arXiv:2301.11435v1 [cs.LG] 26 Jan 2023

# 模理论学习

Matt Fre drikson [1] Kaiji Lu [1] Saranya Vijayakumar [1] Somesh Jha [2] Vijay Ganesh [3] Zifan Wan g [1]


## 摘要

近年来，将求解器层集成到深度神经网络（DNN）中的技术，在弥合归纳学习与符号推理技术之间长期存在的鸿沟方面展现出良好前景。本文提出了一系列将可满足性模理论（SMT）求解器集成到深度网络层前向与反向传播中的技术，并将该层称为SMTLayer。采用这种方法，可以将丰富的领域知识以数学公式的形式编码到网络中。在前向传播过程中，求解器利用前面各层生成的符号以及这些公式来构造推理；在反向传播过程中，求解器为网络参数更新提供指导，推动网络学习与求解器理论相符的表征。值得注意的是，求解器无需可微分。我们将SMTLayer 实现为 PyTorch 模块，实验结果表明，该方法得到的模型*1*）比传统模型所需的训练样本更少，*2*）能够抵御某些类型的协变量偏移，并且*3*）最终学到与符号知识一致、因而天然具有可解释性的表征。


## 1. 引言

近年来，一类技术旨在将求解器层集成到深度神经网络（DNN）中（Wang 等，2019a；Pogancic 等人，2020年；Huang 等人，2021年；Manhaeve 等人，2018年），并在训练和推理过程中使用。这种集成可发挥作用的一类问题，同时包含感知子问题和符号子问题，例如“视觉”数独（Wang 等，2019a），或根据地图图片确定最短路径的问题（Pogancic 等人，2020年）。

融入求解器层最直接的方法是训练能够学习与求解器所用符号相兼容的表征的机器学习模型。例如，若要利用符号域知识对鸟类图像进行分类，或根据 CT 扫描诊断疾病，就可以采用类似于“概念瓶颈”的方式训练模型（Koh 等人，2020年）。这需要详细的标签进行监督，而获取这些标签的成本可能高得令人难以承受，也难以确保它们与可能不断演变的领域理论保持一致。

我们提出了一套技术，将可满足性模理论（SMT）求解器集成到深度神经网络层中，从而利用符号知识学习兼容表示，而无需标签监督。我们的方法具有通用性，能够处理以 SMT 约束编码的广泛领域知识，前提是这些知识通过命题变量与周围的神经网络层交互。与最密切相关的先前工作（Wang 等，2019a）不同，我们并不通过构建可微松弛来近似求解器的行为。相反，我们在求解器处理一组约束时提取信息；这些约束旨在检查求解器前一层模型的输出是否正确。随后，我们利用这些信息在训练期间更新模型（第4.2节）。

我们为此提出两种不同的方法，一种基于不可满足核，另一种基于加权 MaxSMT（第 4.2 节）。这种方法有几个优点。除前文提到的宽松接口约束（即求解器与神经网络层通过布尔变量相互连接）外，我们的方法不会对嵌入层中的理论求解器施加任何限制，例如线性性（Pogancic 等人，2020年），甚至不会限制其可判定性——只要求解器能够高效地处理相关约束，该层就能按预期运行。由于无需为可能想要引入的每种理论或求解器技术提供可微松弛，我们可以充分利用求解器技术持续不断的进步。

我们将所提出的方法实现为 PyTorch（Paszke 等人，2019年）层，并使用 Z3（DeMoura & Bjørner，2008年）SMT 求解器作为求解器层，以求解 SMT 和 MaxSMT 约束。我们在涉及视觉和自然语言的三个应用中进行验证：视觉算术、代数方程求解，以及

[1]美国宾夕法尼亚州匹兹堡卡内基梅隆大学 [2]美国威斯康星州麦迪逊威斯康星大学麦迪逊分校 [3]加拿大安大略省滑铁卢滑铁卢大学。通信作者：MattFredrikson<mfredrik@cmu.edu>。

一种所谓的自然语言“说谎者谜题”，我们证明了该实现可融入深度神经网络架构，从而比传统深度神经网络更有效地解决问题（第5节）。具体而言，我们的结果表明，训练具备符号知识的深度神经网络所需的数据可能比其他情况下所需的数据简单得多；虽然这样做的计算成本更高，但通常效率更高的算法（即不涉及 MaxSMT 的算法）在实践中表现良好。

我们的贡献如下：

1. 我们提出了 SMTLayer，这是一种将 SMT 求解器集成到深度神经网络中的框架，并将其作为一个层，在训练和推理过程中利用符号知识。

2. 我们使用 PyTorch[1],实现了我们的方法，并展示了它可以应用于解决一系列融合符号知识的问题。

3. 我们针对四种不同应用进行的实证评估表明，使用SMTLayer 的模型所需的训练数据显著更少，训练效率更高，而且比基于密切相关的既有工作（Wang 等人，2019年；Huang 等人，2021年）更具鲁棒性。

第3 节介绍了经验风险最小化以及我们框架中使用的一阶理论的背景。第4 节介绍SMTLayer，第 5 节给出我们的实证评估，第 6节总结全文。

## 2. 相关工作

将逻辑求解器与深度模型相结合可能颇具挑战，因为逻辑具有离散结构，而如今构建神经网络最成功的方法则要求具备可微性（Riegel 等，2020年）。

**组合求解器层。** Vlastelica 等人 (2019年) 将一个黑盒、不可微的组合求解器集成在深度网络之上。为了在反向传播过程中将梯度通过求解器传播，他们对损失相对于求解器输入进行线性插值，并将求解器的梯度定义为各线段的斜率。组合求解器层可求解一类问题，在这些问题中，求解器的目标函数必须关于其输入呈线性，例如寻找最短路径和旅行商问题（TSP）。此外，作者假设唯一可用的标签是求解器的输出，例如 TSP 中的最小成本，因此，他们的工具必须自行发现网络输出对应的标签。这些要求限制了求解器层的选择范围。

[1]我们计划在本文发表后将实现作为开源库发布

**神经逻辑编程。** SATNet 在网络之上集成了基于逻辑的求解器，而 DeepProbLog 采取了相反的做法，通过神经谓词扩展概率逻辑编程语言的能力Manhaeve 等人 (2018年）。在我们的工作中，逻辑程序可以被视为一种显式编码符号知识的“求解器层”。Scallop（Huang 等人，2021年）在不牺牲相较于 DeepProbLog 的准确性的前提下扩展了 DeepProbLog，使其能够处理更大规模的问题。与 DeepProbLog 类似，MNIST 中两个数字之和的每种可能结果都被赋予一个概率，并以加权布尔公式的形式表示。他们会剪枝公式中不太可能的子句（以证明表示），只保留最$k$ 有可能的结果。可能性通过加权模型计数计算（Huang 等人，2021年；Chavira 和 Darwiche，2008年）。这些技术非常适用于受益于概率 Datalog 的问题，但也有固有局限：它们无法处理量词和一般否定，而且所支持的一阶理论范围也更加有限。

**SATNet。** Wang 等（2019b）提出了 SATNet，这是一种包含可微近似 MAXSAT 求解器层的网络架构。其近似方法基于坐标下降，用于求解 MAXSAT 问题的半定规划（SDP）松弛。SATNet 不假设问题的逻辑结构已知，而是尝试对其进行学习。SATNet 将 MAXSAT 求解器层置于卷积网络之上，从图像中学习表示，从而直接求解数独等问题；单靠神经网络并不擅长处理这类问题（Wang 等，2019b）。

**可微逻辑。** 近期另一项研究方向探索了可微逻辑（Fischer 等人，2019年；Varnai 和 Dimarogonas，2020年；van Krieken 等人，2022）。这些方法通过使逻辑公式可微，从而能够在将其纳入损失函数时进行优化，为将符号知识融入训练提供了途径。这一研究方向并未明确致力于在推理过程中利用符号信息。相比之下，我们的方法在训练期间从求解器中提取信息，并利用这些信息对模型进行条件化，使其形成一种表征，从而能够在推理期间与求解器有效通信。此外，我们不要求逻辑公式或求解器具备可微性。

## 3. 背景

令 $\mathbf{X}$ 表示特征域，$\mathbf{Y}$ 表示标签域，$\mathcal{D}$ 表示定义在 $\mathbf{X} \times \mathbf{Y}$上的分布。形式上，$\mathcal{D}$ 是一个概率测度，其定义在由 $\sigma$-代数在 $\wp(\mathbf{X} \times \mathbf{Y})$的子集上生成的空间中。学习算法 $A$ 的目标是找到一个函数 $h : \mathbf{X} \to \mathbf{Y}$，对于 $(x, y) \sim \mathcal{D}$，可用它在给定 $x$时预测 $y$。为此，向 $A$ 提供一组

训练样本 $S=(x_1,y_1)\ldots,(x_m,y_m)$，这些样本从 $\mathcal{D}$中独立同分布地抽取；随后，算法使用某种准则，从由函数构成的假设类 $\mathcal{H}$ 中选择 $h$ 。我们将 $h$ 称为学习算法 $A$ 在 $S$上学到的模型。当上下文已明确学习算法 $A$ 时，我们用 $h_S$ 表示根据给定样本生成的模型。本文中，我们通常假设损失为 0-1 损失 $\ell^{01}$或二元交叉熵 $\ell^{\text{bce}}$。

理论 $T$由签名 $\Sigma$ （包含常量、谓词和函数符号）以及关于 $\Sigma$的一组公理构成。理论中的公式由 $\Sigma$的元素、变量以及量词和布尔运算等逻辑符号组成。我们用*判定过程* 来指称一种算法：它接收一个开放的 $T$-公式；如果该公式可满足，则返回*true* ，否则返回 *false* 。此外，该算法还可以返回一个为所有变量赋值、从而证明公式可满足的赋值；如果公式不可满足，则可以返回一个*不可满足核*，即从公式的合取范式表征中取出的一组子句，且这些子句仍然不可满足。宽泛而言，我们也将这类算法称为“求解器”，但这个术语的含义更广，也可以指一种算法，用于确定在合取后仍可满足的最大子句集合，其中子句可能按用户定义的某些值加权。

## 4. 构建SMTLayer

本节介绍 SMTLayer，这是一组用于计算某一层前向与反向传播的算法，该层的行为由一组用户定义的 SMT 约束决定。 SMTLayer 没有可训练参数，其功能完全由模型设计者提供的一组 SMT 约束 $\phi$ 定义。 SMTLayer 可在现代深度学习框架中用作更传统神经网络层的即插即用替代品，例如，全连接层、卷积层和长短期记忆网络（Hochreiter 和 Schmidhuber，1997年）都是广泛使用的层的典型示例。

第 4.1 节概述了我们的方法，4.2 节详细介绍了这些算法，而 4.3 节则开始分析这一设定，我们希望未来的工作能继续深入探讨。

### 4.1. 概述

我们设想 SMTLayer 主要用于深度神经网络的顶层，接收来自一叠常规深度神经网络层的输入；这些层会将原始输入特征转换为约束中的基底项 $\phi(z_0,\ldots,z_{p-1},y_0,\ldots,y_{q-1})$，并嵌入SMTLayer中，生成与$\phi$ 及给定基底项一致的输出。图 1 展示了一个示例，采用此前研究过的 MNIST 加法问题（Manhaeve 等人，2018年；Huang 等人，2021年）。

**算法 1** $\mathsf{F}^{\phi}_{max}(z)$

基于 MaxSMT 的 SMTLayer 前向传播

**输入：** $z\in\mathbb{R}^p$ 层输入
$\phi$ $(z_0,\ldots,z_{p-1},y_0,\ldots,y_{q-1})$ $T$-公式
**输出：** $y\in\mathbb{R}^q$
**开始**
  $z_b \leftarrow [z[i]>0:\quad i=0\ldots p-1]$
  $C \leftarrow \sum_{i\in I}\text{softmax}(z)[i]$
  $y \leftarrow$ 取最大值 $_{y_b}\ \max_I C\cdot\mathbb{1}(\phi\wedge\bigwedge_{i\in I} y_i=y_b[i]\wedge z_i=z_b[i])$
  **返回** $y$
**结束**

在前向传播过程中，前一层的输出会映射到指定的自由变量 $z_0,\ldots,z_{p-1}$。随后，该层会检查 $\phi$的可满足性；$\phi$是在适当的一阶理论组合中构成的公式，检查前会将这些基底项代入 $z_i$。该层的输出则是求解器针对 $y_0,\ldots,y_{q-1}$ 给出的模型。这些输出通过将布尔值映射为浮点值来转换：*false* 映射为 -1，*true* 映射为 1。目前，我们对 $\phi$ 施加的唯一限制是，该层要求 $z$ 和 $y$是布尔值向量，以便将其适当地映射为连续值； $\phi$ 中出现的其他任何符号都可以来自底层 SMT 求解器支持的任意域（例如字符串）。

在反向传播过程中，该层接收其输出相对于正在求导的函数的梯度，我们假设该函数为二元交叉熵损失 $\ell(y,\ y^\star)$。除非另有说明，本节后续部分均假设使用此损失。结合对应前向传播的输入和输出，该梯度首先用于计算修正后的输出 $\hat{}y$ ，该输出对应于一个能够产生更小损失的输出。由于输出为布尔值，因此始终可以根据这些信息确定真实输出$y^\star$ 。利用 $\hat{}y$，该层确定其输入的哪些分量与$\phi$ 及 $\hat{}y$不一致，并将相应的梯度传递给前一层。第 4.2 节详细说明了这些梯度的计算方式。

### 4.2. SMTLayer，前向和反向

现在我们介绍 SMTLayer 前向与反向传播的细节。每种传播过程各有两种算法：$\mathsf{F}^{\phi}_{max}$ 和 $\mathsf{F}^{\phi}_{smt}$ 是前向传播，$\mathsf{B}^{\phi}_{max}$和 $\mathsf{B}^{\phi}_{core}$是反向传播。$\mathsf{F}^{\phi}_{max}$ 和 $\mathsf{B}^{\phi}_{max}$ 都使用 MaxSMT 求解器，而 $\mathsf{F}^{\phi}_{smt}$ 和 $\mathsf{B}^{\phi}_{core}$ 则依赖可满足性求解器（SMT）。尽管各算法使用的求解器类型具有对称性，但它们彼此兼容。也就是说，$\mathsf{F}^{\phi}_{smt}$ 可以与 $\mathsf{B}^{\phi}_{max}$ 或 $\mathsf{B}^{\phi}_{core}$搭配使用，$\mathsf{F}^{\phi}_{max}$也同样如此。

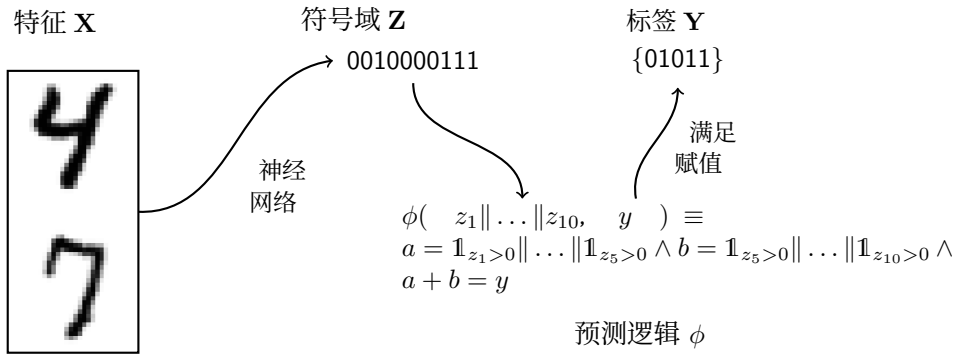



图 *1.* MNIST 加法示例。

**算法 2** $\mathsf{F}^{\phi}_{smt}(z)$
SMTLayer 的基于 SMT 的前向传播
**输入：** $z \in \mathbb{R}^p$ 层输入
$\phi(z_0, \ldots, z_{p-1}, y_0, \ldots, y_{q-1})$ $T$-公式
**输出：** $y \in \mathbb{R}^q$
1 **开始**
2 $z_b \leftarrow [z[i] > 0 : \quad i = 0 \ldots p-1]$
3 $\hat{\phi} \leftarrow \phi(z_b[0], \ldots, z_b[p-1])$
4 **if** $\hat{\phi}$ **可满足则**
5 $y_b[0], \ldots, y_b[q-1] \leftarrow$ 求解$(\quad \hat{\phi}, y_0, \ldots, y_{q-1})$
6 $y \leftarrow [y_b[i] > 0 \quad : \quad i = 0 \ldots q-1]$
7 **else**
8 $y \leftarrow \mathbf{0}$
9 **end**
10 **返回** $y$
11 **结束**

**算法 3** $\mathsf{B}^{\phi}_{max}(z, y, \partial_y \ell(y, y^\star))$
SMTLayer 的基于 MaxSMT 的反向传播
**输入：** $z \in \mathbb{R}^p$ 前向传播的输入
$y \in \mathbb{R}^q$ 前向传播的输出
$\partial_y \ell(y, y^\star)$ 关于输出的梯度
$\phi(z_0, \ldots, z_{p-1}, y_0, \ldots, y_{q-1})$ 一个 $T$公式
**输出：** $\partial_z \ell(y, y^\star) \in \mathbb{R}^p$ $\ell$ 的近似梯度
1 **开始**
2 $G_z \leftarrow \partial_z \ell(z, \text{符号}(z))$
3 $\hat{y} = \text{符号}(y) - 2 \cdot \text{符号}(\partial_y \ell(y, y^\star))$
4 **如果** $\text{sign}(y) \neq \text{sign}(\hat{y})$**那么**
5 $z_b \leftarrow [z[i] > 0 : \quad i = 0 \ldots p-1]$
6 $\hat{y}_b \leftarrow [\hat{y}[i] > 0 : \quad i = 0 \ldots q-1]$
7 $\phi_y \leftarrow \bigwedge_{0 \le i < q} y_i = \hat{y}_b[i]$
8 $C \leftarrow \sum_{i \in I} \text{softmax}(|z|)[i]$
9 $I \leftarrow \arg\max_{I \subseteq [0,p)} \mathbb{1}(\phi \wedge \phi_y \wedge \bigwedge_{i \in I} z_i = z_b[i]) \cdot C$
10 **对于每个** $i \in \bar{I}$ **do**
11 $G_z[i] \leftarrow \partial_{z[i]} \ell(z[i], \quad 1 - \text{符号}(z[i]))$
12 **end**
13 **end**
14 **返回** $G_z$
15 **结束**

**前向传播**。 算法 1 和 2 分别展示了 $\mathsf{F}^{\phi}_{max}$ 和$\mathsf{F}^{\phi}_{smt}$，即分别基于加权 MaxSMT 和 SMT 计算前向传播的方法。这两种算法都以用户提供的一阶公式 $\phi$作为参数，并接收一个由未缩放浮点值（*logits*）构成的单一向量输入。在这两种算法的第 2 行中，通过取这些值的符号将其转换为布尔常量，从而使它们能够与相应的自由变量 $z_0$、…、 $z_{p-1}$相等。

二者的关键区别在于$\mathsf{F}^{\phi}_{max}$ 和 $\mathsf{F}^{\phi}_{smt}$ 在将 $\phi$ 解释为布尔值时，处理与其不一致的输入的方式不同。 $\mathsf{F}^{\phi}_{smt}$ 对此的处理方式是提供一个同样与 $\phi$不一致的输出，即一个全零向量，从而有效地表明其下方的网络未能提供一致的输入。或者，我们可以将网络提供的值解释为带有“置信度”值的布尔值。尽管我们期望输入到 SMTLayer 中的是未缩放的浮点值，但算法 1 会通过 softmax 函数将其缩放为形式化的概率分布（第 3 行），以此作为权重来求解 $\phi$的加权 MaxSMT 解。采用这种方法，

SMTLayer 始终会提供有效的输出，尽管该输出不一定正确；它会根据网络下方部分最“确信”的输入，提供与 $\phi$ 一致的输出（第 4 行）。

**反向传播**。 反向传播负责计算损失相对于层输入的梯度。它接收损失相对于层输出的梯度，并假定已记住前向传播时先前接收的输入以及由这些输入产生的输出。此传播过程返回的梯度随后会用于前面各层的反向传播，并最终用于推导可训练参数的更新，从而降低损失。

为 SMTLayer 设计反向传播时，关键问题在于任一前向传播所计算函数的几何结构。对于任意向量 $v \in \{-1, 0, 1\}^p$和 $x, x'$，若 $\text{sign}(x) = \text{sign}(x') = v$，则 $\mathsf{B}^{\phi}_{\cdot}(x) = \mathsf{B}^{\phi}_{\cdot}(x')$，因此这些

**算法 4** $\mathrm{B}^{\phi}_{core}(z, y, \partial_y \ell(y, y^\star))$

SMTLayer 基于不可满足核的反向传播

**输入：** $z \in \mathbb{R}^p$ 前向传播的输入
$y \in \mathbb{R}^q$ 前向传播的输出
$\partial_y \ell(y,\ y^\star)$ 关于输出的梯度
$\phi(z_0、\ \ldots\ldots、\ z_{p-1}、\ y_0、\ \ldots\ldots、\ y_{q-1})$ 一个 $T$ 公式

**输出：** $\partial_z \ell(y,\ y^\star) \in \mathbb{R}^p$ $\ell$ 的近似梯度

1 **开始**
2 $G_z \leftarrow \partial_z \ell(z,\ \text{符号}(z))\hat{y} = \text{符号}(y) - 2 \cdot$
3 符号$(\partial_y \ell(y,\ y^\star))$
4 **如果** sign$(y) \neq$ sign$(\hat{y})$**则**
5 $z_b \leftarrow [z[i] > 0 :\quad i = 0 \ldots p-1]$
6 $\hat{y}_b \leftarrow [\hat{y}[i] > 0 :\quad i = 0 \ldots q-1]$
7 $\phi_z, \phi_y \leftarrow \bigwedge_{0 \le i < p} z_i = z_b[i], \bigwedge_{0 \le i < q} y_i = \hat{y}_b[i]$
8 $I \leftarrow \arg\min_{I \subseteq [0,p)} \mathbb{1}(\neg\phi \vee \neg\phi_y \vee \bigvee_{i \in I} z_i \neq z_b[i]) \cdot |I|$
9 **对每个** $i \in I$ **执行**
10 $G_z[i] \leftarrow \partial_{z[i]} \ell(z[i],\ 1 - \text{符号}(z[i]))$
11 **end**
12 **对每个** $i \in \bar{I}$ **do**
13 $G_z[i] \leftarrow 0$
14 **end**
15 **end**
16 **返回** $G_z$
17 **结束**

函数是分段常数阶跃函数，其取值遍及 $\mathbb{R}^q$ 单位超立方体的各个顶点。因此，尽管它们几乎处处可微，但梯度对训练没有帮助，因为它始终为零。以往将这类阶跃函数整合到深度网络中的工作，主要通过放宽前向传播所计算的函数来解决这一问题，使其梯度不再是常数，并有望提供更多有用的信息。

相比之下，$\mathrm{B}^{\phi}_{max}$ (算法 3) 和 $\mathrm{B}^{\phi}_{core}$ (算法4) 并不试图为前向传播的松弛形式提供梯度。相反，它们利用求解器在计算前向传播时提供的信息，识别输入中可能导致损失较高的部分。随后，通过构造一个反事实变体作为前向传播的输入来计算梯度。该变体会改变已识别的部分，并返回原始输入在此反事实变体上的二元交叉熵损失的梯度。这两种算法的区别在于它们从求解器提取的信息不同：一种提取 MaxSMT 实例的解，另一种提取不可满足核。

两种算法都先将梯度初始化为 logit 输入与其硬标签之间的损失（第 2 行）。请记住，我们假设损失 $\ell$ 是二元交叉熵，因此结果不会为零。此初始化的目的是模拟使用传统层和交叉熵损失进行训练时的动态过程；当舍入后的输出与目标一致时，损失仍不为零，训练会继续推动参数朝着使其与硬目标“更加”一致的方向变化。第 3 行中，算法随后利用后续层提供的梯度以及前向传播中的记忆化输出，构造出 $\hat{y}$，即满足 $\ell(\hat{y},\ y^\star) \le \ell(y,\ y^\star)$ 的“校正后”输出。如果 $\hat{y}$ 的符号与 $y$相同，那么两种算法都会返回初始化后的梯度。否则，它们会利用前向传播的输入 $z$ 和 $\hat{y}$从求解器中提取信息。

$\mathrm{B}^{\phi}_{max}$ 构造一个子句集合 $\phi_y$，用于约束自由$y_0$、…、$y_{q-1}$ 取值为 $\hat{y}_b$，即 $\hat{y}$ 的布尔值转换形式。随后，它计算绝对值输入 logits $|z|$ 的 softmax 值，并利用这些值找出与 $\phi \wedge \phi_y$ 一致的最大权重子句集合（$\mathrm{softmax}(|z|)[i],\ z_i = z_b[i]$）。直观而言，这些是前一层最确信的输入；通过改变一些置信度较低的输入，可以使它们与修正后的标签 $\hat{y}$ 保持一致。然后，$\mathrm{B}^{\phi}_{max}$ 会更新该实例的解与原始输入符号不匹配的各个索引处已初始化的梯度。

$\mathrm{B}^{\phi}_{core}$ 还会构造 $\phi_y$，但转而识别出一组与 $\phi \wedge \phi_y$不一致的约束 $z_i = z_b[i]$。请注意，第 7 行指定了一个最小不可满足核，但这并非必要。只需确保核中的子句都不是多余的，即从 $I$ 中删除任何一个单独的子句后，该集合都会变为可满足。如果核中仍有多余的子句，那么对应输入所返回的梯度符号就会不正确，这可能导致训练问题。随后，$\mathrm{B}^{\phi}_{core}$ 会使用 $z[i]$ 相对于 $1 - \mathrm{sign}(z[i])$ 的损失，更新核中所识别出的每个输入的梯度；这将使更新朝着这样的方向进行：修改输入，使得 $i$ 不再属于不可满足核。对于不在不可满足核中的索引，其梯度会被设为零，因为它们未出现在核中，并不能证明这些输入是正确还是错误的。

### 4.3. 分析

为了理解 SMTLayer 能够取得最优结果的情形，我们引入一类可分解的学习问题（定义1）。

**定义 1**（可分解问题）。设 $T$ 是一个常数取自 $\mathbf{Z}$ 的一阶理论。若存在函数 $f\colon \mathbf{X} \to \mathbf{Z}$、伴随假设类 $\mathcal{H}_f \subseteq \mathbf{X} \to \mathbf{Z}$ 和一个 $T$-公式 $\phi$，使得经验风险最小化问题 $\mathcal{D}, \mathcal{H}$ 可由 $T$ 分解，则称其为可分解的：

1. 对任意 $h \in \mathcal{H}$，存在 $h_f \in \mathcal{H}_f$ 和 $h'$，使得 $h = h' \circ h_f$。 2. 存在一个随机函数 $g\colon \wp(\mathbf{Y}) \to \mathbf{Y}$，使得对于 $\mathcal{D}^n$ 支持集中的任意 $n > 0$ 和 $\forall S$，

$$\Pr_{(x,y)\sim\mathcal{D}}[(x,y) \in S] = \Pr_{(x,\cdot)\sim\mathcal{D}}[(x, g(\langle x\rangle_{f,\phi})) \in S]$$

其中 $\langle x\rangle_{f,\ \phi} = \{y\colon \phi(f(x)\ y)$ 满足条件$\}$。

在（2）中， $f$ 称为基化函数，而 $\phi$ 称为预测逻辑。

直观而言，以分布 $\mathcal{D}$ 和假设类 $\mathcal{H}$ 定义的学习问题是可分解的，当且仅当 $\mathcal{H}$ 中的成员可以分解为分别负责基化和预测的函数，并且 $\mathcal{D}$ 可以用基化函数和一阶公式 $\phi$来表示。这里有几点值得注意。首先，并不要求基化函数 $f$ 是 $\mathcal{H}_f$的成员。虽然有时确实如此，但我们不应假设数据实际上是由所学习的类中的某个函数生成的，或能由该函数以完全准确的方式描述。事实上，我们不假设 $f$ 可以高效计算，因为它可能对应于某种自然过程，或数据生成中的某个方面，而我们对其了解不足，无法对其计算复杂度作出这样的判断。

其次，对于给定的 $x$， $y$到 $\phi$可能有多个满足赋值（$f$（$x$） $y$）。式（2）中的函数 $g$考虑了这一点，它只要求：当 $\phi$（$f$（$x$） $y$）的解由 $g$采样时，所得结果与 $\mathcal{D}$具有相同的分布。本文将重点讨论 $y$ 的满足赋值唯一的情形，因为这更符合“经典”的经验风险最小化分类问题。更一般情形的研究留待未来工作。

我们注意到，如果基化函数已知、能够高效计算，并且 $\phi$ 可以高效求解，那么这个学习问题实际上有一个闭式解。相反，我们假设只有 $\phi$ 以及可能的 $g$ 是已知的，并且给定了 $\mathcal{D}$ 的一个样本。剩下的挑战是找到一个基化假设 $h_f \in \mathcal{H}_f$，使得式（2）中的构造能够有效解决由 $\mathcal{D}$、 $\mathcal{H}$ 所提出的端到端学习问题。这与传统经验风险最小化形成对比：在传统经验风险最小化中，一个良好的解 $h \in \mathcal{H}$ 必须要么同时解决基化和预测问题，要么找到一种“捷径”，使其在预测 $\mathcal{D}$ 方面的表现不逊于这种分解方法。

**收敛**。关于反向传播，下面的定理2 表明，当 $\phi$ 满足某些条件，且伴随假设类 $\mathcal{H}_f$ 满足足以保证随机梯度下降收敛的条件时，使用$\mathsf{F}^{\phi}_{smt}$ 和$\mathsf{B}^{\phi}_{max}$ 进行训练，将在相应的迭代次数内收敛到最优解。该定理的证明基于这样一个观察：当满足关于 $\phi$ 的条件时，使用$\mathsf{B}^{\phi}_{max}$进行训练，会得到与使用 $\phi$ 的标签进行监督学习时相同的解。因此，关于$\mathcal{H}_f$ 的条件足以确保上述收敛性，正如《Shalev-Shwartz&Ben-David，2014》第14章概述的一项著名结果所述。

还值得注意的是，定理2并不一定成立，如果使用$\mathsf{B}^{\phi}_{core}$代替$\mathsf{B}^{\phi}_{max}$。原因在于，可能存在许多在基数意义下局部最小的不可满足核，而梯度仅针对出现在所计算核中的输入进行设置。这些梯度将不同于基化样本上的损失梯度，因此训练动态很可能有所不同。我们认为，使用 $\mathsf{B}^{\phi}_{core}$ 进行训练可能与分块坐标下降比与梯度下降更为相似，并将对这一主题更详细的探讨留待未来工作。

**定理 2**。 设 $\mathcal{D}, \mathcal{H}$ 是一个具有基化函数 $f$ 和预测逻辑 $\phi$ 的 $T$-可分解问题，其中：

*1.* $\mathbf{Z}$ 和$\mathbf{Y}$ 是布尔值的笛卡尔积。

*2.* 对任意$(x, y) \sim \mathcal{D}$和 $y' \neq y$, $\phi(f(x)\ y')$ 都与 *false* 等价，并且恰有一个 $z$ 使得$\phi(z, y)$ 与 *true* 等价。

*3.* $\mathcal{H}_f$是一个凸集，并且对于所有 $h_f \in \mathcal{H}_f$, $\|h_f\| \leq B$, 损失 $\ell(h_f(\cdot)\ z)$对于任意固定的 $z$，关于$x$是 $M$-利普希茨连续且凸的。

那么，对于任意 $\epsilon > 0$，通过最小化下式来选择 $h_f$：$L_S(\mathsf{F}^{\phi}_{smt}(h_f(\cdot)))$，并使用 $\tau \geq {}^{M^2B^2}/_{\epsilon^2}$次随机梯度下降迭代（梯度由 $\mathsf{B}^{\phi}_{max}$提供），学习率为 $\eta = \sqrt{{}^{B^2}/_{M^2\tau}}$，即可得到一个满足以下条件的基化假设ˆ ˆ$h_f \in \mathcal{H}_f$： $\mathbb{E}[L_{\mathcal{D}}(h_f)] \leq \min_{h_f \in \mathcal{H}_f} L_{\mathcal{D}}(h_f) + \epsilon$。该期望中的随机性取决于随机梯度下降算法的选择。

## 5. 实验评估

本节将对SMTLayer 在四个可分解为感知子任务和符号子任务的学习问题上进行实证评估。结果展示了以下主要发现。 *1*）SMTLayer效果显著：在每个基准测试中，与未编码符号知识的“传统”学习方法相比，它都能取得更优结果。 *2*）SMTLayer 相较于以往方法具有明显优势。与 SAT-Net（Wang 等，2019a）相比，它收敛所需的训练数据显著更少，且在所有情况下都能得到更准确的模型；与 Scallop（Huang 等人，2021年）相比，它的计算开销更低，所需训练数据更少，并且能够编码的知识表达能力更强。 *3*）使用 SMTLayer 训练的模型可能对相对于问题符号组件发生的某些类型的协变量偏移更具鲁棒性；当 SMTLayer 成功学习到兼容表示后，只要感知组件保持稳定，它就能持续产生正确的推理结果。

### 5.1. 数据集

关于我们评估中使用的数据集及其对应架构的更多详情，请参阅附

录A.2；每个数据集训练时使用的具体超参数见附录A.3。

**MNIST 加法**。 MNIST 加法问题如图 1所示，与 Huang 等人 (2021年) 描述的基准测试相似。训练时，我们用“MNIST +$p$%”表示大小为 60,000、包含 $p\%$ 所有可能数字对的训练集。因此 $p = 100$ 表示使用所有可能的数字对，而对于 $p = 10$，我们只使用数字相同的数字对。我们的实验采用 $p = 10$、25、50、75 和 100。在所有情况下，我们都使用同一个测试集，其中包含来自所有可能数字对的实例。

**视觉代数**。 任务是求解图示方程 $x$ 中的变量 $ax + b = c$，其中$a$、$b$ 和 $c$ 是随机选取的数字，每个符号均使用 EMNIST（Cohen 等人，2017a）和 HASY 图像（Thoma，2017b）进行视觉呈现。与 MNIST 加法类似，训练样本从相同数字组成的数字对中均匀抽取 $a$ 和 $b$，并从 0 到 9 的奇数中均匀抽取 $x$。测试样本通过从所有数字对中均匀抽取 $a$、$b$，并从 0 到 9 的所有数字中抽取 $x$ 而生成。

**说谎者谜题**。 说谎者谜题由 Alice、Bob 和 Charlie 三位不同参与者说出的三句话组成。其中一位参与者犯有某种未明确说明的罪行，而在每句话中，相应的参与者都会声称另一方有罪或无辜。例如，*“Alice 说 Bob 是无辜的。”*假设其中两位参与者诚实，而有罪方并非如此。问题的解答就是确定有罪方。底层逻辑的形式化描述见附录 A.2。值得注意的是，该逻辑中存在非分层的否定，因此无法用 Scallop 编码。我们选择的训练样本并未完整指定该逻辑，因此常规训练应不足以识别出良好的模型。

**视觉数独**。 这项任务是完成一个 $9 \times 9$ 数独棋盘，其中每个格子都是一个 MNIST 数字。我们使用 SATNet 评估中的数据集（Wang 等，2019a），并考察从原始训练集分别抽取 10%、50% 和 100% 样本得到的三种配置。尽管已有一些以逻辑程序实现数独求解器的示例，但我们无法在不违反分层否定的情况下用 Scallop 实现一个求解器。在计算准确率时，我们会检查整个数独棋盘是否正确。

### 5.2 设置

我们使用 Py- torch 实现了所提方法的原型（Paszke 等人，2019年）和 Z3（De Moura & Bjørner，2008年），论文发表后将以开源形式提供

本文发表后。使用 SMTLayer 训练模型时，我们采用随机梯度下降，并使用 Nesterov 动量，动量率为 0.9，梯度裁剪率为 0.1。在使用 SMTLayer（或比较技术，除非另有说明）训练模型之前，我们首先通过将 SMTLayer 替换为一个包含一个 512 个神经元隐藏层的全连接网络，对神经网络进行预训练。这可能会减少较低层所需的训练更新次数，但不会得到具有兼容符号知识表征的模型，因此仍需进一步训练。在我们的评估中，标记为“传统”的模型采用了与预训练所用模型相同的架构。训练结果取五次运行的平均值。

我们的评估是在一台配备 Intel i9 1050K CPU、64GB 内存和 GeForce RTX 3080 加速器的机器上进行的，该机器运行 Ubuntu 20.04.4，并使用 CUDA 11.1.0 和 cuDNN 8.0.4。我们使用 PyTorch 1.7.0a0+7036e91 版和 Z3 4.8.14 开发并测试了原型，评估结果也使用了这些版本。

### 5.3 结果

**总体性能**。在准确率方面，表1显示，在所有配置下，SMTLayer 在准确率、训练时间或两者方面均优于传统网络和先前的工作。尽管使用SMTLayer （或上述任何方法）进行训练比传统方法开销更大，但 SMTLayer 始终比 Scallop 快（视觉代数情况下几乎 $4\times$ ）。训练 SATNet 模型的每轮耗时低于 SMTLayer，但这并不能说明全部情况。在视觉数独任务中，10%SMTLayer 模型仅用 15 轮训练就达到了更低的错误率，而 100% SATNet 模型则需要 100 轮；这意味着SMTLayer模型的训练时间不到其十分之一。

还值得注意的是，尽管理论 2 表明算法 3 在学习方面可能优于算法 4，但我们发现，在这些数据集上并非如此。表 1中的所有结果均使用算法 4训练，并使用算法 1进行测试推理。

**训练样本数量**。由于 SMTLayer 编码了明确的知识，而这些知识对于在这些数据集上进行正确推理至关重要，因此，我们的方法能够在数据匮乏的环境中取得良好表现，即使训练样本不足以完整指定学习任务中的符号组件也是如此。表 1中的结果清楚地表明了这一点：在 MNIST 加法和第一个视觉代数配置中，SMTLayer 生成的模型几乎表现完美，尽管在大多数情况下训练样本并不充足。由于 SATNet 必须学习符号组件，因此处于劣势，在这些环境中的表现与传统模型相近。理论上，Scallop 应当能够

| 配置 | 传统方法 | | 带有 SMTLayer | | 带有 SATNet | | 带有 Scallop | |
|---|---|---|---|---|---|---|---|---|
| | *test* 准确率（%） | 训练轮次 时间（秒） | *test* 准确率（%） | 训练轮次 时间（秒） | *test* 准确率（%） | 训练轮次 时间（秒） | *test* 准确率（%） | 训练轮次 时间（秒） |
| MNIST+ 10% | 10.0 | 7.1 | **98.1** | 75.4 | 10.0 | 31.0 | 33.7 | 96.3 |
| MNIST+ 25% | 32.5 | 7.1 | **98.3** | 74.8 | 34.2 | 30.9 | 65.8 | 96.4 |
| MNIST+ 50% | 51.5 | 7.0 | **98.6** | 75.8 | 54.8 | 32.8 | **98.4** | 96.5 |
| MNIST+ 75% | 76.1 | 7.0 | **98.5** | 75.0 | 78.4 | 31.9 | 93.5 | 96.4 |
| MNIST+ 100% | 98.3 | 7.1 | **98.5** | 75.8 | 96.7 | 33.5 | **98.6** | 96.6 |
| 可视化算法 #1 | 24.1 | 13.2 | **98.2** | 168.2 | 19.6 | 80.1 | 18.7 | 602.8 |
| 视觉算法 #2 | **25.4** | 11.2 | **25.4** | 127.2 | 18.6 | 52.5 | 21.3 | 636.1 |
| 说谎者谜题 | 54.2 | 3.1 | **86.1** | 28.7 | 84.6 | 3.0 | — | — |
| 可视化数独 10% | 0.0 | 6.3 | **66.0** | 135.7 | 0.0 | 9.9 | — | — |
| 可视化数独 50% | 0.0 | 28.3 | **73.1** | 608.1 | 0.0 | 45.4 | — | — |
| 可视化数独 100% | 0.0 | 26.7 | **79.1** | 1199.0 | 63.2 | 86.5 | — | — |

*表 1.* 使用 SMTLayer 与传统架构进行训练和推理后的结果。我们使用公开可用的 SATNet（Wang 等，2019年a）和 Scallop（Huang 等人，2021年）的实现，并采用与其代码中相同的超参数。所有 SMTLayer 测试准确率均通过 MaxSMT 前向传播测得。训练轮次耗时取模型训练所经历的所有训练轮次的平均值。标记为 — 的单元格表示该问题与此方法不兼容。

达到与 SMTLayer 同样好的表现，因为它也编码了明确的知识。然而，它无法为任何一种视觉代数配置学习出有用的模型；在 MNIST 加法任务中，只有在训练期间接触到一半可能的数字对之后，它才能学会正确的表征。

SMTLayer 在 Wang 等介绍的视觉数独数据集上表现尤为出色。仅使用原始训练样本的 10% 进行训练时，它就能学到一个函数，其性能超过SATNet 模型一大截，而且随着接触到更多训练数据，这一优势还会不断扩大。另一方面，我们发现，若训练样本少于完整的原始训练样本，SATNet 就无法收敛。

**鲁棒性与可解释性**。SMTLayer 之所以能够表现出色，并且在许多情况下接近最优，即使在其他方法表现不佳的配置下也是如此，是因为它学到的表征与 SMTLayer 中编码的符号知识一致。例如，我们用于 MNIST 加法、视觉代数和视觉数独的约束，都将数字编码为位向量。为了进行正确的推理，神经网络必须学会将 MNIST 数字编码为正确的位向量表征。如果成功学会了这一点，就会带来两个积极结果。首先，模型的表征本质上具有可解释性，因为它与所提供的符号领域知识相吻合，而这些知识（想必）也是可解释的。其次，对于不会影响网络转换为理论符号的感知数据分布、但会影响这些符号组合统计特性的协变量偏移，最终得到的模型天然具有鲁棒性。

这种转变体现在 MNIST 10% 和视觉代数实验中：在训练阶段，模型只能看到由相同数字组成的数字对，而在测试阶段，它会接触到分布显著不同的数字对或公式。我们通过考察 SMTLayer 和 Scallop 在 MNIST 加法 10% 上学习到的表征，验证了这一点；由于 SATNet 在训练期间没有获得可解释的理论，因此不能合理地期待它学习到可解释的表征。正如预期，SMTLayer 以与典型 MNIST 模型相当的准确率生成正确的表征（$\approx 99\%$），而 Scallop 的数字表征正确率约为 50%。然而，架构也会影响这种鲁棒性，这一点从 SMTLayer 在第二种视觉代数配置中的结果可以看出。由于网络接收到的是完整实例，而非单个数字，因此它学会了训练分布中的偏差。尽管 SMTLayer 可以访问符号公式，但它无法将感知符号与其协方差 disentangle。理解这一问题是未来工作的重要方向。

## 6. 结论

我们提出的将逻辑理论融入深度学习的方法 SMTLayer，为如何将符号知识纳入训练和推理提供了一种务实的解决方案。我们在若干涉及感知任务（视觉和自然语言）以及逻辑推理的问题上展示了这一方法。值得注意的是，我们证明，在训练和推理过程中融入符号知识的模型，其性能可以超过传统模型以及该领域以往的研究成果，尤其是在训练数据有限的情况下。过去几十年来，自动推理技术的持续进步在多个领域的发展中发挥了关键作用。我们希望本文的贡献能够推动相关领域进一步发挥其潜力，以应对超出现有学习技术能力范围的挑战。

## 参考文献

Chang、O.、Flokas、L.、Lipson、H. 和 Spranger、M. 评估 SATNet 解决符号接地问题的能力。神经信息处理系统进展，33:1428–1439，2020年。

Chavira, M. 和 Darwiche, A. 通过加权模型计数进行概率推理。《人工智能》，172 (6-7):772–799, 2008年。

Cohen, G., Afshar, S., Tapson, J. 和 Van Schaik, A. Em- nist：将 MNIST 扩展至手写字母。载于《2017年神经网络国际联合会议 (IJCNN)》，第 2921–2926 页。IEEE，2017a。

Cohen, G., Afshar, S., Tapson, J. 和 van Schaik, A. EMNIST：MNIST 的手写字母扩展。*CoRR*，abs/1702.05373，2017b。网址 http://arxiv.org/abs/1702.05373。De Moura, L. 和 Bjørner, N. Z3：一种高效的 SMT 求解器。载于系统构造与分析工具和算法国际会议，第 337–340 页。施普林格，2008年。Deng, L. 用于机器学习研究的手写数字图像 MNIST 数据库。*IEEE*信号处理杂志，29(6):141–142，2012年。Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C. 和 Vechev, M. DL2：使用逻辑训练和查询神经网络。载于第*36*届国际机器学习会议论文集，第 97 卷，机器学习研究论文集，第 1931–1941 页。PMLR，2019年。Hochreiter, S. 和 Sch midhuber, J. 长短期记忆。神经计算，9(8):1735–1780，1997年11月。ISSN 0899-7667。Huang, J., Li, Z., Chen, B., Samel, K., Naik, M., Song, L. 和 Si, X. Scallop：从概率演绎数据库到可扩展的可微推理。神经信息处理系统进展，34:25134–25145，2021年。Koh, P. W., Nguyen, T., Tang, Y. S., Mussmann, S., Pierson, E., Kim, B. 和 Liang, P. 概念瓶颈模型。载于国际机器学习会议，第 5338–5348 页。PMLR，2020年。Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T. 和 De Raedt, L. DeepProbLog：神经概率逻辑编程。神经信息处理系统进展，第 31 卷，2018年。

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., 和 Chintala, S. Pytorch：一种命令式风格的高性能深度学习库。载于神经信息处理系统进展 *32*。2019年。Pennington, J., Socher, R., 和 Manning, C. D. Glove：用于词表示的全局向量。载于*2014*年自然语言处理实证方法会议论文集（*EMNLP*），第1532–1543页，2014年。Pogancic, M. V., Paulus, A., Musil, V., Martius, G., 和 Rol´ınek, M. 黑盒组合求解器的微分。载于第八届国际学习表征会议（*ICLR 2020*），埃塞俄比亚亚的斯亚贝巴，*2020*年*4*月*26*日至*30*日。OpenReview.net，2020年。网址：https://openreview.net/forum?id=BkevoJSYPB。Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I. Y., Qian, H., Fagin, R., Barahona, F., Sharma, U., 等。逻辑神经网络。*arXiv* 预印本 *arXiv:2006.13155*，2020年。Shalev-Shwartz, S. 和 Ben-David, S. 理解机器学习：从理论到算法。剑桥大学出版社，2014年。ISBN 1107057132。Thoma, M. Hasyv2 数据集。*CoRR*，abs/1701.08380，2017a。网址：http://arxiv.org/abs/1701.08380。Thoma, M. Hasyv2——手写符号数据库，2017年1月。网址：https://doi.org/10.5281/zenodo.259444。Topan, S., Rolnick, D., 和 Si, X. 使用 SATNet 进行符号落地的技术。神经信息处理系统进展，34:20733–20744，2021年。van Krieken, E., Acar, E., 和 van Harmelen, F. 对可微模糊逻辑算子的分析。人工智能，302:1–46，2022年1月。Varnai, P. 和 Dimarogonas, D. V. 关于 STL 任务学习的鲁棒性指标。载于*2020*年美国控制会议（*ACC*），第5394–5399页。IEEE，2020年。Vlastelica, M., Paulus, A., Musil, V., Martius, G., 和 Rol´ınek, M. 黑盒组合求解器的微分。*arXiv* 预印本 *arXiv:1912.02175*，2019年。Wang, P., Donti, P. L., Wilder, B., 和 Kolter, J. Z. Sat-net：利用可微可满足性求解器连接深度学习与逻辑推理。载于Chaudhuri, K. 和 Salakhutdinov, R.（编），第*36*届国际机器学习会议论文集（*ICML 2019*），

*2019*年*6*月*9*日至*15*日，长滩，加利福尼亚州，美国，第97卷，机器学习研究论文集，第6545–6554页。PMLR，2019a。网址：http://proceedings.mlr.press/v97/wang19e.html。

Wang, P.-W., Donti, P., Wilder, B. 和 Kolter, Z. Satnet：使用可微可满足性求解器连接深度学习与逻辑推理。载于国际机器学习会议，第 6545–6554 页。PMLR，2019b。

Yurichev, D.《SAT/SMT 示例》，2020年。可在此处获取：https://sat-smt.codes/（2023年1月）。

# A. 附录

## A.1. 证明

**定理 2**。 设 $\mathcal{D}, \mathcal{H}$ 是一个具有基化函数 $f$ 和预测逻辑 $\phi$ 的 $T$-可分解问题，其中：

*1.* **Z** 和**Y** 是布尔值的笛卡尔积。

*2.* 对于任意$(x, y) \sim \mathcal{D}$和 $y' \neq y$, $\phi(f(x)\ y')$，它与 ***false*** 是 $T$-等价的，并且恰好存在一个 $z$，使得 $\phi(z, y)$ 与 ***true*** 是$T$-等价的。

*3.* $\mathcal{H}_f$是一个凸集，并且对于所有 $h_f \in \mathcal{H}_f$, $\|h_f\| \leq B$，损失 $\ell(h_f(\cdot)\ z)$ 都是 $M$-利普希茨连续的，且对于任意固定的 $z$，关于 $x$ 是凸的。

那么，对于任意 $\epsilon > 0$，通过最小化 $L_S(\mathsf{F}^{\phi}_{smt}(h_f(\cdot)))$ 来选择 $h_f$，并使用梯度由 $\mathsf{B}^{\phi}_{max}$提供、学习率为 $\eta = \sqrt{B^2/M^2\tau}$ 的随机梯度下降进行 $\tau \geq {}^{M^2B^2}/_{\epsilon^2}$次迭代，即可得到基化假设 $\hat{}\ h_f \in \mathcal{H}_f$，该假设$\hat{}$满足： $\mathbb{E}[L_{\mathcal{D}}(h_f)] \leq \min_{h_f \in \mathcal{H}_f} L_{\mathcal{D}}(h_f) + \epsilon$。该期望中的随机性取自随机梯度下降算法的选择。

证明。 为证明这一结果，我们引入基化样本的概念。

**定义 4** （基化样本）。 设 $\mathcal{D}, \mathcal{H}$是一个 $T$-可分解问题，其基化函数为 $f$。对于 $S \sim \mathcal{D}$，基化样本 $S_f$定义为 $[(x_i, f(x_i)) : (x_i, y_i) \in S]$，即由$S$ 中每个实例的第一个元素及其在 $f$下的图像所组成的元组。

现在注意到，假设 (3) 中陈述的条件足以得出该结论：如果我们得到的是基化样本 $L_S(\mathsf{F}^{\phi}_{smt}(h_f(\cdot)))$，而不是对其进行优化，并最小化 $S_f$ $L_{S_f}(h_f)$(参见 (Shalev-Shwartz & Ben-David，2014)，定理 14.8)。由假设 (1) 和 (2) 可知，结论仍如前所述成立，因为这两个假设意味着 $\mathsf{B}^{\phi}_{max}$提供的更新向量就是 $L_{S_f}(h_f)$的梯度。

要理解其中的原因，请注意，两个算法在第 3 行计算出的 $\hat{}y$ 的符号必定与 $y^\star$ 的符号相同。这可由以下两点得出：

1. 在任意满足 $y[i] \neq y^\star[i]$ 的坐标 $i$ 处，$\text{sign}(\partial_y \ell(y,\ y^\star))[i] = \text{sign}(y)[i]$。
2. 在任意满足 $y[i] = y^\star[i]$ 的坐标 $i$ 处，$\text{sign}(\partial_y \ell(y,\ y^\star))[i] = -1 \cdot \text{sign}(y)[i]$。

现在需要考虑两种情况。

**情况 1：** $\text{sign}(y) = \text{sign}(\hat{}y)$。在这种情况下，算法返回 $\partial_z \ell(z,\ \text{sign}(z))$。由于 $\phi(\cdot,\ y)$ 的解是唯一的，因此 $\text{sign}(z)$ 是正确的基化结果，也就是说，对于原始特征 $x$，$z = f(x)$。

**情况 2：** $\text{sign}(y) \neq \text{sign}(\hat{}y)$。根据假设 (2)，第 8 行计算出的索引集合将包含 $z$ 与正确值 $z^\star = f(x)$ 相匹配的所有坐标。注意，在这些坐标上，算法返回的向量与 $\ell(z,\ z^\star)$ 的梯度相匹配，即 $\partial_{z[i]} \ell(z[i],\ \text{sign}(z[i]))$。在其余坐标上，向量将包含 $\partial_{z[i]} \ell(z[i],\ 1 - \text{sign}(z[i]))$，这也与 $\ell(z,\ z^\star)$ 的梯度相匹配。结论得证。 □

## A.2. 数据集详情

我们研究的三个问题中，有两个基于 MNIST 手写数字数据集（Deng，2012年），该数据集包含 60,000 张 28x28 的手写数字灰度图像，用于训练，以及 10,000 个用于测试的实例。图 1 左侧的数字是该数据集中实例的示例。为生成视觉代数问题的数据，我们还使用了 EMNIST（Cohen 等人，2017b），该数据集在 MNIST 的基础上增加了手写字母；以及 HASY（Thoma，2017a），其中包含与 MNIST 特征相似的手写符号。对于说谎者谜题，受一些示例（Yurichev，2020年）的启发，这些示例将类似问题表述为 SMT 约束，我们使用自行设计的一组常用短语构造了示例，除此之外并未使用公开数据。

下面，我们将介绍如何利用这些数据源构建训练样本和测试样本，以及用于结合 SMTLayer 求解这些问题的神经网络架构。

**MNIST 加法**。 MNIST 加法问题在示例 1 中进行了描述。在每个实例中，两个 MNIST 数字作为特征输入，模型的任务是给出它们的和，并以位向量表示。我们使用的架构由四个卷积层组成，卷积核大小为 3，深度依次为 64 、 64 、 128 、 128，第一层的步幅为 2 ； 此外还有两个全连接层，宽度分别为 256 和 4。该网络分别处理每个数字，再将结果拼接成一个大小为 8 的向量，并传递给一个 SMTLayer 实例，该层使用示例 1中的 SMT 约束，最终生成一个宽度为 5 的向量，用以表示这些数字之和的位向量。

我们生成了五组训练样本，首先创建一组仅包含相同数字对的数据，即 (0，0)，(1，1) .....。随后，我们从所有可能的数字对中逐步加入更多组合，分别使用 25%、50% 和 100% 的组合，并在所有数据集上以 128 个样本为一批进行训练。请注意，尽管不同样本中出现的数字对数量有所变化，我们始终会将这些数字对映射到随机的 MNIST 图像，以获得 60,000 个训练实例。这样可以确保训练样本包含足够多的 MNIST 图像，从而在测试数据上取得良好表现。在所有情况下，我们使用相同的测试集，其中包含来自所有可能数字对的实例。这样做是为了证明，传统网络只有在见过完整分布后才能实现泛化，而采用 SMTLayer 的模型在见过少得多的样本后就应当能够实现泛化。

**视觉代数**。视觉代数问题如示例2 和示例3所述。回想一下，特征描绘的是手写形式的线性方程，其形式为 $ax + b = c$。 $a$、 $x$ 和 $b$ 的取值均从 1–9 的范围内随机抽取，以确保解唯一。接着，将 $c$ 的相应值分解为 $c = 10 \cdot c_1 + c_2$，并随机选取 MNIST 数字来表示 $a$、 $b$、 $c_1$、 $c_2$。变量使用随机抽取的 EMNIST 字母字符表示，乘法、加法和等号符号则从 HASY 中随机抽取。需要注意的是，HASY 不包含标准等号“=”，因此我们改用“≒”。随后，将这些图像按适当顺序水平拼接。

我们针对该问题评估了两种架构。第一种使用与 MNIST 加法相同的神经网络，以及一个SMTLayer 实例，其约束如示例2所示。它假设问题中的四个数字已被提取，例如由一个能够区分字母和算术符号并识别数字的独立视觉识别程序提取，然后直接提供给模型。四个输入分别送入神经网络，生成四个 4 位位向量；将它们拼接后传入SMTLayer，由其生成一个 4 位位向量结果。我们在结果中将其称为配置 #1。

第二种架构将包含该问题的整幅图像作为一个整体输入，并生成一个 16 位位向量，直接传递给 SMTLayer。该架构使用了类似的卷积层堆栈，但由于输入图像更大，因此包含一个更大的初始全连接层，其中有 26,112 个神经元。卷积层堆栈的区别在于第二层，该层的宽度步幅同样为 2，以减小特征图的大小，从而避免需要更大的全连接结构。这些架构之间的差异与该问题的挑战之一有关。特征中包含的大量信息与解决方案无关，例如，变量选择哪个字母并不重要，算术运算符呈现何种形状也不重要，因此该架构还必须学会忽略实例中的这些部分。我们在结果和示例 3 中将其称为配置 #2。

我们通过从相同数字的数对中均匀选择 $a$ 和 $b$ 来生成训练样本，即 (0，0)，(1，1) .....，并从 0 到 9 之间的奇数中均匀采样 $x$。测试样本则通过从所有数字对中均匀采样 $a$，$b$，并从 0 到 9 的所有数字中采样 $x$ 来生成。随后，我们将这些值映射到随机的 MNIST、EMNIST 和 HASY 图像，以获得 60,000 个样本。我们的意图是研究这样一个问题：模型并未看到所有可能的问题（将数字作为表征时的模意义下），也未看到所有解的示例。这比 MNIST 加法更具挑战性，原因有二：对于给定的解，存在多得多的兼容基项，而且模型看不到测试集所需提供的某些解的示例。因此，为了使 SMTLayer 成功，它必须利用所提供的符号知识来近似正确的基化函数，尽管数据存在这些缺陷。

**说谎者谜题**。 说谎者谜题由三个不同参与者 Alice、Bob 和 Charlie 说出的三个句子组成。其中一名参与者犯有某种未指明的罪行，而在每个句子中，相应的参与者都会声称另一方有罪或无辜。假设其中两名参与者诚实，而有罪方则不诚实。问题的解答就是找出有罪方。示例见示例4

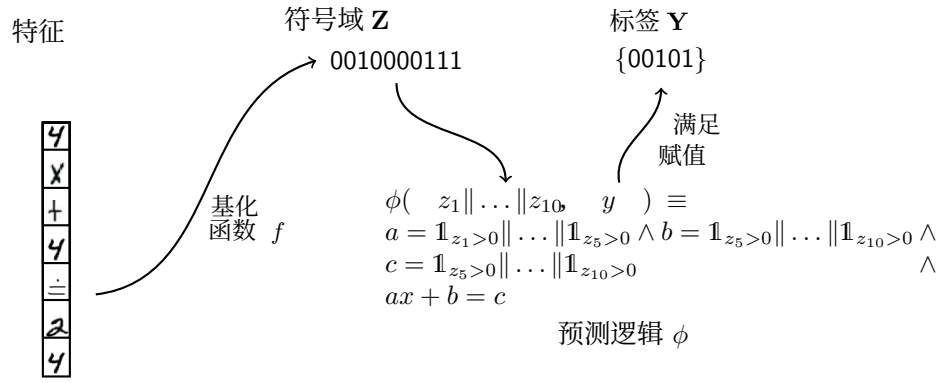



图 *2.* 视觉代数配置 1 示例。

我们根据有限的话语集合合成了一个说谎者谜题数据集，内容涉及每个句子由谁说出、哪个参与者是主语，以及主语是否有罪或无辜。表示说话者的方式有五种：“* 说”、“* 说……”、“* 说了”、“* 说了……”，以及用冒号“*：”将说话者的姓名与句子其余部分分开。表达无辜或有罪的方式有五种：“* 做了／没做这件事”、“* 有罪／无辜”、“* 肯定有罪／无辜”、“* 肯定做了／没做这件事”、“* 是罪犯／是个好人”。我们生成了主语、说话者以及所声称的无辜或有罪情况的所有组合，并将其与这些话语的所有可能组合进行笛卡尔积。最终得到一个包含 375,000 个实例的数据集，每个实例都包含三个自然语言句子。

此问题的预测逻辑假设存在一组基底谓词 speaker(agent, sentence)、subject(agent,sentence)、accuses(sentence) 和 guilty(agent)。例如，如果第一句话是“Alice 说 Bob 是无辜的”，那么基底谓词将是 speaker(alice, 1)、subject(bob, 1) 和 $\neg$guilty($bob$)。然后，预测逻辑如方程 1 所示。

$$
\begin{aligned}
& |\{a : \mathsf{guilty}(a)\}| = 1 \\
\wedge \quad & \forall a.|\{s : \mathsf{speaker}(a, s)\}| = 1 \\
\wedge \quad & \forall s.|\{a : \mathsf{subject}(a, s)\}| = 1 \\
\wedge \quad & \forall s, a_1, a_2.\mathsf{speaker}(a_1, s) \wedge \mathsf{subject}(a_2, s) \rightarrow \\
& \quad (\neg\mathsf{guilty}(a_1) \leftrightarrow \mathsf{accuses}(s) \leftrightarrow \mathsf{guilty}(a_2))
\end{aligned}
\tag{1}
$$

在我们的实现中，参与者和句子标识符均以一元编码表示为 3 位位向量。通过针对每个基底项或句子标识符进行替换，消除量词，并将基数约束展开为命题公式。我们采用的架构是双层双向长短期记忆网络（LSTM），每层维度为 512，并配有一个 300 维的可训练嵌入层，该层以 GloVe-6B 初始化（Pennington 等人，2014）。最后一个 LSTM 层的隐藏单元连接到一个包含两层的全连接网络，第一层有 128 个神经元，第二层有 7 个神经元。该网络应用于输入中的每个句子，拼接后的结果被传递给 SMTLayer，由其求解公式 1 ，生成有罪方的一元编码。

为了评估求解器层在此问题上的表现，我们首先从全部 375,000 个可用实例中抽取一半，再据此选取训练样本和测试样本。随后，我们将该子样本中三个句子所涉及的 说话者、主语、指控 谓词配置各自的一半划入训练样本，另一半划入测试样本。为了进一步限制训练样本中的信息量，我们为每种谓词配置随机选取一种排列，仅保留在训练集中。最终得到 9,400 个训练实例和 28,062 个测试实例。按上述方式限制训练集，可确保模型仅在可能句子配置的有限子集上进行训练，且这些配置在逻辑上与测试样本中的配置不相交。由于训练样本中的信息不足以学习公式 1，我们预计只有带有 SMTLayer 的模型能够成功；但要做到这一点，它必须根据有限的样本很好地近似基化函数。

### A.3. 超参数

我们评估的四个问题在规模和复杂度上差异很大，用于训练这些问题的模型也需要采用不同的考量。本节将详细介绍这些差异。表2列出了各个

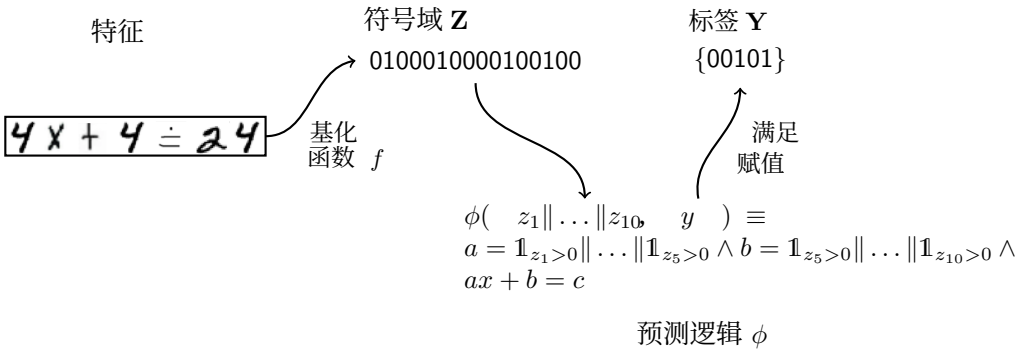



图 *3.* 视觉代数配置 2 示例。

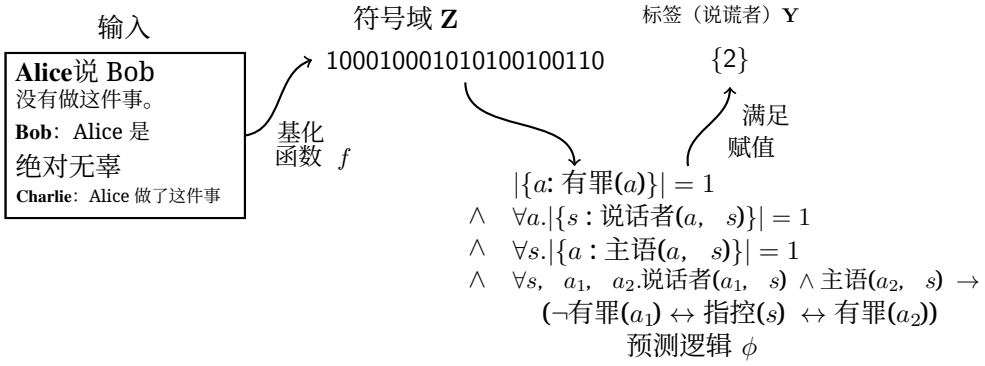



图 *4.* 说谎者谜题示例。

数据集以及求解器层配置。对于 SATNet 和 Scallop，我们采用其各自论文中所述、并见于其公开实现中的优化设置。对于 SATNet 模型，MaxSAT 子句参数的训练速率为 2e-3，卷积网络的训练速率为 1e-5。当注明 SGD(1.0) 时，我们在第一个训练轮次采用预热期，之后的训练则使用余弦退火。

**MNIST 加法**。 MNIST 加法问题是我们研究的问题中最简单的，至少在其完整（100%）配置下是如此。我们发现，对于 100% 配置，所有求解器层经过三轮监督式预训练，以及随后五轮附加求解器层的训练后，都会收敛到近乎最优的解。对于子采样配置，在这些参数下，所有求解器层也都会收敛到稳定的解，尽管许多情况下并非最优。对于 SMTLayer，我们将所有参数的梯度裁剪至 0.1，而其他求解器层模型的梯度则不进行裁剪。所有配置均使用大小为 128 的批次。

**视觉代数**。 尽管视觉代数比 MNIST 加法更难学习，表 1中的结果也证明了这一点，但我们发现，相同的参数足以使评估中研究的所有配置都收敛。训练 3/5 轮后，模型趋于稳定；在某些情况下，进一步训练会导致模型过拟合。对于 SMTLayer，我们将所有参数的梯度裁剪至 0.1，而其他求解器层模型的梯度则不进行裁剪。所有配置均使用大小为 64 的批次。

**说谎者谜题**。 说谎者谜题是唯一使用循环模型的问题。我们发现，为了降低最终带有求解器层的模型的方差，需要进行更多轮次的预训练。此外，在其他数据集上将随机梯度下降优化器与 SMTLayer 搭配使用，会导致模型收敛到局部极小值。我们发现，使用较高的学习率进行预训练，并使用默认学习率的 Adam，可以取得最佳结果。此外，我们没有对 SMTLayer 模型的梯度进行裁剪。SATNet 模型采用相同的参数，但使用常规的 SATNet 优化器。对于所有配置，我们使用大小为 32 的批次。

| | 传统方法 | | 搭配 SMTLayer | | 搭配 SATNet | | 搭配 Scallop | |
|---|---|---|---|---|---|---|---|---|
| | 优化器 | 训练轮次 | 优化器 | 训练轮次 | 优化器 | 训练轮次 | 优化器 | 训练轮次 |
| MNIST+ 10% | 随机梯度下降(1.0) | 0/5 | 随机梯度下降(1.0) | 3/5 | Adam(2e-3, 1e-5) | 3/5 | Adam(1e-3) | 3/5 |
| MNIST+ 25% | 随机梯度下降(1.0) | 0/5 | 随机梯度下降(1.0) | 3/5 | Adam(2e-3, 1e-5) | 3/5 | Adam(1e-3) | 3/5 |
| MNIST+ 50% | 随机梯度下降(1.0) | 0/5 | 随机梯度下降(1.0) | 3/5 | Adam(2e-3, 1e-5) | 3/5 | Adam(1e-3) | 3/5 |
| MNIST+ 75% | 随机梯度下降(1.0) | 0/5 | 随机梯度下降(1.0) | 3/5 | Adam(2e-3, 1e-5) | 3/5 | Adam(1e-3) | 3/5 |
| MNIST+ 100% | 随机梯度下降(1.0) | 0/5 | 随机梯度下降(1.0) | 3/5 | Adam(2e-3, 1e-5) | 3/5 | Adam(1e-3) | 3/5 |
| 可视化算法 #1 | 随机梯度下降(1.0) | 0/5 | 随机梯度下降(1.0) | 3/5 | Adam(2e-3, 1e-5) | 3/5 | Adam(1e-3) | 3/5 |
| 视觉算法 #2 | 随机梯度下降(1.0) | 0/5 | 随机梯度下降(1.0) | 3/5 | Adam(2e-3, 1e-5) | 3/5 | Adam(1e-3) | 3/5 |
| 说谎者谜题 | Adam（2e-3） | 0/15 | Adam（1e-3） | 15/5 | Adam（2e-3，1e-5） | 15/5 | — | — |
| 可视化数独 10% | 随机梯度下降（1.0） | 0/100 | 随机梯度下降(1.0) | 30/15 | Adam(2e-3, 1e-5) | 30/100 | — | — |
| 可视化数独 50% | 随机梯度下降(1.0) | 0/100 | 随机梯度下降(1.0) | 30/5 | Adam(2e-3, 1e-5) | 30/100 | — | — |
| 可视化数独 100% | 随机梯度下降(1.0) | 0/100 | 随机梯度下降(1.0) | 30/5 | Adam(2e-3, 1e-5) | 0/100 | — | — |

表2。 训练超参数。优化器后括号中的数字表示学习率；如果有多个数字，则对不同参数组应用了不同的学习率，详见正文。每对训练轮次对应预训练和训练阶段，即 3/5 表示先进行 3 个训练轮次的预训练，再使用求解器层进行 5 个训练轮次的训练。

**视觉数独**。 在我们研究的所有问题中，视觉数独最具挑战性，无论是对于所有求解器层还是传统模型都是如此。我们没有使用监督式预训练，因为该问题中的监督会将正确标签直接泄露给模型，从而绕过求解器层及其更新需求（Chang 等人，2020年）。相反，我们采用了（Topan 等人，2021年）中描述的无监督预训练方法，并发现进行 30 轮次的无监督预训练，足以使求解器层稳定、快速地收敛。对于数据最稀缺的配置（10%），求解器层需要训练 15 轮次才能收敛，其他配置则训练五轮次即可。对于SMTLayer 模型，预训练后我们使用大小为 1 的批次（预训练期间批次大小为 64），主要是因为SMTLayer 只会返回数独棋盘上被标记为非提示元素的索引。由于每个实例中的提示元素数量各不相同，这会导致训练期间出现不规则张量，而 PyTorch 不支持这种张量。

在评估 SATNet 在视觉数独上的表现时，除完整（100%）配置外，我们无法收敛到有用的模型，详见第 5.2 节。我们使用作者公开的实现和训练脚本，尝试了 10% 和 50% 配置，分别采用和不采用无监督预训练，分别采用和不采用（Chang 等人，2020年）中为防止标签泄漏而采取的措施，并分别使用大小为 40（与原论文相同）和 1 的批次，但均未能成功。对于 100% 配置，我们成功复现了一个有用的模型；为保持一致，我们在表 1 中采用原论文报告的准确率，因为我们得到的平均准确率与之没有显著差异。