**Real-Time Adversarial Attacks**

**Yuan Gong** , **Boyang Li** , **Christian Poellabauer** and **Yiyu Shi**

**Abstract**

常见对抗攻击都是静态输入，敌手可以观察整个原始样本，并在样本中任何一点添加扰动。

本文考虑当模型输入为stream时的实时对抗攻击，这种情况下敌手只能观察到过去的数据，且只能将干扰添加到未观察到的数据上。另外，本文是第一个对实时对抗攻击进行研究的。

**Introduction**

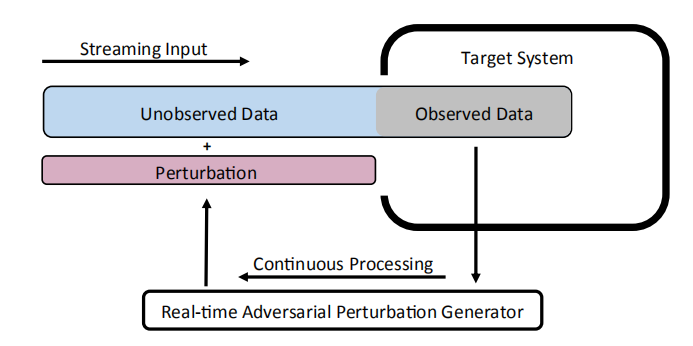
**1.情景**

实时情景下，敌手只能观察到过去的样本，只能在未来的样本上添加扰动（言外之意：不能改动过去的样本），但是目标模型的决策却是基于整个数据集的。

**2.实时对抗攻击难点在于：**

实时情景下，敌手面临着observation和action space的平衡问题。对input，敌手选择前面一部分作为observation观察数据，剩余的是action space用于添加扰动。极端情况下，observation为input全集时，虽然观察的最准确，但没有实施扰动的余地；当action space为input全集时，又因没有观察而无法优化扰动。

为解决这个trade-off，本文使用了深度**强化学习**框架，如下图，对抗扰动生成器连续地利用观察到的数据来优化针对未来数据的最佳扰动。（不太了解强化学习的话可以拉到最后先看一下基本原理）。



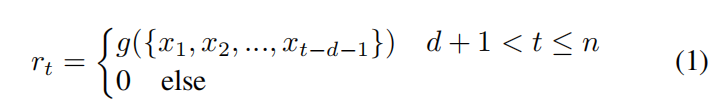
**Real-time Adversarial Attacks**

**1.问题描述（分两种理解形式）：**

**第一种理解：优化问题**

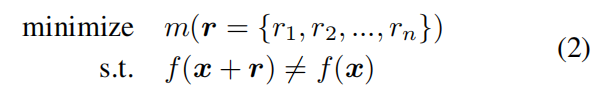
输入x = {x1, x2, x3, ..., xn}，f为将时间序列映射到对应label的分类器。

敌手目标是构建一个扰动生成器*g*(*·*)，*g*(*·*)连续观察数据{x1, x2, ..., xt}以近似未来在t+d+1时刻的最优扰动*rt*+*d*+1，其中d是处理数据或产生扰动引起的时延，也即：



（r是生成的扰动，第一行表示不断观察过去的数据，生成未来对应时刻的扰动；第二行指对过去的数据不进行扰动）

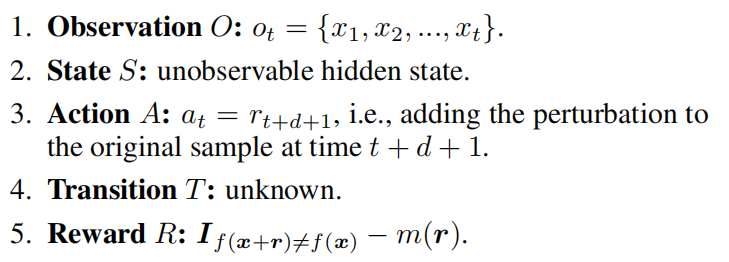
定义*m*(*·*)用于度量扰动的可感知性，对non-targeted对抗攻击，敌手目标是优化下式：



（最小化扰动的可感知性，限制条件是x加上扰动后分类为其他类即可）

**第二种理解：强化学习表示**

将扰动生成器看作agent，这个问题可以看作是**部分可观察的决策过程**问题，即该发生器连续观察数据流，对如何进行扰动作出一系列决策。问题可用<*O, S, A, T, R>*表示

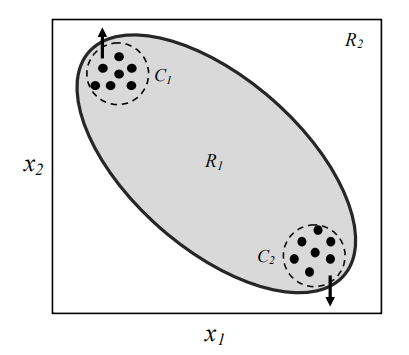


基于观察ot，敌手在t+d+1时刻执行一个动作at（增加扰动到原始样本），根据未知的转换规则T，这样会改变内部隐藏状态。生成器只能在最后得到reward。强化学习目标是学习最优策略*πg* : *at* = *g*(*ot*)以最大化reward的期望。

**2. 敌手面临的两个难点：**

（1）基于已观察到的数据预测未来的扰动。

解决：输入样本是遵循一些固定分布的，在样本之间往往存在依赖关系。如图，以低维为例子，敌手试图将原始样本（黑点）从决策区域R1移动到R2。若观察不到x1，敌手就不知道怎么在x2上进行扰动；但观察到x1后，敌手便知道在x2上应该加的扰动方向，如对C1簇施加向上的扰动，对C2簇施加向下的扰动。



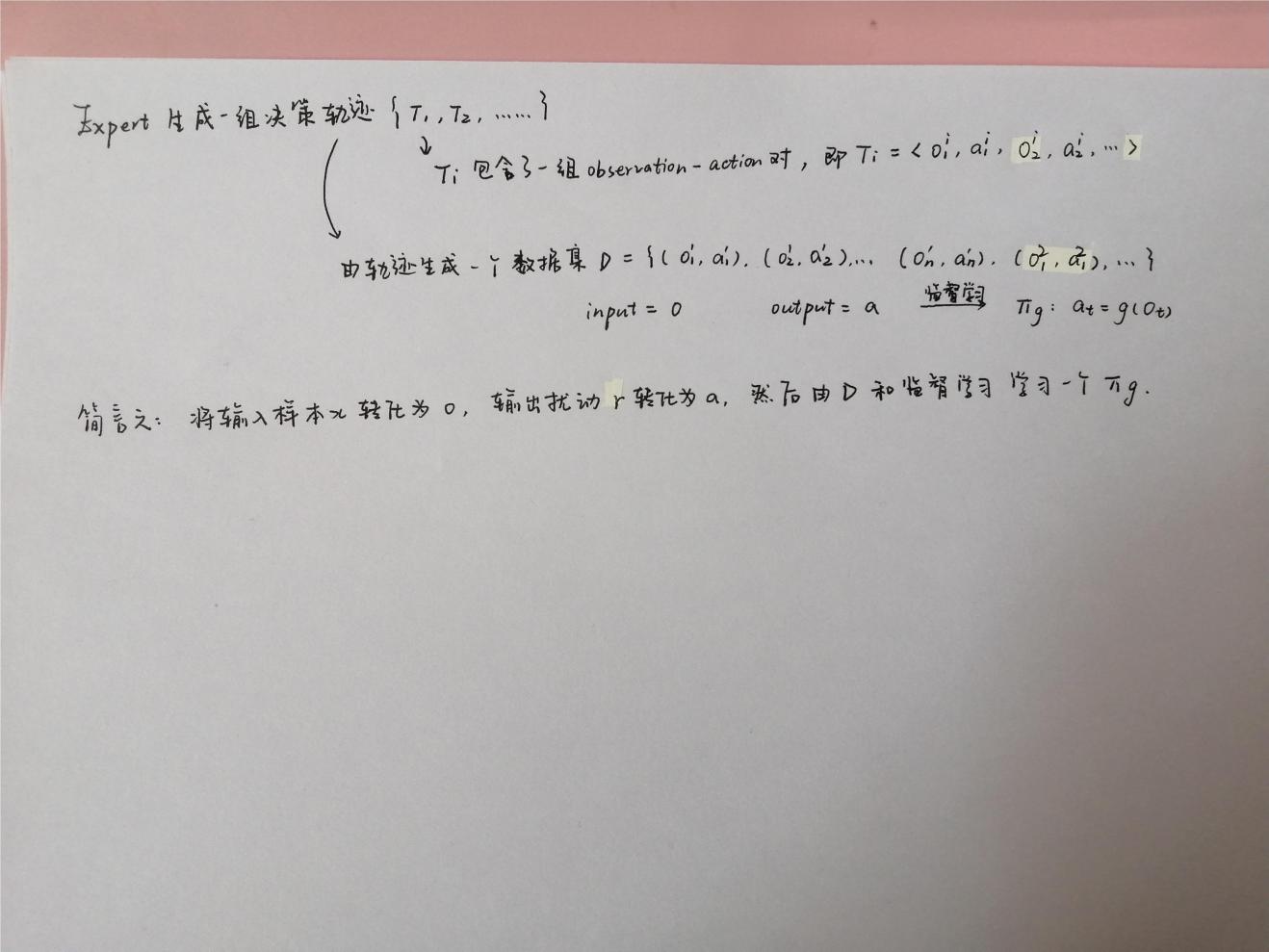
（2）稀疏rewards问题。

敌手只能在最后得到reward，而在任何时间点得到reward的估计很难。

解决：利用**模仿学习和行为克隆**，用**非实时的对抗生成算法**得到observation-action pairs的轨迹。

**3. 模仿学习是什么**

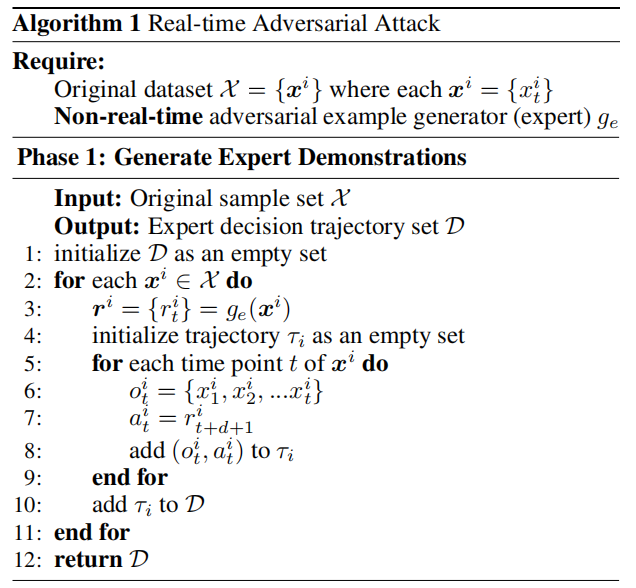
这是RL的一种技术，**模仿一个expert的行为**，学习到优化策略*πg。*

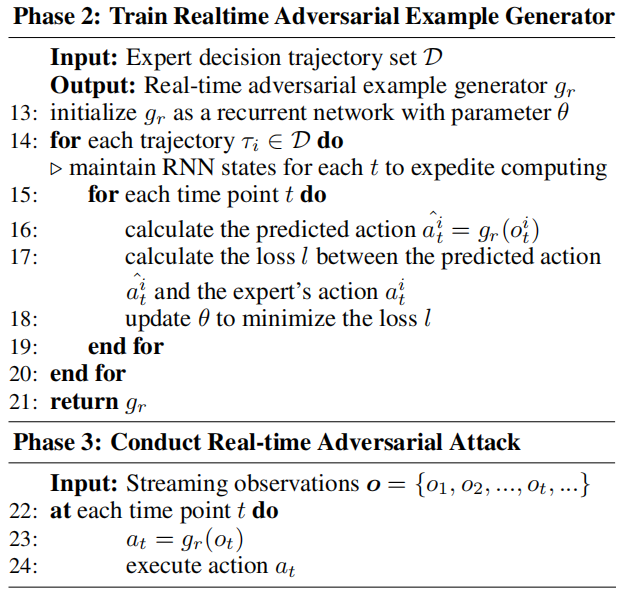


**4. 算法思想**

过程分三步：选取非实时的对抗样本生成方式作为expert，得到训练集D；将D作为输入，训练实时对抗样本生成器g；用g预测最佳扰动，实施攻击。

算法伪代码如下：

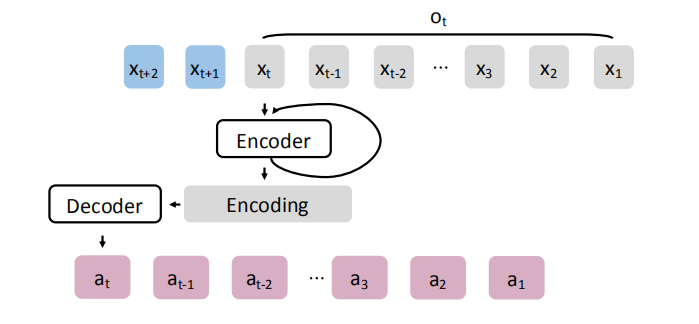




另：需要注意两点

（1）由于o是变长的，采用RNN作为网络的一部分。神经网络被分为encoder和decoder两部分，encoder是一个RNN，将变长映射为固定维度的encoding；接着decoder对action作出decision。[例如，2（1）图，encoder用于指出样本属于哪一个簇，接着decoder找到最优扰动。] 然后，计算预测action与真实action之间的误差，并使用标准的反向传播更新g。

（2）训练g的过程如下图。ot+1和ot只多出一个样本xt+1，这样RNN（encoder）就存在大量重复计算。 为减小计算开销，在将输入ot输入到g中得到at后，不将ot+1输入到g中，而是直接将xt+1输入到g中，得到g在at+1时的输出：



**实例：攻击一个语音命令识别系统**

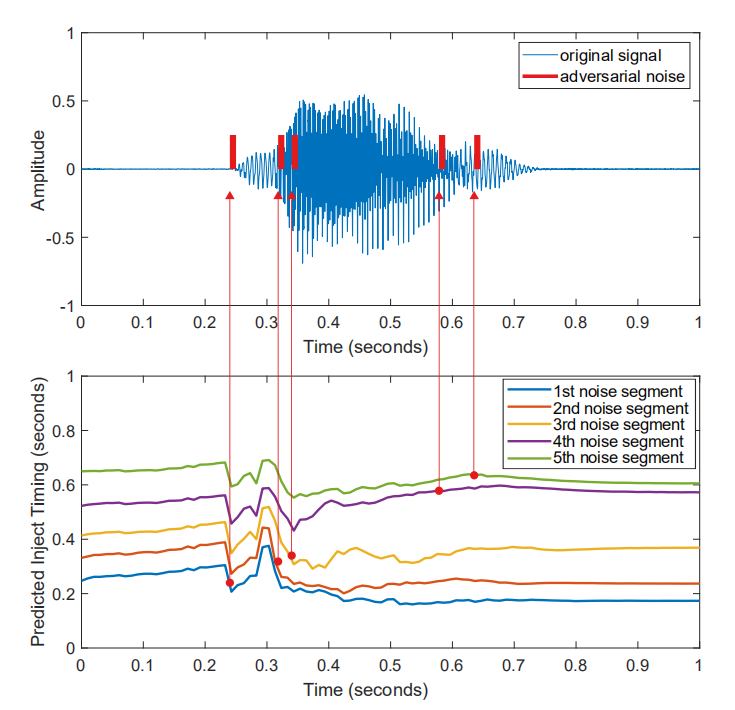
将前面的框架用到实例中，进行语音对抗攻击。

依照此前的工作Tensorflow example，以同样方式训练一个语音识别系统，唯一区别在于本文只使用80%数据作为训练集，剩下的20%作为测试集。所有语音样本长度为1秒（时长不够的补充到1秒），频率16Hz。模型可以识别出十个关键词（label）：yes,no,up,down,left,right,on,off,stop,go。

与传统非实时对抗攻击相比，实时攻击最大的优势在于可以对正在进行的会话（即实时人机交互）进行攻击。

鉴于实际情况，考虑半黑盒攻击，即敌手不知道模型内部细节，但问询次数不受限制，问询得到预测或置信度。不考虑信号衰减与失真。

扰动生成器g连续预测出5个0.01秒的最佳注入扰动时间，到时间或快到时间时注入扰动。如图所示，上图中红线是g预测出的扰动，下图中红点是最优扰动，g预测的可能不是基于全部观察的全局最优，但是部分观察的最佳优化：



**实验设置：**

g是一个深度神经网络DNN，输入是一个变长观察o，输出是一个五元组（即注入扰动的5个最佳时间点）。g每0.01秒更新一次。

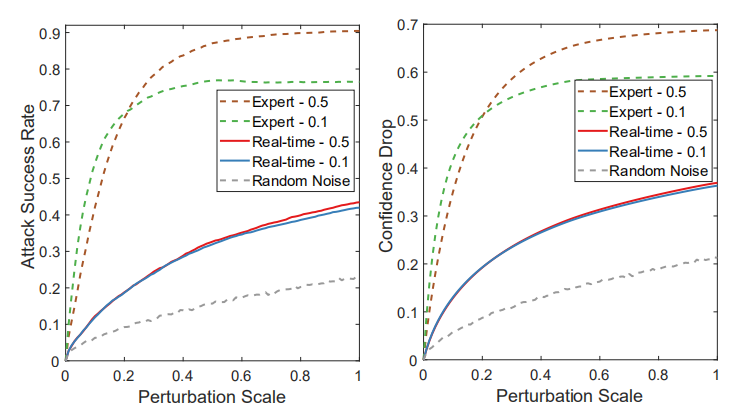
数据划分设置：

首先20%的数据作为测试集(test2)来评估攻击性能（所以目标模型和攻击模型都看不到它）。剩下80%的数据来训练目标语音识别模型；然后重用这个集合来开发攻击模型。其中，该集合的75%作攻击训练集，6.25%用于验证，18.75%用于测试(test1)（test1可以被目标模型看到，而攻击模型看不到）。

评估攻击效果的指标有两个：攻击成功率；攻击造成的对正确分类的confidence score drop。

**实验结果：**

1、在test1上测试了两个非实时experts的攻击性能和本文实时对抗模型g的攻击性能(0.1和0.5是振幅)：



（1）振幅为0.1时，g的攻击成功率达到43.5%。g的成功率基本在expert的30%-50%。

（2）振幅会影响expert的攻击准确率，振幅大时成功率更高（0.5）。振幅虽然对g也有影响，但影响小一些。这是g的一个优秀性能，因为现实中会存在信号衰减的情况，对g的影响会更小一些。

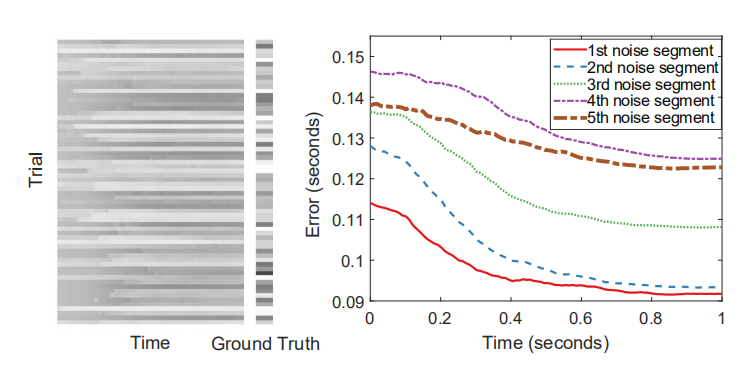
（3）进一步在test2上测试(成功率高达42.2%)，发现test1和2的结果没有显著差异，说明攻击模型g可以推广到目标模型没有看到的数据样本上。

2、左图：g对最佳扰动注入时间的估计（暗色表示较晚时间，亮表示较早时间），每行是一次攻击实验，共64次实验。

右图：平均预测误差随时间的变化。

观察：每次实验开始时，由于没有观察到数据，g对不同样本的先验概率是近似的。随着观察的数据越来越多，g的预测越来越接近事实。

结论：g的平均预测误差随着观察的增多而变小，g的性能更好（但观察到全部数据后，g的预测依然有误差，这也是本文需要改进的地方）。



**Future Work**

1.进一步提高g性能，如引入补救机制。

2.找到针对g的防御方式。

**附：强化学习基本原理**

强化学习不同于监督学习，主要表现在强化信号上。由环境提供的强化信号是对产生动作的好坏作一种评价，而不是告诉强化学习系统RLS如何去产生正确的动作。由于外部环境提供的信息很少，RLS必须靠自身的经历进行学习。具体地说：

智能体（Agent）以“试错”的方式进行学习，通过与环境进行交互获得的奖赏r指导行为，目标是使Agent获得最大的奖赏。如果Agent的某个行为策略a导致环境**正**的奖赏(强化信号)，那么Agent以后产生这个行为a的趋势便会加强。

