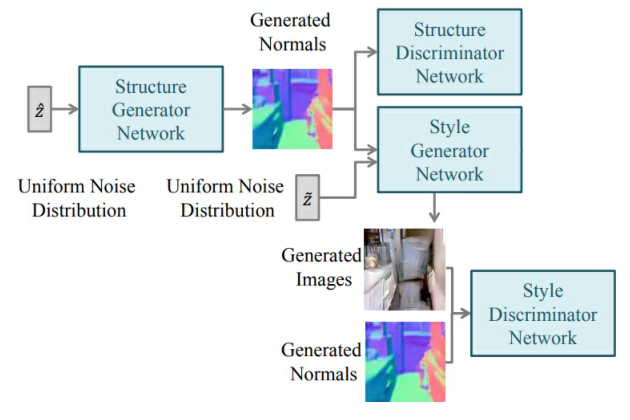


图4 S2GAN 模型

## 实验

在本节中，我们将首先介绍我们在实验中通过tensorflow库实现的双DQN算法(算法1 )，然后解释我们如何为学习代理设置环境。最后展示了实验结果。

### The earning Agent

我们的学习代理是一辆电动汽车，通过选择不同的行动(东、南、西、北)在谷歌地图环境中导航。该动作可以由双DQN或随机确定。在学习过程中，代理将首先在地图上随机导航以探索地图，但我们将逐渐减少随机选择动作的部分，而是采用双DQN模型提供的q值最高的动作。γ为0.9，Nb为32。

### 神经网络体系结构

第一个图层是输入图层，它采用当前位置的地理编码并除以180。输入层之后是两个完全连接的层(分别为10个神经元和6个神经元),具有relu激活函数，丢失率为0.25。最后一层是四维输出，代表每个动作的q值。我们初始化了两个相同的网络，第一个是Q网络，代理在当前状态下确定动作。第二个是目标网络，作为Q-网络要实现的目标。我们只进行反向传播，并通过Adam优化器以0.0001的学习率更新Q网络中每一步的权重，然后每五步将Q网络中的权重复制到目标网络。

### 环境

为了知道与Google map API(地理编码API、方向API和海拔API)提供的路线相比，我们的代理是否能够在双DQN算法下找到具有最小能量成本和可接受持续时间的最佳路线，我们设置了如下实验。

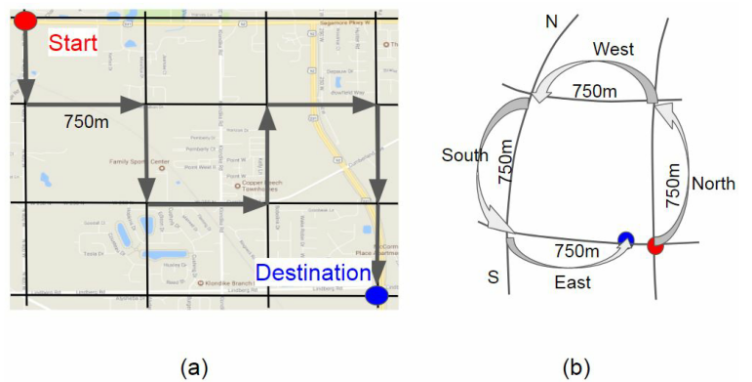


图5 (a)代理在其上导航的环境的网格地图。每个箭头的长度和方向表示代理采取的步骤和动作的某个位移长度。(b)在真实世界中，如果采取一系列循环步骤，代理将不会回到相同的位置。这是由球体几何造成的。

**与谷歌地图API交互**：我们首先指定起始位置(可能是地名、地址或地理编码)和目的位置，并输入地理编码API来检索这两个位置的地理编码。然后，我们使用起始地理编码和目的地地理编码作为两个对角来构建网格地图的矩形边界，我们的代理只能在该边界上导航。网格图和符号如图5(a)所示。严格来说，网格图中的每个网格都不是矩形。这种现象是由球体几何形状和我们对步幅长度的限制造成的，如图5(b)所示。

代理有四个方向选择(北、东、南、西)。每个箭头表示代理从当前位置s导航到下一个位置s’，在网格地图上有750米的位移。然而，代理的实际导航距离将大于或等于750米，这取决于方向API提供的路线。例如，在图6中，假设代理在由A表示的当前位置，并且向南行进到由b表示的下一个位置。显然，由方向API提供的路线是395号公路，并且距离大于750米。我们可以得到如下形式的导航指令列表：{ A的地理编码，从A到1的持续时间，从A到1的距离，1的地理编码}，{ 1的地理编码，从1到2的持续时间，1到2的距离，2的地理编码}……其中，在我们将A的地理编码和B的地理编码输入到方向API之后，A，1，2，3，B如图6所示。指令的数量基于方向API，并且在我们的图6的情况下，从A到b有四个指令。我们使用导航指令列表中的每个地理编码来从海拔API获得每个位置的高度，并且计算每个指令内的海拔，例如A(图7中的位置1)和1(图7中的位置2)之间的海拔、1和2之间的海拔、2和3之间的海拔以及3和b之间的海拔。为了简化图7所示的复杂性，我们忽略了每个指令中提供的两个位置之间的道路中间的高度。然而，代理从A到C旅行的情况将取决于方向API返回的信息。如果C在一个湖中或某个不可到达的地方，方向API将返回false，代理将把C点视为一个块。否则，位置C被认为是可到达的，并且方向API返回的导航指令列表是路线66，那么我们将分析位置A、1和4之间的高程，即使位置4与位置C不相同。我们允许方向API返回的路线位于网格地图边界之外，而代理应该总是在网格地图边界内的点上导航。

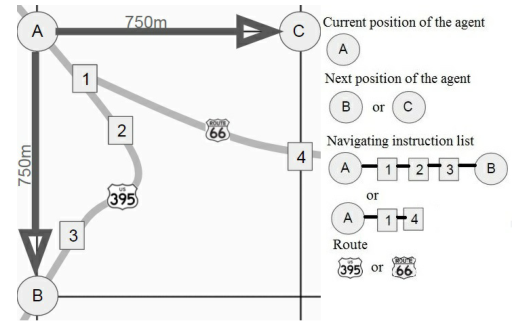


图6 该图仅显示了网格图的一部分。代理只能从一个点导航到附近的另一个点，而不能越过边界。如果代理从当前位置A移动到下一个位置b，我们将分析道路信息，例如道路395。

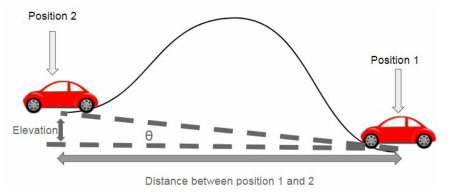


图7 我们只计算位置1和位置2之间的高程，忽略途中的高程

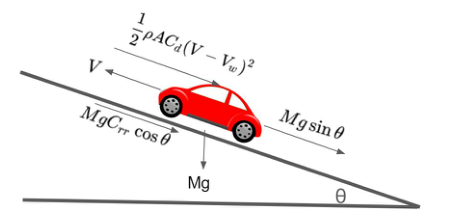


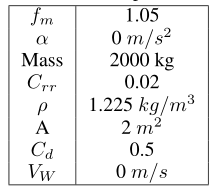
图8 施加在车辆上的力的自由体图

**能耗**：我们根据栅格地图上750米位移(实际距离超过750米)行进所需的能量来评估每个动作。例如，为了计算图6中位置A和位置1之间所需的能量，我们使用位置A和位置1之间的持续时间和距离来计算平均速度V，将V与高度结合，我们可以得到道路的角度θ，并认为道路的高度线性增加或减少，如图7所示。因为我们在实验中没有考虑再生制动，所以我们将下坡路视为平地。在图8中，我们演示了如何计算质量为M (kg)的汽车在速度为V (m/s)的情况下以角度θ(度)在道路上行驶所需的功率(Garcia-Valle和Lopes，2012 )，如下所示



其中P (W)是功率，FMI是质量系数，M (kg)是总质量，g (m/s2)是重力加速度，Crr是轮胎和路面之间的滚动阻力系数，ρ是空气密度(kg/m3)，A (m2)是车辆正面面积，Cd是气动阻力系数，VW是风速。我们不考虑加速前期，所以α为零。参数列于表1中。能量消耗应该通过功率P乘以持续时间来计算。

表 1 功率计算参数



**奖励设置**：定义奖励的基本概念是基于从当前位置到下一个位置的一步中的能量消耗，例如从图6所示的A到B。能量通过上一节提供的方法计算。我们将能量除以10000乘以-1。为了使训练过程中的总步数最小化，如果下一个位置是可到达的，我们给每个转换加-0.1。换句话说，采取任何可达步骤的回报r将是r = -0.1 -(能耗/ 10000)。如果下一个位置是不可到达的，比如湖泊或河流，r = -1和代理停留在相同的当前位置，并采取其他行动。如果下一个位置与目的位置的距离小于预定义位移的长度(图6中为750m)，那么采取这个动作的奖励r就变成r = +1 -(从当前位置到下一个位置的能耗/ 10000) -(从下一个位置到目的位置的能耗/ 10000)。注意这里+1出现在奖励中，因为这个动作的成功引导代理到达目的地。

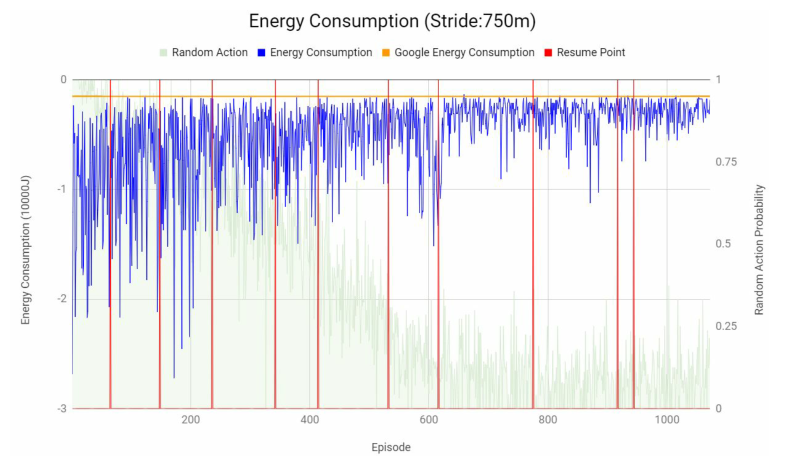


图9 学习代理的能量消耗。蓝线是消耗的能量，橙线是走谷歌地图推荐的路线所需的能量。绿线代表随机采取行动的概率，红线意味着我们从先前的训练模型中恢复训练

**电动汽车电池**：在实验中，电池性能不会影响训练过程。该电池能够携带总共50000瓦时的能量，这是电动汽车制造商特斯拉的标准产品。在电化学中，建议使用电池90%∾20%的充电状态(SOC)来延长电池寿命，我们在本例中也采用了这种方法。SOC通过当前能量和总能量之间的比率来计算。我们不会将电池退化纳入实验。作为训练过程的一部分，进一步的工作可以考虑电池性能的实际因素。对于这个实验，我们只演示在理想条件下电池需要消耗多少能量和充电多少次。

## 结果分析

代理正在从起始位置(地理编码:40.4682572，-86.9803475)和目的地(地理编码:40.445283，-86.948429)进行训练，步幅为750米。每集内超过64步的步骤将被视为失败。注意到在训练期间，谷歌地图api经常阻塞我们的服务器，我们被迫结束训练过程，并从中断的情节(红线)恢复模型。这个问题将导致空的重放缓冲区，在那里我们均匀地随机选择样本来计算损失并在学习期间更新权重。因此，我们将需要恢复模型，并开始随机选择动作，以重新填充重放缓冲区，并逐渐减少随机动作的比例。

图9中的蓝线显示了代理消耗的能量。代理能够在600集之后找到一种最小化能量消耗的方法。前600集的振荡是由Qnetwork提供的高度随机的动作(绿线)和不准确的Q值造成的。当不准确的Q值的损失被最小化时，振荡被减轻并且能量消耗变得更少和稳定。代理可以用随机动作实现的最小能量是1327(J ),对应的时间是659秒，其中Google map提供的路线的能量和时间是1489(J)和315秒。

## 结论

双DQN能够从奖励和经验中学习。将步幅长度减少到20米或50米可以提高精确度，并可能减少能量，但仍需要更多的实验来证实。我注意到你获得的精度越高，你需要的计算资源和时间就越多。源代码:https://github.com/Dungyichao

## 参考文献

1. Garcia-V alle, R., and Lopes, J. A. P . 2012. Electric vehicle integration into modern power networks. Springer Science & Business Media.
2. Tamar, A.; Wu, Y .; Thomas, G.; Levine, S.; and Abbeel, P .2016. V alue iteration networks. In Advances in Neural Information Processing Systems, 2146–2154.
3. V an Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In AAAI, 2094–2100.
4. Wang, X., and Gupta, A. 2016. Generative image modeling using style and structure adversarial networks. In European Conference on Computer Vision, 318–335. Springer.