CGDTest：一种用于测试神经网络的约束梯度下降算法

Vineel Nagisetty[1] , **Laura Graves**[1] , **Guanting Pan**[1] 和 **Piyush Jha**[1] 以及 **Vijay Ganesh**[11]滑铁卢大学{vineel.nagisetty, laura.graves, g6pan, piyush.jha, vganesh}@uwaterloo.ca

arXiv:2304.01826v1 [cs.LG] 4 Apr 2023

摘要

本文提出了一种新的深度神经网络（DNN）测试算法，称为约束梯度下降（CGD）方法，并实现了名为 CGDTest 的工具，旨在揭示 DNN 中的安全性与鲁棒性问题，例如对抗鲁棒性和偏见。我们的 CGD 算法是一种梯度下降（GD）方法，其特点是用户还可以指定用于刻画所需输入类型的逻辑属性。这一功能使 CGDTest 区别于其他类似的深度神经网络测试工具，因为它允许用户通过指定逻辑约束来测试 DNN，不仅可以测试基于 ℓ_p ball 的对抗鲁棒性，更重要的是，还支持更丰富的属性，例如伪装和流量对抗约束，以及自然语言处理领域中的对抗鲁棒性。我们通过在视觉和自然语言处理领域开展广泛实验，展示了 CGDTest 的实用价值和能力，并将其与这些不同领域中的 32 种最先进的方法进行了比较。结果表明，在基于 ℓ_p ball 的对抗鲁棒性测试方面，CGDTest 优于最先进的测试工具；在测试其他对抗鲁棒性方面也显著优于其他工具，在某些情况下，PAR2 分数比次优工具提升了 1500% 以上。评估结果显示，在表达能力（即丰富的约束语言及相应的工具支持，可表达多种属性）、可扩展性（即可应用于参数量高达 1.38 亿的大型真实世界模型）和通用性（即可用于测试大量模型架构）方面，我们的 CGD 方法优于所比较的竞争方法。

1 引言

在过去十年中，深度神经网络（深度神经网络）已无处不在，应用场景从图像识别到安全关键系统不等 [Sarker, 2021]。与此同时，针对深度神经网络的各种攻击也层出不穷，包括对抗攻击、数据投毒攻击、成员推理攻击和模型反演攻仅举几例攻击：规避攻击等 [Huang 等人， 2020]。更广泛地说，机器学习社区普遍认识到，机器学习模型（尤其是深度神经网络）缺乏可靠性和安全性这一问题值得关注，并且需要得到有效解决 [VNNLIB， 2019]。

认识到上述问题后，研究人员提出了多种专为深度神经网络场景量身定制的测试、分析与验证（TAV）方法 [Huang 等， 2020]。**测试** 方法旨在寻找满足某些约束的输入，通常会采用梯度下降（GD）方法的变体 [Huang 等， 2020]。虽然测试方法具有良好的可扩展性，并适用于多种模型架构，但无法保证模型在安全性或鲁棒性属性方面的正确性。此外，这些方法往往不允许用户指定逻辑属性，而且其适用范围通常局限于一些众所周知的属性，例如基于球形区域的对抗鲁棒性 ℓ_p 。相比之下，**验证／程序分析** 方法通过符号化方式“捕获”模型，并通常使用求解器来寻找违反给定属性的输入，或保证深度神经网络遵循这些属性 [Katz 等， 2019]。遗憾的是，验证方法存在可扩展性和通用性不足的问题（即只能处理某些类型的架构），但表达能力往往很强。最后，**可达性** 方法定义一个允许输入区域，并将该区域沿网络传播，得到一个输出域，其中包含该输入区域内任意输入所能产生的所有可能输出 [Singh 等， 2019]。这些方法在可扩展性和通用性方面优于验证技术，但不及测试方法。（相关工作第5节详细介绍了每种方法。）

在许多工业场景中，模型开发者需要能够扩展到数百万个参数（可扩展性）、适用于多种架构（通用性），且表达能力足够强、允许用户指定各种感兴趣属性的测试/验证方法。因此，我们工作的重点是设计一种实用的深度神经网络测试工具，使其在可扩展性、通用性和表达能力方面表现出色。

问题陈述： 更确切地说，我们要探讨的研究问题是：能否创建一种测试方法，让模型开发者能够指定属性（无需将被测深度神经网络转换为验证速度缓慢的符号形式），同时又能

扩展到真实世界的模型，并且足够通用，能够处理各种各样的模型架构?

CGDAlgorithm 概述：为了解决上述研究问题，我们提出了一种名为约束梯度下降（CGD）的方法。该方法增强了梯度下降（GD）方法，使其能够指定逻辑约束，从而针对特定属性测试深度神经网络，而目前常用的大多数测试、验证或可达性工具尚不支持这些属性。具体而言，CGD 按照概率软逻辑中提出的方法，将一大类逻辑约束转换为可微损失函数（称为“约束损失”） [Kimmig 等， 2012]。随后，可以使用梯度下降算法最小化该约束损失。

1.1 我们的贡献：

1.**约束梯度下降（CGD）算法及其实现：**我们的主要贡献是约束梯度下降算法（以及我们称为 CGDTest 的工具）。该方法允许用户将逻辑约束指定为梯度下降算法损失函数的一部分，旨在生成满足用户指定逻辑约束的深度神经网络测试输入。我们的方法解决了模型开发者最关心的问题：1）测试工具能够扩展至工业级深度神经网络（**可扩展性**）；2）架构无关（**通用性**）；最重要的是，3）允许用户以逻辑方式指定所需的输入类型（**表达能力**）。CGD 可表达的约束类别比其他基于梯度的方法更广，因此基于 CGD 的方法能够应用于具有更复杂输入约束的场景。（见第 3 节。）

2. **展示 CGDTest 在 NLP 领域的可扩展性、通用性和表达能力：**为了展示 CGDTest 能够测试比视觉领域中 ℓ_p 基于球形约束的对抗属性更丰富的属性类别，我们选择自然语言处理（NLP）领域来研究对抗鲁棒性。针对视觉领域构建的现有深度神经网络测试、分析与验证工具，在 NLP 领域效果不佳，这是因为输入空间具有离散特性，且存在固有的语义和句法约束 [Morris 等人， 2020]。我们提出一种在 NLP 领域应用基于梯度的测试技术的方法，并在一个语义分类数据集上训练的目标模型上，将其与 5 种专为 NLP 领域设计的最先进对抗攻击工具进行比较。结果表明，在生成对抗样本方面，CGDTest 比竞争的最先进工具更有效、更高效。（见第4.3节）。

3. **展示 CGDTest 在视觉领域的可扩展性、通用性与表达能力：**我们报告了详细的实验，通过实验评估 CGDTest 的可扩展性和通用性，并将其与最先进的测试、验证和可达性方法进行比较。实验覆盖多个基准测试套件，其中的模型规模从 6000 个到 13800 万个参数不等。我们展示了，采用基于 ℓ_p 球的对抗样本定义时，CGDTest 工具的表现可与最先进的深度神经网络测试、分析与验证工具相媲美。此外，在对于伪装型和流量对抗样本，CGDTest 展现出显著优势，而其他方法的表现则差得多，这是因为它们无法扩展、无法指定属性，或对深度神经网络架构施加了限制。（见第 4.3 节。）

2 背景

2.1 对抗样本

对于给定的遭受攻击的深度神经网络，对抗样本是指与被正确分类的输入相似（根据适当的数学相似度度量），同时又会导致遭受攻击的深度神经网络将其错误分类的输入。通常，这类相似度度量是使用输入空间上的 ℓ_p 范数定义的。在阈值为 ϵ的 ℓ_p-范数相似度度量下，若示例 x'满足以下条件，则称其为对抗样本：

$$(\|x - x'\|_p \leq \epsilon) \wedge (argmax(M(x)) \neq argmax(M(x')))$$

研究人员开发了多种测试和验证方法，用于寻找对抗样本 [Huang 等,2018]。此外，每年还会举办竞赛，例如 VNN-COMP [Bak 等，2021], 旨在评估深度神经网络测试（又称不完备求解器）和验证（又称完备求解器）工具在针对模型和对抗鲁棒性属性的基准测试集上的表现。

2.2 个体公平性

Dwork等人将个体公平性概括为确保“相似的个体受到相似的对待” [Dwork等人,2012]。换句话说，对于特定任务而言，任何两个相似的个体都应被相似地分类。形式化地说，给定一个用于捕捉两个个体 x 和 x'之间相似性的（通常较为复杂的）距离函数 d，如果模型 M 违反以下约束，则可以认为该模型存在不公平性：

$$(d(x, x') \leq \epsilon) \wedge (argmax(M(x)) \neq argmax(M(x')))$$

在实践中，选择合适的距离函数可能很有挑战性，并且可能需要领域专家的意见 [Dwork等人。，2012]。或者，如 [Yurochkin 等人。,2019], 所示，这种距离可以从数据中学习。在本研究中，距离按如下方式计算： $d_x^2(x, x') = \langle x - x', P(x - x')\rangle$，其中 P定义了一个投影矩阵，该矩阵与某个敏感子空间正交。

2.3 将约束转换为损失

我们所采用的约束语言由项之间比较关系的布尔组合构成，其中每个项要么是常数，要么是定义在深度神经网络的输入、中间层值和输出上的可微函数。为了将约束 φ 转换为可微损失函数，我们进行简单解析，并使用 *Lukasiewicz t*范数 及其对应的余范数来映射布尔联结词和比较关系。这种转换旨在确保在极值处（“完全为真”或“完全为假”）精确，并为中间值提供一致的映射，从而使我们能够计算平滑梯度信息。下面给出了这些松弛的示例。

算法 1CGD 算法

要求： 深度神经网络 M、标签 y、权重 α、约束 φ
确保： 深度神经网络输入 x，使得 $(argmax(M(x)) = y) \wedge \varphi(x)$

```
1:  φ′ =约束损失(φ)
2:  初始化 x
3:  当 True 时
4:      如果(argmax(M(x)) = y) AND φ′(x) = 0 则
5:          跳出
6:      结束 if
7:      l1 = φ′(x) //计算约束损失
8:      l2 = 损失(M(x) y)
9:      ∇x1 = ∂l1/∂x
10:     ∇x2 = ∂l2/∂x
11:     x = x − α ∗ (∇x1 + ∇x2)
12: 结束循环
13: 返回 x
```

1. $L(\varphi' \wedge \varphi'') := L(\varphi') + L(\varphi'')$,
2. $L(\varphi' \vee \varphi'') := L(\varphi') * L(\varphi'')$。

这会产生一个几乎处处可微且连续的函数。我们将此函数称为“约束损失”。附录 A 列出了所用术语的完整清单，并提供了有关约束转换过程的更多信息。

3 CGD 算法

本节概述我们的约束梯度下降（CGD）方法（参见算法 1），该方法针对第 2.3 节所述的约束损失执行梯度下降。CGD 算法的输入是一个被测深度神经网络 M、一个标签 y、一个权重 α，以及一个用于编码感兴趣的某种深度神经网络属性的约束 φ。算法的输出是一个满足约束 φ 且符合 $(argmax(M(x)) = y)$ 的深度神经网络输入 x。

在 CGD 算法中，我们首先解析用户指定的约束 φ，并通过一小段代码将其转换为约束损失函数 φ'（第 1 行，也请参阅附录 A）。该约束损失函数经过构造，只有在所有约束均满足时才会计算为 0。我们将输入 x 初始化为一个随机向量，并将其作为深度神经网络 M 的输入，其大小与 M 的输入大小相同（第 2 行）。随后，我们进入一个循环，直到约束得到满足（即 $\varphi'(x) = 0$），且模型将 x 分类为预期的标签 y 时才终止（第 4–6 行）。如果 x 满足这些条件，我们就跳出循环并将其作为解返回（第 13 行）。否则，我们分别计算约束损失（第 7 行）和分类损失（第 8 行），然后计算这两种损失相对于 x 的梯度（第 9 和第 10 行）。最后，我们合并两个梯度值，并稍微调整 x（使用较小的α 值），使其朝着损失降低的方向移动（第 11 行）。此过程会持续进行，直到 x 满足这两个条件，或工具达到超时时限。

直观地，我们可以将约束和误分类损失的梯度（算法 1 中第 9 行和第 10 行）视为具有不同的损失景观，CGD 算法沿着这些景观遍历以寻找全局最小值。如果这些损失景观不相互-如果不存在同时属于这两个损失曲面的点，或不存在同时作为两个损失曲面全局最小值的点，那么就没有可能的解。然而，即使存在这样的点，CGD 也不能保证找到解，这与其他梯度下降算法一致。这是因为一个或两个损失函数可能是非凸的，因此该算法可能会陷入局部最小值。

正因如此，我们会加入少量经过精心设计的噪声（类似于 [Yu 等人， 2020]），以帮助避免局部最小值，并让我们更快地找到结果。我们也可能希望找到大体上满足约束的输入。在这种情况下，我们可以添加一个较小的界限 δ，并更新第 4 行中的条件语句，以检查约束是否基本得到满足（即 $\varphi'(x) < \delta$）。此外，CGD 不会穷举搜索整个空间，因此我们会限制其运行的迭代次数（或时间），并在未能在该限制内找到解决方案时返回“未完成”。如果对每个梯度值使用单独的权重 α_1, α_2，我们可能会得到更好的结果。这样可以控制每个梯度值的影响，从而更好地改变x。此外，我们还会初始化一批输入，而不是只初始化一个输入，并确保随机化的输入在初始化步骤中遵守输入约束。

请注意，这种梯度下降方法与快速梯度符号法（FGSM） [Goodfel-*low* 等人， 2014]存在一些相似之处。通过使扰动的符号与梯度的符号保持一致，FGSM 根据对抗样本的定义，沿着损失增加的方向移动，以搜索对抗样本。然而，FGSM 不考虑约束，因此只能在视觉领域测试对抗鲁棒性。相比之下，CGD 通过朝着误分类损失与约束损失之和减小的方向移动，搜索满足约束的输入。此外，除了视觉领域之外，CGDTest 还可以为自然语言处理和表格领域构造输入。

3.1 CGDTest 工具实现细节。

我们使用 TensorFlow 2.2 实现了 CGDTest 工具。CGDTest 接收使用第 2 节所述语言给出的约束作为输入。随后，这些约束通过第 2 节所述的简单解析器转换为约束损失。然后，将该约束损失与模型 M 和标签 y一同输入 CGDTest，以寻找满足约束的样本。由于 CGD 可能陷入无限循环，如果达到最长时间限制，我们可以终止 CGDTest，并返回“未完成”。

4 CGDTest 工具评估

4.1 实验设置：硬件与软件

我们的所有实验均在一台配备四核 Intel i5-4300U（CPU @1.9 GHz）处理器、16 GB 随机存取存储器并运行 Ubuntu 20.04 的机器上完成。为运行这些实验，我们使用 Python 3.8、Tensorflow 和 Keras 实现了一个评估框架，并将最先进的深度神经网络 TAV 工具应用于视觉和自然语言处理领域进行比较。有关视觉领域对抗鲁棒性测试的详细信息，请参阅第4.3节；有关自然语言处理领域对抗鲁棒性测试的详细信息，请参阅第4.4节；有关测试的详细信息，请参阅第4.5节

TAV工具	ℓ_p ball PAR2（秒）	伪装的 PAR2（秒）	Flow PAR2（秒）
CGDTest	18,476.36	**143,010.88**	**45,170.87**
AutoAttack	**17,459.21**	*	*
B&B	18,259.62	*	*
C&W	19,713.19	*	*
BIM	23,120.65	*	*
FGSM	43,220.72	*	*
Crown-IBP	585,099.22	606,399.84	*
ERAN	600,040.70	647,970.45	*
nnenum	604,635.81	642,475.34	*
MIP Verify	653,736.16	703,006.80	*
DLFuzz	707,066.27	*	*
Marabou	703,057.30	728,376.78	*
Fuzz	734,821.58	780,000.00	780,000.00
遗传算法	703,211.67	735,509.32	761,186.62

表 1：在来自视觉领域的 VNN-LIB 基准测试集上进行的实验结果，汇总了所有基准测试套件。* 表示该工具无法针对给定类型的对抗样本进行测试。

这些工具以及来自 VNN-COMP 2021年和 VNN-COMP 2022年的基准测试集。

结果衡量：为了比较各种工具，我们使用攻击成功率以及 PAR2 分数。PAR2 分数是 SAT/SMT/VNNCOMP 求解器社区常用的一项指标，用于在一组基准测试集上有效比较各种求解器。PAR2 的计算方式如下：如果某个工具在给定的时间限制内得出结果（输入违反逻辑规范，或满足属性），我们就记录得出该结果所用的时间（以秒为单位）。反之，如果某个工具未能在规定时间内得出结果，则计入 2×时间限制的惩罚。PAR2 分数越低越好，因为这表示与 PAR2 分数更高的方法相比，该工具平均能更快地找到违规输入（或验证被测深度神经网络满足属性）。

4.2 基准测试集

所使用的基准测试集如下：

1.**VNN-LIB 和 VNN-COMP 2021/2022**：第一套基准测试集包含来自 VNN-LIB 和 VNNCOMP 2021/2022 的模型。这些国际竞赛旨在比较最先进的深度神经网络验证和测试工具 [VNNLIB，2019]。关于 VNN-LIB 基准测试集的详细介绍，见 [Guidotti 等人， 2020]，；关于 VNNCOMP 2021 和 2022 的介绍，见 [Bak 等人，2021；M¨uller等人， 2022]。

2. **TweetEval**：该套件包含一个语义分类模型，使用 BERT 模型的最后几层。该模型使用 TweetEval 数据集进行微调；该数据集由一组推文组成，任务是判断给定推文是否包含冒犯性内容。整个 BERT 模型拥有超过 1.1 亿个参数，而我们的模型使用约 100,000 个参数（见第 4.4 节）。

4.3 视觉领域的对抗鲁棒性（VNN-LIB）

在对视觉领域中各类深度神经网络测试、分析与验证工具进行广泛而深入的调研后，我们选取了 VNN-LIB 基准测试中的 [VNNLIB、 2019]，快速梯度符号法（FGSM） [Goodfellow 等人、 2014]，基本迭代法（BIM） [Kurakin 等人、 2016]，Carlini–Wagner 攻击（C&W） [Carlini 和 Wagner、 2017], Brendel 和 Bethge 攻击（B&B） [Brendel 等人，以及 2019], AutoAttack [Croce 和 Hein, 这些属于梯度下降方法；还选取了模糊测试方法类别中的 DLFuzz [Kunhimangalam 等人和 Random Fuzz，以及基于进化算法的方法类别中的遗传算法（GA）。在现有的验证／程序分析方法中，我们选取了 Marabou [Katz 等人、 2019], Crown-IBP [Zhang 等人、2019], nnenum [Bak 和 2021], MIP Verify [Tjeng 等人的工具。对于基于可达性分析的方法，我们选取了 ERAN [Singh 等人 2019]。

选择这些工具，大多是因为它们在 VNN-COMP 2020 竞赛中表现出色 [VNNLIB,2019]，或为了确保评估的完整性（例如遗传算法）。（在第4.5节中，我们还将比较 CGDTest 与 VNN-COMP 2021 和 VNN-COMP 2022 的工具及基准测试集。）

在实验评估中，我们选择了一组基准测试套件，其中包含本地训练的深度神经网络模型，以及从 VNN-COMP 网站 [VNNLIB，2019] 获取的模型。有关这些基准测试的更多详情，请参见第4.2节。为了比较这些工具的表达能力，我们使用约束来定义三种对抗样本（更多信息请参见附录 B）：

1. ℓ_p **基于球的对抗约束**：此约束采用基于 ℓ_p 球的对抗样本定义，使用 L_∞ 距离， ϵ 值为 0.2。

2. **伪装对抗约束** [1]：与基于 ℓ_p 球的对抗约束不同，伪装对抗约束要求原始标签 y 是置信度第二高的标签。这里，x' 在某种意义上仍与 x 相似，因为 M 对 y仍具有较高的置信度（因为它是置信度第二高的标签）。在先前采用基于 ℓ_p 球的对抗约束的定义中，我们要求 x' 的新标签可以是除 y之外的任何标签；而在这里，我们要求 y 是第二高的标签，且不附加基于 ℓ_p 球的约束。这些样本之所以有趣，是因为它们刻画了一大类与给定输入相似、但存在细微差异，且会被归入前二的样本（例如，试图欺骗人脸识别系统的伪装恶意行为者）。

3. **流对抗约束：** 此类约束利用 [Xiao等人所提出的空间变换（流） 2018] 来寻找对抗样本。更具体地说，给定一个深度神经网络模型 M、标签 y、原始输入

[1]附录B.2中给出了示例，以说明此类约束的相关性

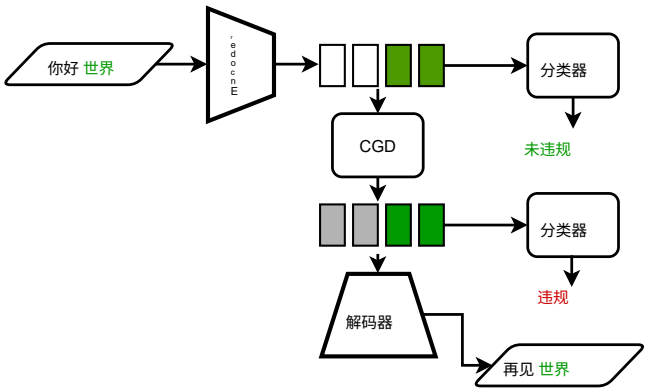

图1：在自然语言处理领域应用 CGD 的过程。

x 和新输入 x'，这些约束检查 $argmax(M(x')) \neq y$，同时 x' 和 x 具有由流方程（ [Xiao 等， 2018] 中的公式 1）引起的最小局部失真。当空间变换可用于欺骗深度神经网络时（例如，路障图像在某些角度下可能具有对抗性，因为它在遭受攻击的深度神经网络看来不像路障），这类对抗约束就很有意义。

我们创建了一个框架，为评估中的每个工具提供以下输入：一个深度神经网络模型 M、一个从 x 到 M 的输入，以及一项逻辑规范 φ；并检查工具能否在规定的 600 秒内给出结果。对于输入，我们从目标模型所使用的训练数据集的测试集中随机选取了 25 个样本。我们设置框架使用这些三元组运行所有工具，并记录每种工具生成的成功对抗样本数量。

实验结果： 表1展示了各项约束下所有基准测试套件汇总后的 PAR2 分数结果。

总体而言，在伪装对抗约束和流对抗约束下，CGDTest 在所有工具中取得了最低（即最佳）的 PAR2 分数，分别比次优工具提升了 309.13% 和超过 1500%（见表1）。此外，在 ℓ_p球形对抗约束下，CGDTest 的表现与取得最佳分数的 AutoAttack 工具相当。在 ℓ_p 球形对抗约束实验中，B&B 和 AutoAttack 的表现最佳，因为它们能够绕过对抗训练，因此在基准测试中经过对抗训练的深度神经网络模型上，其表现显著优于其他方法。即便如此，CGDTest 的表现仍与 AutoAttack 相当（见表1）。

正如我们的实验所示，大多数其他工具无法在另外两种约束下找到对抗样本。虽然观察到测试工具在基于 ℓ_p 球的对抗实验中表现良好，但它们无法构造满足另外两种约束的输入。相比之下，验证／程序分析工具和可达性工具能够针对基于 ℓ_p 球的约束和伪装约束进行测试，并在规模较小的 VNN-LIB 套件上取得了良好的结果。然而，它们无法扩展到更大的模型，并且

TAV工具	PAR2（秒）	ASR
CGDTest	**32,604.40**	73%
PSO	33,343.85	**74%**
HotFlip	41,440.28	67%
Pruthi	44,494.51	64%
TextFooler	51,617.14	59%
deepwordbug	55,221.63	56%

表 2：自然语言处理实验结果。攻击成功率（ASR）表示攻击成功率。

不支持流约束实验。此外，CGDTest 的性能明显优于其他基于 GD 的方法（例如快速梯度符号法（FGSM）），这主要归功于 CGD 对规范的感知能力，其中基于梯度的搜索使我们能够满足约束。最后，模糊测试和遗传算法工具在全部三种约束下均持续表现出较差的 PAR2 结果。

4.4 自然语言处理领域的对抗鲁棒性

为了让 CGDTest（以及其他梯度下降工具）能够测试自然语言处理模型[2]，我们提出以下流程（参见图 1）。首先，为确保扰动保留原始输入的句法和语义，我们添加约束，使输入中只有特定部分会被修改。具体而言，我们设计的约束仅用于扰动命名实体。命名实体是语言学中的专有名词（即人物、地点等的名称）。在许多情况下，扰动命名实体不会影响文本的句法或语义 [Mayer 等人， 2020]。命名实体识别是一项流行的自然语言处理任务，当前最先进的自然语言处理模型能够很好地完成该任务。作为预处理步骤，我们识别输入中的所有命名实体，并添加约束，要求对抗输入与原始输入之间仅在非命名实体特征子集上存在最小变化。随后，我们使用编码器模型（Enc）将离散文本输入投影到连续域。在此，我们采用定长输入，并在每个输入末尾添加额外填充。这样做是为了自动识别连续嵌入区域中对应于输入的部分，从而确定命名实体的位置。接着，我们应用基于梯度的 CGD 攻击，确保仅扰动命名实体区域，再使用解码器模型（Dec）将生成的对抗样本转换回离散域。请注意，我们要求 $Dec(Enc(x)) = x$，以确保离散文本与连续值之间的转换（反之亦然）不会导致输入发生任何修改。实际上，这一点并非总能成立，因此我们会舍弃所有在非命名实体区域与原始输入不相似的对抗输入。

为了评估 CGDTest 在这一领域的表现，我们使用 TweetEval 基准测试集，将其与通过 TextAttack框架实现的 5 种最先进的自然语言处理对抗攻击工具进行了比较。目标模型经过训练，能够利用输入的连续编码，对给定推文是否包含冒犯性内容进行语义分类。我们抽取了 100 个输入，并要求所有工具寻找对抗样本，随后比较它们在该测试集上的 PAR2 分数。

[2]请注意，很遗憾，参与 VNNCOMP 的工具尚未设置为接受自然语言处理输入属性。

表3：VNN-COMP 2021年结果

TAV工具	PAR2（秒）
α,β-CROWN	7,373
VeriNet	8,210
ERAN	9,770
oval	10,924
CGDTest	**11,009**
Marabou	12,851
Debona	14,222
venus2	15,836
Nnenum	16,095
nnv	21,352
NV.jl	21,985
RPM	22,372
DNNF	22,849

表 4：VNN-COMP 2022年结果

排名	TAV工具	得分
1	α,β-CROWN	1274.9
2	MN-BAB	1017.5
3	VeriNet	892.4
4	nnenum	534.0
5	CGDTest	**408.4**
6	Peregrinn	399.0
7	Marabou	372.2
8	Debona	222.9
9	FastBATLLNN	100.0
10	VeraPak	98.2
11	AveriNN	29.1

实验结果：表 2 展示了本次实验的结果。再一次，CGDTest 表现最佳，其 PAR2 分数比次优工具（pso）高出 2.3%。需要注意的是，pso 成功生成的对抗样本略多（74%），而 CGDTest 为 73%。不过，我们的工具生成对抗样本的速度更快，因此 PAR2 分数更高，尽管它并非专为这一领域设计。

4.5 VNN-COMP 2021年和2022年

我们没有正式参加 VNN-COMP 2021，但在竞赛组织者提供的 VNN-COMP 2021 基准测试和 AWS 工具实例上，对 CGDTest 进行了大量实验比较。参加 VNN-COMP 2021 的工具包括：Marabou [Katz 等人，2019], VeriNet [Henriksen 等人，2021], ERAN [M¨uller 等人，2020], α,β-CROWN [Zhang* 等人，2022],DNNF [Shriver 等人，2021], NNV [Tran 等人，2020],OVAL [De Palma 等人，2021], RPM [Vincent 和 Schwager，2020], NV.jl [Liu 等人，2019], Venus [Kouvaros 和 Lomuscio，2021], Debona [Brix 和 Noll，2020], 以及 nnenum [Bak，2021]。在支持的 2021 年基准测试中，CGDTest 总体排名第五；在 acasxu 基准测试中排名第二，在 marabou-cifar10 和 verivital 基准测试中排名第三。总体结果列于表 3。

我们确实向 VNN-COMP 2022 提交了 CGDTest，在 11 个参赛工具中总体排名第五。更重要的是，CGDTest 在三个基准测试类别（cifar100-tinyimagenet-resnet、cifar-biasfield 和 sri-resnet-a）中排名第二。请注意，CGDTest 是参加竞赛的唯一一个不完整工具。对于基准测试中的每个实例，组织者根据工具的准确率和相对耗时进行评分。最终得分为每个基准测试的标准化得分之和。表 4 列出了所有参赛工具的最终得分。有关评分的更多详情，请参阅 VNN-COMP 2022 报告第 2 节 [M¨uller 等人，2022]。为参加竞赛，我们使用 Python 3.8 和 Pytorch 1.11.0 实现了工具。为确保结果具有可比性，组织者使用 Amazon Web Services（AWS）在成本相同的硬件上评估所有工具。我们使用获准的实例之一 m5.16xlarge（配备 64 个虚拟 CPU 和 256 GB 随机存取存储器）运行工具。更多详情请参阅 VNN-COMP 2022 报告 [M¨uller 等人，2022]。

参加 VNN-COMP 2022 的其他工具包括：$\alpha\beta$-CROWN [Zhang* 等人、2022], AveriNN [Prabhakar 和 Rahimi Afzal、2019], Debona [Brix 和 Noll、2020], FastBATLLNN [Ferlez 等人、2022], Marabou [Katz 等人、2019], MN-BAB [Ferrari 等人、2022], nnenum [Bak、2021], PeregriNN [Khedr 等人、2021], VeraPak [Smith 等人，以及 2021],VeriNet [Henriksen 等人 2021]。

5 相关工作

深度神经网络测试方法：测试方法可分为三大类：模糊测试、遗传算法（GA）和梯度下降方法。梯度下降方法采用某种形式的梯度下降，尝试寻找满足特定属性的样本。示例包括快速梯度符号法（FGSM）[Goodfellow 等人、2014], Carlini & Wagner（C&W）攻击 [Goodfellow 等人，以及 2014], Brendel & Bethge 攻击 [Brendel 等人 2019]。这些方法效率高、泛化能力强，而且只需访问待测模型（有时还需要一个待修改的输入）。但这也意味着它们在表达能力和保证方面有所欠缺。相比之下，CGDTest 具有规范感知能力，因此能够生成满足复杂约束的输入。另一个近期工具是 DLFuzz [Kunhimangalam 等人 2013], ，它利用神经元覆盖率统计来引导模糊测试朝着提高神经元覆盖率的方向进行，以寻找对抗样本。DLFuzz 难以扩展至工业规模网络，并且仅限于搜索对抗样本。相比之下，CGDTest 在可扩展性、表达能力和通用性方面表现出色。请注意，通过 CGD 可表达的约束类别远大于其他基于梯度的方法所能表达的约束类别，因此基于 CGD 的方法能够应用于输入约束更复杂的情形。

验证/分析方法：这些方法以符号形式编码待测模型，并将属性表示为公式或约束，旨在提供保证证书（即不存在违反属性的情况）或生成反例。有多种工具，例如 MIP Verify [Tjeng 等人、2017], ，它们对深度神经网络进行编码并且

将这些性质表示为混合整数线性规划问题，并使用相应的求解器证明或证伪是否存在违反性质的示例。它们通常将深度神经网络的激活函数和架构限制为分段线性深度神经网络，而 CGDTest 并非如此。另一方面，像 Marabou [Katz等这样的工具是单纯形法，可以处理任意分段线性激活函数。与这些工具相比，CGDTest 效率更高，且可扩展至大型工业规模的深度神经网络。此外，CGDTest 通过使用梯度下降，在牺牲保证的同时也具有很高的通用性；在表达能力方面，它与验证工具相当。

可达性方法： 这些方法定义一个允许输入区域，并将该区域传播通过网络，从而得到一个输出域，其中包含给定任一输入值时所有可能可达的输出。通过这种方法，可以将输入区域限制为示例周围的某种形状（例如，扰动界限），并验证可达输出不包含错误分类，以此验证鲁棒性。这类工具包括 ERAN [M¨uller 等， 2020] 以及 NNV [Tran 等，2020]。遗憾的是，与 CGDTest 相比，这些方法扩展性较差，通用性也较低。

基于约束损失的学习： 据我们所知， [Kimmig 等 2012] 最早提出了概率软逻辑，它允许将逻辑约束编码为损失函数；这些损失函数大多是连续的，并使用求解器推断最可能解释（MPE）。 [Fis-cher等 2019] 随后推出了 DL2，这是一种可将某些逻辑约束转换为几乎处处可微的损失函数的工具。这些工具使用由此得到的约束损失函数训练深度神经网络，通过更新其参数来最小化约束损失，从而减少约束违反。相比之下，我们的 CGD 方法与上述方法有以下不同：首先，我们将约束损失与一种改进的基于 GD 的方法相结合（具体而言，是 FGSM [Goodfel-low 等 2014]）。此外，我们的方法旨在生成满足特定约束的深度神经网络输入，而 DL2 则使用基于约束损失的 GD 方法训练深度神经网络。

自然语言处理领域中的深度神经网络对抗攻击： 与视觉领域不同，基于梯度的对抗攻击通常无法在自然语言处理领域中发挥作用，因为输入是离散的，并且具有固有的句法和语义限制。此外，研究表明，即使是业界使用的最先进的语言模型，也容易受到对抗攻击。目前，研究人员已经开发出专用于该领域的深度神经网络测试、分析与验证工具，这些工具使用组合搜索来寻找符合所需约束的对抗输入。大多数搜索过程要么修改输入的部分内容（例如翻转某个字符），要么将输入的特定部分替换为相似的词。TextAttack [Morris 等， 2020] 是一个广受欢迎的框架，实现了多种最先进的自然语言处理对抗攻击。我们将其作为对比对象，并展示CGDTest的性能优于这些攻击。也就是说，CGDTest是一个非常通用的工具，但面对专门设计的、特定于领域的最先进测试工具时，它仍然能够达到同等或更好的性能。

个体公平性测试： [Yurochkin 等人， 2019] 将个体公平性测试表述为模型对敏感输入扰动的鲁棒性。然而，这项工作关注的是训练公平模型，而不是对其进行测试。相比之下， [Maity 等人的工作 2021] 提出了一种方法，使用基于梯度的搜索来扰动输入的敏感特征，从而寻找不公平点。相比之下，我们的工具是一种更通用的方法，可用于测试各种约束——包括这一版本的个体公平性。

CGD 与 PGD： 投影梯度下降（PGD） [Madry等人， 2017] 在每一步沿负梯度方向移动，从而最小化损失函数，然后“投影”到约束所定义的可行集上。需要注意的是，该投影是在梯度下降完成后执行的。相比之下，CGD 将约束集合转换为可微形式，同时计算约束损失和误分类损失，并直接应用梯度下降。

6 结论与未来工作

为解决与深度神经网络相关的可靠性和安全性问题，我们提出了一种名为约束梯度下降（CGD）方法的新型测试算法。该方法是一种梯度下降方法，其损失函数会将用户指定的逻辑约束纳入考虑。通过使用基于 ℓ_p 球的对抗样本以及流约束等其他更丰富类别的对抗样本，对 CGDTest 与另外 32 种最先进方法进行广泛的实证评估，我们表明，在视觉领域，CGDTest 的 PAR2 分数最高比次优方法提升了 1500%。我们还展示了将 CGDTest 应用于自然语言处理领域时的有效性：通过将输入投影到连续嵌入空间，并添加约束以遵循自然语言处理中固有的句法和语义限制。在这一设置下，将 CGDTest 与 5 种最先进的自然语言处理 TAV工具进行比较的实验表明，与下一种最优的专用测试方法相比，我们的方法使 PAR2 分数提升了 2.2%。此外，在 VNN-COMP 2022 中，在 11 个参赛工具中我们排名第五，尽管 CGDTest 是比赛中唯一的不完备工具。

我们开展的大量实验展示了我们的工具 CGDTest 相较于先进 TAV 方法的优势：它能够在包含超过 1.38 亿个参数的大型真实世界模型上良好扩展（可扩展性），允许用户在多个领域中指定种类广泛的深度神经网络属性（可表达性），并且不限制其可分析的深度神经网络类型（通用性）。与之比较的其他所有工具至少在其中一个方面表现更差。例如，在 VNNCOMP 竞赛中占据优势的验证工具无法处理流约束或自然语言处理领域，而且由于将许多常见的深度神经网络架构转换为符号形式存在困难，也无法处理这些架构。此外，模糊测试或遗传算法工具主要针对视觉领域中的对抗输入进行设计。同样，自然语言处理测试工具仅适用于自然语言处理模型。未来，我们计划将该工具扩展到其他领域，例如安全关键场景，并将其用作完备验证工具的一个组件。

参考文献

[Bak等人， 2021] Stanley Bak、Changliu Liu 和 Taylor Johnson。《第二届神经网络验证竞赛（ VNN-COMP 2021）：总结与结果》，2021年。

[Bak， 2021] Stanley Bak。nnenum：通过优化抽象细化验证 ReLU 神经网络。载于*NASA* 形式化方法研讨会，第 19–36 页。Springer，2021年。

[Brendel等， 2019] Wieland Brendel、Jonas Rauber、Matthias K¨ummerer、Ivan Ustyuzhaninov 和 Matthias Bethge。准确、可靠且快速的鲁棒性评估。*NeurIPS*，32：12861–12871，2019年。[Brix 和 Noll， 2020] Christopher Brix 和 Thomas Noll。Debona：用于收紧界限并加快对抗鲁棒性证明的解耦边界网络分析。*CoRR*，abs/2006.09040，2020年。[Carlini 和 Wagner， 2017] Nicholas Carlini 和 David Wagner。迈向神经网络鲁棒性的评估。载于*2017 ieeesymposium on security and privacy(sp)*，第 39–57 页。IEEE，2017年。[Croce 和 Hein， 2020] Francesco Croce 和 Matthias Hein。通过多种无参数攻击的集成可靠地评估对抗鲁棒性。载于*ICML*，第 2206–2216 页。PMLR，2020年。[De Palma等， 2021] Alessandro De Palma、Rudy Bunel、Alban Desmaison、Krishnamurthy Dvijotham、Pushmeet Kohli、Philip H. S. Torr 和 M. Pawan Kumar。通过拉格朗日分解改进神经网络验证的分支定界算法，2021年。[Dwork等， 2012] Cynthia Dwork、Moritz Hardt、Toniann Pitassi、Omer Reingold 和 Richard Zemel。基于认知的公平性。载于第三届理论计算机科学创新会议论文集，第 214–226 页，2012年。[Ferlez等， 2022] James Ferlez、Haitham Khedr 和 Yasser Shoukry。快速 BATLLNN：对双层格神经网络进行快速盒分析。载于 Ezio Bartocci 和 Sylvie Putot 编，*HSCC '22*：第 *25* 届 *ACM* 国际混合系统：计算与控制会议，米兰，意大利，*2022*年*5*月*4*日至*6*日，第 23:1–23:11 页。ACM，2022年。[Ferrari等， 2022] Claudio Ferrari、Mark Niklas M¨uller、Nikola Jovanovic 和 Martin T. Vechev。通过多神经元松弛引导的分支定界实现完备验证。载于*ICLR*。OpenReview.net，2022年。[Fischer等， 2019] Marc Fischer、Mislav Balunovic、Dana Drachsler-Cohen、Timon Gehr、Ce Zhang 和 Martin Vechev。DL2：使用逻辑训练和查询神经网络。载于*ICML*，第 1931–1941 页，2019年。[Goodfellow等， 2014] Ian J Goodfellow、Jonathon Shlens 和 Christian Szegedy。解释并利用对抗样本。*arXiv* 预印本 *arXiv:1412.6572*，2014年。

[Guidotti 等人， 2020] Dario Guidotti、Francesco Leofante、Luca Pulina 和 Armando Tacchella。神经网络验证：通过剪枝提升可扩展性。arXiv 预印本 arXiv:2003.07636，2020年。[Henriksen 等人， 2021] P. Henriksen、K. Hammernik、D. Rueckert 和 A. Lomuscio。大型神经图像分类器的偏置场鲁棒性验证。载于第32届英国机器视觉会议（BMVC21）论文集。BMVA Press，2021年。[Huang 等人， 2018] Xiaowei Huang、Daniel Kroening、Wenjie Ruan、James Sharp、Youcheng Sun、Emese Thamo、Min Wu 和 Xinping Yi。深度神经网络安全性与可信性的综述。arXiv 预印本 arXiv:1812.08342，2018年。[Huang 等人， 2020] Xiaowei Huang、Daniel Kroening、Wenjie Ruan、James Sharp、Youcheng Sun、Emese Thamo、Min Wu 和 Xinping Yi。深度神经网络安全性与可信性的综述：验证、测试、对抗性攻击与防御，以及可解释性。《计算机科学评论》，37:100270，2020年。[Katz 等人， 2019] Guy Katz、Derek A Huang、Duligur Ibeling、Kyle Julian、Christopher Lazarus、Rachel Lim、Parth Shah、Shantanu Thakoor、Haoze Wu、Aleksandar Zelji´c 等人。用于深度神经网络验证与分析的 Marabou 框架。载于 CAV，第443–452页。Springer，2019年。[Khedr 等人， 2021] Haitham Khedr、James Ferlez 和 Yasser Shoukry。Peregrinn：惩罚松弛贪心神经网络验证器。载于 CAV，第 287–300页。Springer，2021年。[Kimmig 等人， 2012] Angelika Kimmig、Stephen Bach、Matthias Broecheler、Bert Huang 和 Lise Getoor。概率软逻辑简述。载于 NIPS概率编程研讨会：基础与应用论文集，第1–4页，2012年。[Kouvaros 和 Lomuscio， 2021] Panagiotis Kouvaros 和 Alessio Lomuscio。通过基于依赖关系的分支，实现可扩展的 ReLU 神经网络完备验证。载于 IJCAI，第2643–2650页，2021年。[Kunhimangalam 等人， 2013] Reeda Kunhimangalam、Sujith Ovallath 和 Paul K Joseph。一种用于识别腕管综合征严重程度的新型模糊专家系统。《生物医学研究国际》，2013，2013年。[Kurakin 等人， 2016] Alexey Kurakin、Ian Goodfellow 和 Samy Bengio。物理世界中的对抗样本。arXiv 预印本 arXiv:1607.02533，2016年。[Liu 等人， 2019] Changliu Liu、Tomer Arnon、Christopher Lazarus、Christopher Strong、Clark Barrett 和 Mykel J. Kochenderfer。深度神经网络验证算法，2019年。[Madry 等人， 2017] Aleksander Madry、Aleksandar Makelov、Ludwig Schmidt、Dimitris Tsipras 和 Adrian Vladu。迈向能够抵御对抗攻击的深度学习模型。arXiv 预印本 arXiv:1706.06083，2017年。

[Maity 等人， 2021] Subha Maity、Songkai Xue、Mikhail Yurochkin 和 Yuekai Sun。个体公平性的统计推断。arXiv 预印本 arXiv:2103.16714，2021年。[Mayer 等人， 2020] Tobias Mayer、Santiago Marro、Elena Cabrio 和 Serena Villata。为主题相关论证分类生成对抗样例 1。载于《论证计算模型》，第 33–44 页。IOS出版社，2020年。[Morris 等人， 2020] John X Morris、Eli Lifland、Jin Yong Yoo、Jake Grigsby、Di Jin 和 Yanjun Qi。Textattack：一个用于自然语言处理中的对抗攻击、数据增强和对抗训练的框架。arXiv 预印本 arXiv:2005.05909，2020年。[M¨uller 等人， 2020] Christoph M¨uller、Franc¸ois Serre、Gagandeep Singh、Markus P¨uschel 和 Martin Vechev。基于图形处理器扩展多面体神经网络验证。2020年。[M¨uller 等人， 2022] Mark Niklas M¨uller、Christopher Brix、Stanley Bak、Changliu Liu 和 Taylor T. Johnson。第三届国际神经网络验证竞赛（VNN-COMP 2022）：总结与结果，2022年。[Prabhakar 和 Rahimi Afzal， 2019] Pavithra Prabhakar 和 Zahra Rahimi Afzal。基于抽象的神经网络输出范围分析。NeurIPS，32，2019年。[Sarker， 2021] Iqbal H Sarker。机器学习：算法、现实应用与研究方向。SN计算机科学，2(3)：1–21，2021年。[Shriver 等人， 2021] David Shriver、Sebastian Elbaum 和 Matthew B. Dwyer。简化深度神经网络属性，以便通过对抗攻击进行证伪。载于《2021年IEEE/ACM第43届国际软件工程会议（ICSE）》，第275–287页，2021年。[Singh 等人， 2019] Gagandeep Singh、Timon Gehr、Markus P¨uschel 和 Martin Vechev。用于认证神经网络的抽象域。编程语言原理大会，3：1–30，2019年。[Smith 等人， 2021] Joshua Smith、Jarom Allan、Viswanathan Swaminathan 和 Zhen Zhang。基于反驳的深度神经网络对抗鲁棒性验证。载于《面向机器学习赋能自主系统的形式化方法》，2021年7月。[Tjeng 等人， 2017] Vincent Tjeng、Kai Xiao 和 Russ Tedrake。使用混合整数规划评估神经网络的鲁棒性。arXiv 预印本 arXiv:1711.07356，2017年。[Tran 等人， 2020] Hoang-Dung Tran、Xiaodong Yang、Diego Manzanas Lopez、Patrick Musau、Luan Viet Nguyen、Weiming Xiang、Stanley Bak 和 Taylor T Johnson。NNV：用于深度神经网络和机器学习赋能的网络物理系统的神经网络验证工具。arXiv 预印本 arXiv:2004.05519，2020年。[Vincent 和 Schwager， 2020] Joseph A. Vincent 和 Mac Schwager。可达多面体推进（RPM）：针对包含深度神经网络组件的机器人系统的安全验证算法，2020年。[VNNLIB， 2019] 神经网络验证库（VNN-LIB）。www.vnnlib.org，2019年。[Xiao等， 2018] Chaowei Xiao、Jun-Yan Zhu、Bo Li、Warren He、Mingyan Liu 和 Dawn Song。《空间变换对抗样本》。*arXiv* 预印本 *arXiv:1801.02612*，2018年。[Yu等， 2020] Tianhe Yu、Saurabh Kumar、Abhishek Gupta、Sergey Levine、Karol Hausman 和 Chelsea Finn。《多任务学习的梯度手术》，2020年。[Yurochkin等， 2019] Mikhail Yurochkin、Amanda Bower 和 Yuekai Sun。《通过敏感子空间鲁棒性训练个体公平机器学习模型》。*arXiv* 预印本 *arXiv:1907.00020*，2019年。[Zhang等， 2019] Huan Zhang、Hongge Chen、Chaowei Xiao、Sven Gowal、Robert Stanforth、Bo Li、Duane Boning 和 Cho-Jui Hsieh。《迈向可验证鲁棒神经网络的稳定高效训练》。*arXiv* 预印本 *arXiv:1906.06316*，2019年。[Zhang*等， 2022] Huan Zhang*、Shiqi Wang*、Kaidi Xu*、Linyi Li、Bo Li、Suman Jana、Cho-Jui Hsieh 和 J Zico Kolter。《用于基于界传播的神经网络验证的通用割平面》。*NeurIPS*，2022年。

A 术语与约束翻译

本节介绍术语，以及将约束转换为可微损失函数的过程。有关更多详情，请参阅 [Kimmig等，2012年；Fischer等，2019] 。

A.1 术语

本文使用的术语与布尔逻辑中使用的语言一致，且属于其子集。该语言由若干通过不同符号连接的术语组成。本文使用以下符号：

1. φ：表示一个公式或子公式。请注意，它的长度可以任意，并且通过使用其他符号（包括其自身）递归定义。
2. $\wedge$：布尔与 3. $\neg$：非

4. $\vee$ 布尔或
5. 比较项：指由两个 t项通过 $\wedge$ 或 $\vee$连接而成的术语。
6. t：指可微函数（例如ℓ_p 距离）或某个常数。

A.2 约束松弛

在传统逻辑中，每个约束（或规则）都被映射为 1（真）或 0（假）的真值。在这种设置下，很难捕捉不确定性，而这一特性对许多人工智能问题至关重要。此外，使用传统逻辑求解在计算上可能难以处理。与此相反，概率软逻辑（PSL） [Kimmig等人， 2012] ——一种用于关系域中集体概率推理的框架——使用区间 [0,1]内的软真值。这使得在这种设置下，推理可以转化为连续优化问题，从而高效求解。此外，还可以在该领域中将相似度函数及其他关系编码为约束。

为了确定约束满足的程度，PSL 使用*Lukasiewicz t*范数及其对应的余范数来松弛逻辑联结词。这些松弛形式以及约束的真值，都经过设计，确保在极端情况下（“完全为真”或“完全为假”）精确无误；更重要的是，它们为介于两者之间的值提供了一致的映射。这使我们能够计算平滑的梯度信息。下面给出了这些松弛形式的示例。

A.3 CGDTest 中的约束语言

与 [Fischer 等人类似， 2019], 在我们的设定中，约束语言由各项之间比较项的布尔组合构成，其中每一项要么是常数，要么是可微函数。更正式地说，约束语言可定义如下：

- $\varphi \Rightarrow \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid$ 比较项
- 比较项 $\Rightarrow t \leq t \mid t = t \mid t \neq t \mid t < t$
- $t \Rightarrow$ 常数 | 可微函数

每个项 t 可以是常数，也可以是定义在深度神经网络的输入、参数（例如中间层输出）和输出上的实值可微函数。在我们的实现中，我们纳入了多种常用的可微函数（例如 ℓ_∞ 距离），并且只要其他函数可微，也可以将其扩展纳入。

A.4 将约束转换为可微损失

由此得到的公式 φ 通过递归下降解析算法转换为可微损失函数。该算法的核心思想是将布尔连接词和比较项映射为一种近似松弛，使用*Lukasiewicz t*范数及其对应的余范数。具体过程如下。

比较项转换如下：

1. $\mathcal{L}(t \leq t') := max(t - t', 0)$,
2. $\mathcal{L}(t \neq t') := \epsilon * [t = t']$, 3. $\mathcal{L}(t = t') := \mathcal{L}(t \leq t' \wedge t' \leq t)$, 4. $\mathcal{L}(t < t') := \mathcal{L}(t \leq t' \wedge t \neq t')$。

约束的布尔组合转换如下：

1. $L(\varphi' \wedge \varphi'') := L(\varphi') + L(\varphi'')$,
2. $L(\varphi' \vee \varphi'') := L(\varphi') * L(\varphi'')$。

如果 φ 是否定 $\neg\varphi'$，则在将其转换为损失之前，会先将其改写为逻辑等价的约束，如下所示：

1. $\mathcal{L}(\neg(t = t')) := \mathcal{L}(t \neq t')$, 2. $\mathcal{L}(\neg(t \leq t')) := \mathcal{L}(t' < t)$, 3. $\mathcal{L}(\neg(t \neq t')) := = \mathcal{L}(t = t')$, 4. $\mathcal{L}(\neg(t < t')) := \mathcal{L}(t' \leq t)$, 5. $\mathcal{L}(\neg(\varphi' \wedge \varphi'')) := \mathcal{L}(\neg\varphi' \vee \neg\varphi'')$, 6. $\mathcal{L}(\neg(\varphi' \vee \varphi'')) := \mathcal{L}(\neg\varphi' \wedge \neg\varphi'')$, 7. $\mathcal{L}(\neg(\neg\varphi')) := \mathcal{L}(\varphi')$.

该过程会生成一个具有以下性质的函数：

1. 函数值非负，且约束违反数量越多，生成的值越大。2. 函数值为 0 表示完全满足所有约束。3. 该函数是次可微的（即几乎处处可微）。

由于这些性质，我们可以使用梯度下降算法，沿着约束损失减小方向的梯度修改一个样例，因为这相当于寻找一个能更好地满足这些约束的输入。或者，也可以在遗传算法中将其用作适应度函数，目标是找到使该值最小化的候选项。

B 约束

在本节中，我们提供有关实验中所使用约束的其他信息。我们的实验使用了五个约束：

B.1 ℓ_p 基于球的对抗约束

一个基于 ℓ_p 球的对抗样本使用 ℓ_∞ 来衡量相似性，并定义为：

$$(\|x - x'\|_\infty \leq \delta) \wedge (argmax(M(x)) \neq argmax(M(x')))$$

将约束转换为可微约束损失（使用附录 A 中提供的转换）后，我们得到：

$$max(\|x - x'\|_\infty - \delta, 0) + \eta * [argmax(M(x)) = argmax(M(x'))].$$

其中，$\eta > 0$是控制不等式损失值的参数，而 $[argmax(M(x)) = argmax(M(x'))]$ 是一个指示函数：如果 M 将 x 和 x'分类为相同标签，则返回 1，否则返回 0。可以看出，只有当 x' 是定义所述的对抗样本时，所得函数才为 0；否则，其值大于 0。这就是算法 1 第 2 行给出的约束损失。我们的实验中，δ 取值为 0.02，η 取值为 0.5。

B.2 伪装对抗约束

伪装对抗约束要求原始标签是 M的第二高标签。这里，x' 在某种意义上仍与 x 相似，因为 M 对 y仍有较高的置信度（因为它是第二高标签）。在之前对 ℓ_p 基于球的对抗约束的定义中，我们要求 x' 的新标签可以是除 y之外的任何标签；而在这里，我们要求 y 是第二高 标签。它们也与目标对抗样本略有不同，后者旨在使 M将 x' 分类为另一个标签 $y' \neq y$。

以手写数字识别为例。对于某些笔迹，数字1和7看起来很相似。因此，在执行分类任务时，我们更关注数字1被误分类为7或反之的情况。

同样，考虑以下场景：机场部署了一个图像识别系统，旨在通过照片识别不法分子。对于未经伪装的照片，模型能够正确识别不法分子。然而，不法分子很可能会进行伪装。在这种情况下，伪装图像可能会被列入前二，但不一定排名最高。伪装对抗样本会攻击模型，迫使其将前二输入误分类。

为了确保这一点，我们添加约束，使得 M 不会将 x' 分类为 y，但 y 的排名高于除一个标签以外的所有其他标签。换句话说：

$$argmax(M(x') \neq y \wedge (\bigvee_{y' \in Y - y} \bigwedge_{y'' \in Y - \{y, y'\}} argmax(M(x')) > y'')$$

换句话说，我们首先设置约束，使得 M 将 x'标记为 y以外的某个类别。此外，我们添加约束，使得 x' 上的 M 的标签大于除一个标签之外的所有标签。为简洁起见，我们省略可微函数。

B.3 流约束

流约束利用空间变换（流）寻找具有最小空间变换的对抗样本。其定义如下：

$$(L_{Flow}(x, x') \leq \delta) \wedge (argmax(M(x)) \neq argmax(M(x'))),$$

其中，L_{Flow} 实现了 Xiao等人所述的流损失。[Xiao 等人, 2018]。

与基于 ℓ_p 球的对抗约束类似，这可表示为：

$$max(L_{flow}(x, x') - \delta, 0) + \eta * [argmax(M(x)) = argmax(M(x'))]$$

请注意，以前的测试方法只是通过优化来提高错误分类分数，从而寻找对抗样本，并通过小幅增量更新确保其与原始输入的相似性。这种方法大多数时候效果很好，但并非总是如此，因为我们并未明确寻找相似的输入。相比之下，CGD 直接同时基于提高错误分类分数和相似性进行优化。

B.4 NLP 对抗约束

对于 NLP，约束被施加在非命名实体区域上。作为预处理步骤，会使用 Python 中的 nltk 包检查每个词是否为命名实体。随后，文本被投影到一个定长的连续空间，并确定连续空间中对应于非命名实体词元的区域。请注意，由于我们使用定长向量进行投影，因此会向长度较小的文本输入添加填充。

一旦我们识别出非命名实体区域，只需添加约束，使这些区域内的所有特征在原始输入和对抗性输入中保持一致。给定表示所有非命名实体特征的区域 R，自然语言处理中的对抗约束如下：

$$(\bigwedge_{r=1}^{R} x_r = x'_r) \wedge (argmax(M(x)) \neq argmax(M(x')))$$

相应的可微函数为：

$$(\sum_{r=1}^{R}(max(x_r - x'_r, 0) + max(x'_r - x_r, 0))) + \eta * [argmax(M(x)) = argmax(M(x'))]$$

B.5 公平性约束

对于我们的公平性约束，我们使用从数据中学习得到的距离度量，参见 [Maity 等人, 2021]。逻辑回归分类器旨在预测受保护变量，距离 $d^2 = ||(I - \prod_{\mathcal{H}})x - x'||_2^2$，其中 $\mathcal{H}$ 是由回归模型学习得到的敏感子空间，而 $\prod_{\mathcal{H}}$ 是其投影。

一旦我们有了这个距离度量，寻找不公平点就等同于：

$$argmax(M(x)) \neq argmax(M(x')) \wedge ||(I - \prod_{\mathcal{H}})x - x'||_2^2$$

对应的可微函数为：

$$\eta * [argmax(M(x)) = argmax(M(x'))] + max(||(I - \prod_{\mathcal{H}})x - x'||_2^2 - \delta, 0)$$

B.6 动机

我们使用后四种约束的动机，是为了揭示现有深度神经网络测试、分析与验证工具的一大弱点：遗憾的是，它们全都基于 ℓ_p-球验证。换言之，如果某个深度神经网络对于原始图像 ℓ_p-球范围内的扰动图像具有鲁棒性，那么（假设这些验证工具能够扩展到足够大的规模）如今的验证工具就会认证该网络能够抵御对抗攻击。遗憾的是，这样的认证意义不大，因为如前所述，攻击者仍然可以通过这些其他对抗约束攻击此类网络。此外，目前的深度神经网络测试、分析与验证工具都是针对特定领域定制的。这是因为不同领域（例如自然语言处理）会带来额外的约束，并且具有离散输入。我们的方法是将输入投影到连续编码中，并添加约束以遵循该领域固有的限制；这一方法有望应用于其他领域，从而开发出适用于多个领域的深度神经网络测试、分析与验证工具。