arXiv:2301.03765v2 [cs.CL] 9 Mar 2024

用于提升语言理解中神经元效用的跨模型比较损失

YUNCHANG ZHU，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学中国，中国
LIANG PANG*，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所，中国
KANGXI WU，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学中国科学院，中国
Yanyan Lan, 清华大学人工智能产业研究院，中国
Huawei Shen, 中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学中国科学院
Xueqi Cheng,中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学中国科学院

当前，自然语言理解（NLU）模型在模型规模和输入上下文方面持续扩展，从而引入了更多隐藏神经元和输入神经元。虽然这通常能提升平均性能，但额外的神经元并不能始终改善所有实例的表现。这是因为一些隐藏神经元具有冗余性，而混入输入神经元的噪声往往会分散模型的注意力。以往的工作主要通过额外的后处理或预处理来从外部减少效用较低的神经元，例如网络剪枝和上下文选择，以避免这一问题。除此之外，我们能否通过从内部提升每个神经元的效用，让模型减少冗余参数并抑制输入噪声？如果模型能够高效利用神经元，那么无论消融（禁用）哪些神经元，消融子模型的表现都不应优于原始完整模型

本文是 SIGIR 2022 会议论文 [80] 的扩展版本。早期的会议论文仅通过比较使用不同大小反馈集时的检索损失，来解决伪相关反馈中的查询漂移问题。然而，比较原则和比较损失实际上具有通用性，且与任务无关。此外，除了比较模型的输入，我们还可以比较模型的参数。因此，在本研究中，我们（1）给出了更通用、更完整的比较原则和比较损失表述；（2）直接使用统一的比较损失作为最终优化目标，从而无需在比较正则化项与任务特定损失之间设置权重系数；（3）改进了此前通过比较不同上下文大小的输入进行比较的方法，并提出一种替代性的基于 Dropout 的比较方法，以提高参数对模型的效用；（4）将比较损失应用于更多任务和模型，并通过实验证明其普遍有效性。

*通讯作者

本研究得到中国国家重点研发计划（2022YFB3103700、2022YFB3103704）、国家自然科学基金委员会（NSFC）项目（编号：62276248、U21B2046）以及中国科学院青年创新促进会项目（编号：2023111）的资助。作者地址：Yunchang Zhu，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学，中国北京，zycdev@gmail.com；Liang Pang，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所，中国北京，pangliang@ict.ac.cn；Kangxi Wu，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学，中国北京，wukangxi22s@ict.ac.cn；Yanyan Lan，人工智能产业研究院、清华大学，中国北京，lanyanyan@tsinghua.edu.cn；Huawei Shen，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学，中国北京，shenhuawei@ict.ac.cn；Xueqi Cheng，中国科学院人工智能安全重点实验室、中国科学院计算技术研究所；中国科学院大学，中国北京，cxq@ict.ac.cn。

允许为个人或课堂使用而免费制作本作品的全部或部分数字副本或纸质副本，前提是不得为营利或商业利益制作或分发副本，且副本须附有本声明及首页上的完整引文。本作品中归他人而非 ACM 所有的组件，其版权必须得到尊重。注明出处后允许摘录。以其他方式复制、再版、发布到服务器或分发至列表，均须事先获得具体许可和/或支付费用。请向 permissions@acm.org 申请许可。© 2018年 美国计算机协会。提交给 ACM 的稿件

模型。基于模型之间的这种比较原则，我们提出了一种适用于多种任务的跨模型比较损失。比较损失本质上是在完整模型和消融模型的任务特定损失之上进行排序，并期望完整模型的任务特定损失最小。我们基于 5 种广泛使用的预训练语言模型，在 3 类不同的自然语言理解任务的 14 个数据集上进行了大量实验，证明了比较损失具有普遍的有效性；我们发现，它对于参数较少或输入较长的模型尤其有效。

CCS 概念： • **信息系统** → **信息检索查询处理**；**检索模型与排序**；**问答**；**聚类与分类**。

其他关键词 词语和短语：自然语言理解、问答、伪相关反馈、损失函数

ACM 参考格式：
Yunchang Zhu、Liang Pang、Kangxi Wu、YanyanLan、Huawei Shen 和 Xueqi Cheng。2018年。《用于增强语言理解中神经元效用的跨模型比较损失》。*ACM* 信息系统汇刊37卷，第4期，文章111（2018年8月），28 页。
https://doi.org/XXXXXXX.XXXXXXX

1 引言

深度神经模型推动自然语言理解（NLU）取得了显著进展。为了进一步提升深度模型的性能，扩大模型规模 [7, 8, 33, 57] 和增加输入上下文 [6, 32, 67] 是两种常见且有效的方法：前者会引入更多隐藏神经元，后者则会带来更多输入神经元。尽管隐藏神经元或输入神经元更多的神经网络模型平均而言准确率更高，但大规模模型并非总是优于小型模型。例如，一方面，许多网络剪枝方法表明，大幅减少参数（神经元连接）的压缩模型仍能保持准确率 [27, 39, 45]，甚至提高泛化能力 [2]. 。Meyes 等人 [48] 发现，在某些特定类别中，消融神经元能够持续提升性能；Zhong 等人 [77] 通过实证证明，较大的语言模型确实会在相当一部分实例上表现更差。这些现象表明，当前训练的模型中有些隐藏神经元是可有可无的，甚至会产生阻碍。另一方面，许多关于问答 [14, 74]和查询理解 [18, 50, 81] 的研究指出，输入更多上下文信息更可能分散模型的注意力并损害性能。这并不令人意外，因为更多输入神经元不仅意味着更多相关特征，也可能引入更多干扰模型的噪声。与通过后处理去除低效参数的网络剪枝类似，许多上下文选择方法 [23, 49, 62, 76] 通过预处理从输入上下文中删去噪声片段。从本质上说，网络剪枝和上下文选择都是通过额外处理来减少低效的隐藏神经元或输入神经元。然而，除了从外部减少低效神经元之外，我们能否在模型训练过程中从内在提升神经元的效用？

设想一个理想的神经网络，其中所有神经元都应能够高效协作，以最大限度地发挥每个神经元的效用。如果该网络中的一部分输入神经元或隐藏神经元被消融[1]（即禁用部分输入上下文或模型参数），那么即使被消融的神经元带有噪声，消融子模型的表现也不应更好。这是因为高效的[2] 模型理应已经抑制了这些噪声。遵循这一思路，我们可以大致得出原始完整模型与其消融模型之间的**比较原则**：**模型中被消融的神经元越少，模型的表现就应越好**。在训练期间，我们可以用任务特定损失作为模型在训练样本上表现的代理指标，任务特定损失越低，意味着表现越好。例如，图1 中的高效完整模型 (a) 的任务特定损失理应最小；如果消融

[1]神经元的输出值设为 0，这相当于将与该神经元相连的所有连接权重设为 0。[2]本文中，“高效”特指神经元具有较高的效用。

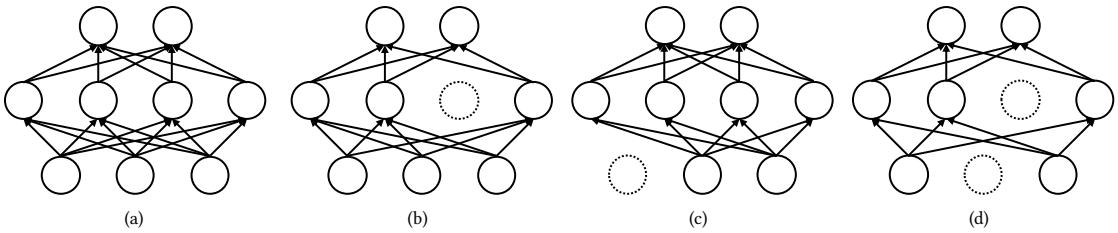


图 1. 完整神经网络模型 (a) 及其消融模型 (b、c 和 d) 的示意图：在 (b) 中消融了一个隐藏神经元，在 (c) 中消融了一个输入神经元，而 (d) 则在 (b) 的基础上额外消融了另一个输入神经元。根据比较原则，如果完整模型 (a) 是一个高效模型，那么这些模型得到的任务特定损失之间的比较关系应为 (a) ≤ (b)、(c)、(d)。如果消融模型 (b) 在其参数空间中也高效，那么它们之间的比较关系还可进一步写为 (a) ≤ (b) ≤ (d)。请注意，(b, c) 和 (c, d) 是两对不可比较的模型。这是因为消融模型 (c) 不是 (b) 和 (d) 的子模型，反之亦然。

模型 (b) 在其受限参数空间内同样高效，那么消融模型 (d) 的任务特定损失理应高于 (b)，因为 (d) 比 (b) 多消融了一个输入神经元。

注意到理想模型与现实之间的差距 [49, 77], 我们旨在通过在训练过程中确保这一必要性（比较原则），提高模型对神经元的利用率。基于模型之间自然存在的比较原则，我们提出一种跨模型比较损失，无需额外的人工监督即可训练模型。一般而言，比较损失是在多个任务特定损失之上构建的排序损失。首先，这些任务特定损失来自完整神经网络模型以及若干可比较的消融模型，这些模型中的神经元经过了不同程度的消融。其次，排序损失是一种成对铰链损失，用于惩罚消融神经元较少但任务特定损失较大的模型。具体而言，如果消融神经元较少的模型所产生的任务特定损失大于另一个消融神经元较多的模型，那么这对模型的任务特定损失之差将计入最终的比较损失；否则，这对模型符合比较原则，不会产生任何训练损失。通过这种方式，比较损失可以促使任务特定损失的排序与消融程度的排序保持一致。通过理论推导，我们还表明，比较损失可以视为对多个任务特定损失进行动态加权，从而能够根据完整模型／消融模型的表现自适应地分配权重。

多个消融模型之间的可比较性是比较损失的基本前提。举一个反例，尽管图1 中的消融模型 (c) 消融的神经元少于 (d)，但二者并不可比较，因此无法应用比较损失。为了使各消融模型彼此可比较，我们逐步对模型进行消融。第一个消融模型是在完整模型的基础上进行一次消融得到的。如果需要更多消融模型，则在后续每个消融步骤中，我们都会在前一步得到的消融模型基础上进一步消融，从而确保新得到的消融模型是先前模型的可比较子模型。我们为每个消融步骤提供两种备选的受控消融方法，分别称为 CmpDrop 和 CmpCrop。CmpDrop 通过随机失活 [30] 技术消融隐藏神经元，理论上适用于所有兼容随机失活的模型。CmpCrop 则通过裁剪多余的上下文片段来消融输入神经元，理论上适用于输入上下文中包含多余内容的所有任务。

我们在 14 个数据集上应用比较损失，并结合 CmpDrop 和/或 CmpCrop。这些数据集涵盖 3 项自然语言理解任务（文本分类、问答和查询理解），且具有不同的预测类型（分类、抽取和排序）

基于 5 个广泛使用的预训练语言模型（PLM） [3, 15, 21, 37, 44]。实证结果表明，相较于最先进的基线方法，比较损失是有效的，同时也能提升参数和上下文的效用。我们的分析还证实，正如推导所示，比较损失确实能够以更恰当的方式为多个任务特定损失分配权重。通过探索不同的比较策略，我们发现，先使用 CmpCrop、再使用 CmpDrop 进行消融的模型之间进行比较，能够带来最大的提升。有趣的是，我们发现比较损失对于参数较少或输入较长的模型尤其有效。这可能意味着，比较损失有助于容量较低的模型更好地拟合更多或更长的训练样本；而容量较高的模型本身就容易拟合较少的数据，因此比较损失带来的帮助较小。此外，我们还发现，不同的消融方法对训练的影响各不相同：CmpDrop 能帮助任务特定损失更快地降至较低水平，而 CmpCrop 则能在一定程度上缓解过拟合。

主要贡献可概括如下：

- 我们提出了比较损失，这是一种基于完整模型与其消融模型之间比较原则的跨模型损失函数，旨在提升神经元效用，且无需额外的人工监督。
- 我们逐步对模型进行消融，使多个消融模型具有可比性，并提出两种基于随机失活和上下文裁剪的受控消融方法，适用于广泛的任务和模型。
- 我们从理论上阐释了比较损失的工作原理，并通过在 3 种不同的自然语言理解任务上开展实验，从实证角度证明了其有效性。我们已在此发布代码和处理后的数据：https://github.com/zycdev/CmpLoss。

2 预备知识

在介绍我们的跨模型比较损失之前，我们先回顾一些后文会用到的概念和符号。首先介绍模型的典型训练方法，随后形式化定义网络剪枝和上下文选择方法，这些方法可以通过移除低效的输入或隐藏神经元，进一步提升模型性能。最后，我们将详细阐述消融的概念，该概念在本文中反复出现。

2.1 常规训练

给定指定任务的训练数据集 D，以及一个由 $\boldsymbol{\theta} \in \mathbb{R}^{|\boldsymbol{\theta}|}$ 参数化的神经网络 f，对于每个样本（x， y） $\in$ D，训练目标是最小化经验风险

$$\mathcal{L}_{\text{emp}}(x, y, \boldsymbol{\theta}) = L(y, f(x; \boldsymbol{\theta})), \tag{1}$$

其中， x 表示输入上下文， y 属于输出空间 $\mathcal{Y}$ 的标签，而 $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ 是任务特定损失函数，$\mathbb{R}_{\geq 0}$ 表示非负实数集合。在自然语言理解任务中， x 通常是一个词序列，而 y 可以是用于分类的单个类别标签 [55, 66, 73], , 也可以是用于抽取的一对起始和结束边界 [54, 59, 74],, 或用于排序的相关性等级序列 [51, 52, 56, 78]。

2.2 网络剪枝

训练神经网络模型 $f(x; \boldsymbol{\theta})$ 后，为了降低测试时的内存和计算需求，网络剪枝 [5]通过事后处理生成一个准确率相近的小型模型 $f(x; \boldsymbol{m} \odot \boldsymbol{\theta}\prime)$。这里， $\boldsymbol{m} \in \{0, 1\}^{|\boldsymbol{\theta}|}$是一个二值掩码，它通过逐元素乘积 $\odot$将特定的剪枝参数固定为 0；参数向量 $\boldsymbol{\theta}\prime$可能不同于 $\boldsymbol{\theta}$，因为通常会从 $\boldsymbol{m} \odot \boldsymbol{\theta}$ 重新训练 $\boldsymbol{m} \odot \boldsymbol{\theta}\prime$， 以适应剪枝后的网络结构。

尽管剪枝通常被视为一种压缩模型的方法，但其动机也包括防止过拟合。剪枝会系统地移除对性能贡献不显著的冗余参数和神经元，因此这些参数和神经元的预测方差也小得多，这让人联想到随机失活 [36], ——另一种广泛用于避免过拟合的技术。同样，随机失活也会使用掩码禁用一定比例（例如 $p\%$）的参数或神经元。不过，两者的显著区别在于，随机失活中的掩码 $\boldsymbol{m}$ 是从一个伯努利$(1-p\%)$ 分布中随机采样得到的，而不是像剪枝那样根据某个准则确定（例如，将幅值最低的 $p\%$ 参数设为掩码）。这样也带来了便利：使用随机失活训练的模型无需针对特定掩码重新训练，因为在先前的训练过程中，模型的神经元已经开始学会适应部分神经元缺失的情况。

2.3 上下文选择

为了消除输入上下文 x 中的噪声内容并进一步提升模型性能，上下文选择会选择性地裁剪出精简的上下文 $x\prime \sqsubseteq x$，以生成最终的模型预测。通常，模型需要经过专门训练才能适应所选上下文。因此，相对于训练而言，上下文选择属于事前处理，需要预先从训练样本中去除噪声。稍微滥用一下符号表示法，我们在此用$x\prime \sqsubseteq x$ 表示 $x\prime$ 是 x的一个精简子序列（也可能与之相等）。通常， $x\prime$ 是 x中若干片段的有序组合，这些片段通常以句子 [49, 53], 、文本块 [76], 、段落 [14], 或文档 [23, 62]为粒度。值得注意的是，用于选择片段的选择器通常需要额外的监督训练，并且需要在预测之前运行，这会带来额外的计算开销。

2.4 消融

为了评估某些组件对模型整体的贡献，消融研究会在受控环境下，通过移除或替换这些组件来考察模型的行为 [17]。在此，机器学习语境中的“消融”指移除模型的组件，这与生物学中消融性脑外科手术（移除生物体的组件）类似 [48]。我们将移除组件后的模型称为“消融模型”，它应能继续正常运行。不过，如果被移除的组件有助于提升性能，那么消融模型的性能预计会更差 [24]。

在本文中，我们用“消融”专指移除神经网络模型中的部分神经元，即将某些特定神经元的输出设为零。从神经元的角度来看，网络剪枝和上下文选择可视为两种消融：前者在训练后移除贡献较低的隐藏神经元，后者则在训练前移除信息量较低的输入神经元。然而，与旨在考察被消融神经元作用的消融研究不同，我们旨在学习如何提升被消融神经元的效用。

3 方法论

这项工作的主要动机是通过跨模型训练目标，从根本上提升自然语言理解模型中神经元的效用，而不是事后进行网络剪枝，或事先进行上下文选择来剔除低效神经元。下文将首先介绍一种比较原则。然后，我们基于比较原则的推论提出一种新颖的比较损失，并介绍如何通过两种受控消融方法，使用比较损失训练模型。最后，我们讨论比较损失的作用机制。

3.1 比较原则

对于一个高效的模型，我们认为其中所有神经元都应能高效协作，以最大化每个神经元的效用。这意味着每个神经元都应对整体模型有所贡献，或者至少不造成损害，因为神经元之间的协作理应消除个别神经元可能产生的噪声带来的负面影响。因此，如果我们消融某些神经元，即使这些神经元会产生噪声，由于消融神经元的贡献缺失，消融后的子模型表现也不应优于原始完整模型；换言之，其任务特定损失应不小于原始模型。

形式上，当且仅当某神经网络模型的表现不弱于其任何消融模型时，我们才将其定义为高效模型；我们将高效模型与其消融模型之间的比较原则形式化如下。

比较原则。假设 $f(x;\ \boldsymbol{\theta})$ 是针对输入 x、相对于参数空间 $\mathbb{R}^{|\boldsymbol{\theta}|}$ 的高效神经网络模型，令 $x' \sqsubset x$ 为消融输入，$\boldsymbol{\theta}' = \boldsymbol{m} \odot \boldsymbol{\theta}$ 为消融参数。那么，对于 x 的任意子序列 x'，只要其标签仍为 y，输入消融模型 $f(x';\ \boldsymbol{\theta})$ 的表现就不应优于完整模型 $f(x;\ \boldsymbol{\theta})$；对于被任意 $\boldsymbol{m}$ 掩码的任意参数 $\boldsymbol{\theta}'$，参数消融模型 $f(x;\ \boldsymbol{\theta}')$ 的表现也不应优于 $f(x;\ \boldsymbol{\theta})$，即，

$$L(y, f(x;\boldsymbol{\theta})) \leq L(y, f(x';\boldsymbol{\theta})), \forall x' \sqsubset x \ with\ g(x') = y, \tag{2}$$

$$L(y, f(x;\boldsymbol{\theta})) \leq L(y, f(x;\boldsymbol{\theta}')), \forall \boldsymbol{\theta}' = \boldsymbol{m} \odot \boldsymbol{\theta}\ with\ \boldsymbol{m} \in \{0,1\}^{|\boldsymbol{\theta}|}, \tag{3}$$

其中 g（x，）表示 x，的真实输出。

在上述定义中，我们认为高效的神经网络模型应当兼具输入高效和参数高效的特性。具体而言，输入高效是指模型能够高效利用输入神经元（词）。如果模型 $f(x;\ \boldsymbol{\theta})$ 满足式（2），我们就称 $f(\cdot;\ \boldsymbol{\theta})$ 对输入 x 是输入高效的。参数高效是指模型能够高效利用隐藏神经元。如果模型 $f(x;\ \boldsymbol{\theta})$ 满足式（3），我们就称 $f(x;\ \boldsymbol{\theta})$ 关于参数空间 $\mathbb{R}^{|\boldsymbol{\theta}|}$ 对输入 x 是参数高效的。根据式（3），我们肯定能找到至少一个对输入 x 参数高效的向量 $\boldsymbol{\theta}$，即零向量和使经验风险最小化的最优参数向量。如果参数空间足够大，那么在那些对 x参数高效的向量中，我们可以找到一些同时满足式（2）的参数，也就是具有输入高效特性的参数。换言之，如果参数空间 $\mathbb{R}^{|\boldsymbol{\theta}|}$ 足够大，就存在至少一个参数向量 $\boldsymbol{\theta}$，使模型 $f(x;\ \boldsymbol{\theta})$ 对 x 高效。特别地，如果神经网络模型 f 中的所有激活函数在输入为零时的输出值都为零，那么 $f(x';\ 0) = f(x;\ 0)\ \forall x' \sqsubset x$，因此参数向量 $\boldsymbol{\theta} = \mathbf{0}$ 对任意 x都高效。

值得注意的是，为进行比较，我们将消融输入 x'限制为仅包含真实输出$g(x')$保持不变的那些子序列，即 $g(x') = g(x) = y$。这是因为消融可能会从原始输入 x 中移除一些关键信息，例如分类任务中的触发词，从而导致标签 y发生未知变化。在这种特殊情况下，$L(y, f(x';\ \boldsymbol{\theta}))$ 将不再是输入消融模型性能的合理代理指标，因此将其与原始模型的任务特定损失进行比较毫无意义。例如，在二元分类（$y \in \{0,1\}$）中，对于原始输入 x，$f(x;\ \boldsymbol{\theta})$ 以较低置信度预测出正确类别 y；而对于类别标签变为 $g(x') = 1 - y$ 的消融输入 x'，$f(x';\ \boldsymbol{\theta})$ 则以较高置信度预测出 $1 - y$。即使 $L(y, f(x;\ \boldsymbol{\theta})) \leq L(y, f(x';\ \boldsymbol{\theta}))$，输入消融模型 $f(x';\ \boldsymbol{\theta})$ 实际上优于原始模型 $f(x;\ \boldsymbol{\theta})$，即 $L(y, f(x;\ \boldsymbol{\theta})) \geq L(1 - y, f(x';\ \boldsymbol{\theta}))$，我们也不能认为 $f(\cdot;\ \boldsymbol{\theta})$ 对 x 是输入高效的。尽管我们可以使用 $L(g(x')\ f(x';\ \boldsymbol{\theta}))$ 作为式 (2) 中输入消融模型的性能代理指标，但在实际应用中，很难知道消融输入的标签会如何变化，因此我们会尽量避免这类标签变化情形。为简洁起见，提交给 ACM 的稿件

除非另有说明，以下默认对输入上下文进行消融不会改变输出标签，即 $g(x') = y$。

此外，我们可以对完整模型 $f(x(0); \theta(0))$ 进行多次消融（c 次），但这些消融模型 $\{f(x(i); \theta(i))\}_{i=1}^{c}$ 彼此之间是否可比较？比较原则只指出高效模型与其任一消融模型之间的比较关系，不能直接用于多个消融模型。然而，若我们假设这些消融模型是逐步构建的，即每个消融模型 $f(x(j); \theta(j))$ 都是基于其前一个模型 $f(x(j-1); \theta(j-1))$，逐步消融输入（$x(j) \sqsubset x(j-1)$）或参数（$\theta(j) = \boldsymbol{m}(j) \odot \theta(j-1)$）得到的，那么 $f(x(j); \theta(j))$ 就可以视为其所有祖先模型 $\{f(x(i); \theta(i))\}_{i=0}^{j-1}$ 的消融模型。为简便起见，我们将它们的任务特定损失简记为 $l(i) = L(y, f(x(i); \theta(i)))$。此外，若同时假设所有 $\{f(x(i); \theta(i))\}_{i=0}^{c-1}$ 相对于其参数空间 $\mathbb{R}^{\|\boldsymbol{m}^{(i)}\|_0}$ [3] 都是高效的，我们便可应用比较原则来比较任意两个模型的任务特定损失，即 $l(i) \le l(j)\ \forall i < j$。

形式上，我们将高效模型定义为遗传高效，当且仅当其所有消融模型都是高效的。类似地，如果参数高效模型的参数消融模型全都是参数高效的，我们就称该参数高效模型具有遗传参数高效性。遗传输入高效模型也以相同方式定义。具体而言，参数向量 $\theta = \mathbf{0}$ 具有遗传参数高效性；对于任意 x，如果 f 的所有激活函数在输入为零时的输出值都为零，那么 $f(x;\mathbf{0})$ 也具有遗传高效性。

根据遗传高效模型的定义和比较原则，我们可以得出以下推论。

推论 *3.1.* 假设 $f(x(0); \theta(0))$ 是关于参数空间 $\mathbb{R}^{|\theta(0)|}$、针对输入 $x(0)$ 的遗传高效神经网络模型，设 $\{f(x(i); \theta(i))\}_{i=1}^{c}$ 是其多个逐步消融模型，其中 $x(i) \sqsubset x(i-1)$ 或 $\theta(i) = \boldsymbol{m}(i) \odot \theta(i-1)$。那么，所有 $\{f(x(i); \theta(i))\}_{i=1}^{c-1}$ 也都是相对于各自参数空间 $\mathbb{R}^{\|\boldsymbol{m}^{(i)}\|_0}$ 的高效模型，并且其任务特定损失应随着消融程度的增加而单调不减，即，

$$l^{(0)} \le l^{(1)} \dots \le l^{(i)} \dots \le l^{(c)}. \tag{4}$$

简而言之，该推论描述了一个理想的传递性比较：遗传高效的神经网络模型与其消融模型之间，消融越少，性能越好。遗憾的是，这一自然属性此前在很大程度上被忽视了，这促使我们利用它来训练能够更高效地利用神经元的模型。

3.2 比较损失

基于推论 3.1，我们可以通过优化式 (4) 中的有序比较关系目标，训练一个遗传高效模型。为衡量与理想顺序之间的差异，我们可以使用成对铰链损失 [29]，评估完整模型及其消融模型的任务特定损失的排序，例如 $\sum_{i=0}^{c-1}\sum_{j=i+1}^{c}\max(0, l(i) - l(j))$。然而，仅优化这一排序损失并不能保证这些任务特定损失达到最小，也就是说，完整模型/消融模型相对于其参数空间可能并未实现经验风险最小化（ERM）[63]。为使这些模型达到经验风险最小化，我们引入一个特殊标量 b 作为任务特定损失的基线值，并假设该值来自一个虚拟消融模型 $f(x(c+1); \theta(c+1))$。该虚拟模型被设定为具有最高程度的消融，原则上，其任务特定损失 $l(c+1)$ 应当最高。然而，为了降低真实模型的任务特定损失 $\{f(x(i)\,\theta(i))\}_{i=0}^{c}$，我们通常将 $l(c+1) = b$ 设为较小值（例如 0），并期望所有 $\{l(i)\}_{i=0}^{c}$ 都因这一目标而降低。如此一来，我们的比较损失仍可写成成对排序损失，只是在任务特定损失之上，

[3] 掩码中的非零元素数量（L0 范数）决定了可用参数的数量。

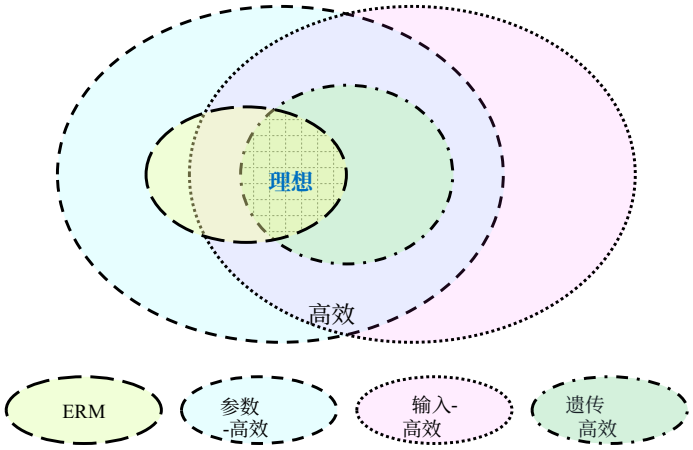


图 2. 本文部分概念的维恩图。经验风险最小化（ERM）指公式（1）的最小化，它是参数高效（满足公式（3））的一个子集。比较原则中的高效模型（紫色相交区域）除了参数高效外，还需输入高效（满足公式（2））。遗传高效模型不仅要求完整模型高效，也要求其任意消融模型都高效，即满足推论 3.1 中的公式（4）。比较损失公式（5）的训练目标既是遗传高效的，也是 ERM，即位于中央重叠网格区域。

任务特定损失，

$$\mathcal{L}_{\text{cmp}}(x, y, \theta) = \sum_{i=0}^{c} \sum_{j=i+1}^{c+1} \max\left(0, l^{(i)} - l^{(j)}\right). \tag{5}$$

图 2 展示了比较损失理想模型的定位（中央网格区域），该模型既是 经验风险最小化（*ERM*）也是遗传高效。 遗传高效 是高效模型的子集，而高效模型 是输入高效与输入高效 和参数高效的交集。从这个角度看，比较损失设定的训练目标比经验风险最小化（*ERM*）更严格。当我们将 c 和 b 设为 0 时，公式（5）可退化为公式（1）。此外，比较损失等价于

$$\sum_{l^{(i)} > b} l^{(i)} + \sum_{i=0}^{c-1} \sum_{j=i+1}^{c} \max\left(0, l^{(i)} - l^{(j)}\right),$$

其中，第一项用于最小化未达到目标 b的模型的经验风险，第二项则约束比较关系，以促使完整模型实现遗传高效。

要使用比较损失进行训练，我们首先需要获得若干可比较的消融模型及其任务特定损失。如图3所示，我们将以完整上下文为输入的原始模型视为完整模型 $f(x(0)\,;\,\theta(0))$。根据推论 3.1，我们以完整模型为基础，逐步执行 c步消融。在第 i（$1 \le i \le c$）次消融步骤中，我们使用 CmpCrop 或 CmpDrop，根据前一步的模型 $f(x(i-1)\,;\,\theta(i-1))$裁剪少量输入或隐藏神经元，使新消融的模型 $f(x(i);\,\theta(i))$能够与其所有祖先模型进行比较。所有这些模型各自完成一次预测后，我们便获得了 $c+1$ 个可比较的任务特定损失。再结合虚拟消融模型的 $l(c+1) = b$，即可使用公式（5）计算最终损失。以随机梯度下降优化为例，算法1更正式地说明了训练过程。

算法 1 中的 CmpDrop 和 CmpCrop 是我们针对每个消融步骤提出的两种替代消融方法：前者用于消融参数（隐藏神经元），后者用于消融输入上下文（输入神经元）。

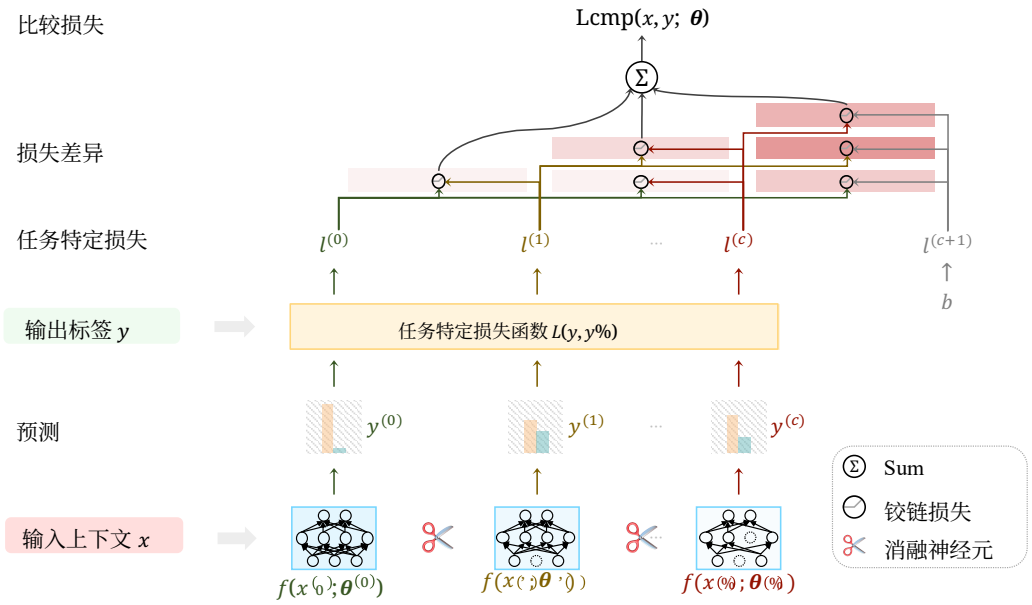


图3. 比较损失概览（彩色显示效果最佳）。给定一个数据样本（x, y），常规训练通常会将输入上下文 x输入神经网络模型，以获得预测结果 $y^{(0)}$，然后仅最小化任务特定损失 $l^{(0)}$。相比之下，比较损失不仅会逐步消融原始模型，以最小化多个任务特定损失 $\{l^{(i)}\}_{i=0}^{c}$，还会通过成对铰链损失约束它们之间的比较关系。

这两种方法都会在先前模型的基础上，以受控方式随机消融神经元，从而无需重新训练每个消融模型，就能覆盖所有可能的消融模型。这是因为随机消融后的模型经过联合训练，并在训练过程中适应了部分神经元缺失的情况。每个消融步骤具体采用哪种方法，可以根据模型和任务数据集而定。理想情况下，只要模型兼容随机失活，就可以使用 CmpDrop；只要任务的输入上下文包含可省略片段，就可以使用 CmpCrop。下面将详细介绍 CmpDrop 和 CmpCrop。

3.2.1 CmpDrop：通过随机失活消融参数。随机失活会以概率 p随机禁用每个神经元，这与我们随机消融隐藏神经元的需求一致。为了获得一个具有更多消融参数的模型 $f(\cdot;\ \boldsymbol{\theta}(i))$，我们不直接对原始模型 $f(\cdot;\ \boldsymbol{\theta}(0))$ 使用更大的丢弃率，而是以概率 p持续消融前一个消融模型 $f(\cdot;\ \boldsymbol{\theta}(i-1))$ 中保留的神经元。具体而言，被丢弃神经元的输出值设为零，而保留神经元的输出值按 $1/(1-p)$ 缩放，以确保所有完整模型/消融模型 [36]中的神经元期望输出值保持一致。这相当于将带缩放因子的掩码[4] $\boldsymbol{m}(i) \in \{0,\ 1,\ 1/(1-p)\}^{|\boldsymbol{\theta}|}$ 应用于前一个参数 $\boldsymbol{\theta}(i-1)$，从而得到消融参数

[4]略有不同参数消融与比较原则中的二值掩码略有不同，我们将缩放因子一并纳入掩码，以便仍能表示简洁地通过 $\boldsymbol{\theta}(i) = \boldsymbol{m}^{(i)} \odot \boldsymbol{\theta}^{(i-1)}$。

算法1 使用比较损失进行训练

输入： 训练数据集 D，消融步骤 c，丢弃率 p，任务特定损失的基线值 b，学习率 η。
输出： 模型参数 θ。

```
随机初始化模型参数 θ
当未收敛时执行
    随机抽取一个数据对（x， y） ~ D
    x(0) ← x , θ(0) ← θ
    l(0) ← L(y, f(x(0) ; θ(0) ))
    对于 i ← 1 至 c，执行
        如果消融隐藏神经元那么                                  ▷ 消融模型参数
            θ(i) ← CmpDrop(θ(i−1)p)
            x(i) ← x(i−1)
        else                                                    ▷ 消融输入上下文
            x(i) ← CmpCrop(x(i−1) )
            θ(i) ← θ(i−1)
        结束 if
        计算任务特定损失： l(i) ← L(y, f(x(i); θ(i) ))
    结束 for
    设置虚拟项的任务特定损失： l(c+1) ← b
    根据公式 ℒcmp(x, y, θ) 计算比较损失，见公式 (5)
    更新参数： θ ← θ − η∇θLcmp(x, y, θ)
end while
```

$\boldsymbol{\theta}(i) = \boldsymbol{m}(i) \odot \boldsymbol{\theta}(i-1)$。$\boldsymbol{m}(i)$ 中的每个元素对应于每个参数的缩放因子，

$$m_k^{(i)} = \begin{cases} 1, & \text{if no dropout in } \theta_k\text{'s layer;} \\ \frac{1}{1-p}, & \text{if neurons at both ends of } \theta_k \text{ survive;} \\ 0, & \text{otherwise.} \end{cases}$$

对于上述公式中的第三种情况，一旦某个神经元被新近消融，与其相连的所有连接参数都设为零；此外，已被 $\boldsymbol{m}(i-1)$ 消融的参数仍设为零。

在实践中，我们可以利用现有的随机失活来实现这一点。不过，为了确保消融模型的可比较性，每次执行 CmpDrop 时都必须使用相同的随机种子和相同的随机数生成器状态。这样，假设当前消融步骤是第 n 次执行 CmpDrop，我们只需以 $1-(1-p)^n$ 的丢弃率运行模型即可。

3.2.2 CmpCrop：通过裁剪消融输入。给定输入上下文 x，CmpCrop旨在裁剪出一个精简上下文$x\prime$，使其不会改变原始真实输出，即 $x\prime \sqsubset x$ 和 $g(x\prime) = g(x) = y$。假设我们在训练时已知 x 的最小支持上下文 $x\star$，即 $\forall x\prime \sqsupseteq x\star\ g(x\prime) = g(x\star)$，且 $\forall x\prime \sqsubset x\star\ g(x\prime) \neq g(x\star)$。然后，CmpCrop可以从非支持上下文 $x \setminus x\star$中随机裁剪掉若干无关紧要的片段，从而生成精简上下文。这样，裁剪后的精简上下文必定包含最小支持上下文，因此真实输出不会改变。

在实践中，要使用 CmpCrop，我们必须确保原始上下文中留有足够多的无关紧要片段，以便进行裁剪$x(0)$。这些片段可以是文档、段落或句子粒度。例如，在问答任务中，任何不影响问题答案的检索段落都可以视为无关紧要的片段。如果提交给 ACM 的稿件

数据集未标注最小支持上下文，我们可以手动向 $x_{(0)}$ 中注入一些额外的噪声片段。

3.3 讨论

进一步推导式（5），我们发现，比较损失可以视为对多个任务特定损失进行动态加权。具体而言，该损失可改写如下：

$$
\begin{aligned}
\mathcal{L}_{\text{cmp}} &= \sum_{i=0}^{c}\sum_{j=i+1}^{c+1}\max\left(0, l^{(i)} - l^{(j)}\right) \\
&= \sum_{i=0}^{c}\sum_{j=i+1}^{c+1}\mathbb{1}_{l^{(i)}>l^{(j)}}\cdot l^{(i)} - \sum_{i=0}^{c}\sum_{j=i+1}^{c+1}\mathbb{1}_{l^{(i)}>l^{(j)}}\cdot l^{(j)} \\
&= \sum_{i=0}^{c+1}\sum_{j=i+1}^{c+1}\mathbb{1}_{l^{(i)}>l^{(j)}}\cdot l^{(i)} - \sum_{j=0}^{c}\sum_{i=j+1}^{c+1}\mathbb{1}_{l^{(j)}>l^{(i)}}\cdot l^{(i)} \\
&= \sum_{i=0}^{c+1}\sum_{j=i+1}^{c+1}\mathbb{1}_{l^{(i)}>l^{(j)}}\cdot l^{(i)} - \sum_{i=1}^{c+1}\sum_{j=0}^{i-1}\mathbb{1}_{l^{(j)}>l^{(i)}}\cdot l^{(i)} \\
&= \sum_{i=0}^{c+1}\sum_{j=i+1}^{c+1}\mathbb{1}_{l^{(i)}>l^{(j)}}\cdot l^{(i)} - \sum_{i=0}^{c+1}\sum_{j=0}^{i-1}\mathbb{1}_{l^{(j)}>l^{(i)}}\cdot l^{(i)} \\
&= \sum_{i=0}^{c+1}\left[\sum_{j=i+1}^{c+1}\mathbb{1}_{l^{(i)}>l^{(j)}}\cdot l^{(i)} - \sum_{j=0}^{i-1}\mathbb{1}_{l^{(j)}>l^{(i)}}\cdot l^{(i)}\right] \\
&= \sum_{i=0}^{c+1} l^{(i)}\cdot\left[\sum_{j=i+1}^{c+1}\mathbb{1}_{l^{(i)}>l^{(j)}} - \sum_{j=0}^{i-1}\mathbb{1}_{l^{(i)}<l^{(j)}}\right] \\
&= \sum_{i=0}^{c+1}\sum_{j\neq i}\text{CMP}(i,j,l^{(i)},l^{(j)})\cdot l^{(i)},
\end{aligned}
\tag{6}
$$

其中，$\mathbb{1}_C$ 是一个指示函数：当条件 C 成立时取值为 1，否则取值为 0；CMP 函数用于判断模型 $f(x_{(i)};\theta_{(i)})$ 相较于 $f(x_{(j)};\theta_{(j)})$ 是否遵循比较原则，并调整 $l_{(i)}$ 的权重。不遵从的情况有两种：当 $f(x_{(i)};\theta_{(i)})$ 的消融程度较低（$i<j$），但所得损失较大时，我们会增加 $l_{(i)}$ 的权重；当 $f(x_{(i)};\theta_{(i)})$ 的消融程度较高（$i>j$），但所得损失较小时，我们会降低 $l_{(i)}$ 的权重。形式上，CMP 函数可写为：

$$
\text{CMP}(i,j,l^{(i)},l^{(j)}) = \begin{cases} 1, & \text{if } i<j \text{ and } l^{(i)} > l^{(j)}; \\ -1, & \text{if } i>j \text{ and } l^{(i)} < l^{(j)}; \\ 0, & \text{otherwise.} \end{cases}
$$

在这里我们可以注意到，对于一对不符合比较原则的模型，我们会提高（+1）消融程度较低的模型的任务特定损失的权重，并相应降低（-1）消融程度较高的模型的损失权重。因此，令 $\alpha_{(i)} = \sum_{j\neq i}\text{CMP}(i,j,l_{(i)}l_{(j)})$ 表示 $l_{(i)}$ 的权重，则所有任务特定损失（包括虚拟项）的权重之和为 0，即 $\sum_{i=0}^{c}\alpha_{(i)} = -\alpha_{(c+1)} = \sum_{i=0}^{c}\mathbb{1}_{l^{(i)}>b}$。由于$l_{(c+1)} = b$是一个常数，式（6）也等价于 $\sum_{i=0}^{c}\alpha_{(i)}l_{(i)}$，即，总权重等于任务特定

劣于虚拟基线的损失 b，并根据其表现自适应地分配给 $c+1$ 损失。通过这种方式，表现不佳的完整模型/消融模型将得到更大力度的优化。我们还在§5.1中对其他启发式加权策略进行了实证比较。

对于参数消融，除了能够对各项任务特定损失赋予不同权重之外，采用 CmpDrop 的比较损失还可以分别计算不同部分参数的梯度。根据公式 (6)，比较损失等于所有违反比较原则的任务特定损失对之间的差值之和，即$\sum_{i<j \wedge l(i)>l(j)} \left(l(i)-l(j)\right)$，因此我们可以根据各任务特定损失对之间的差值分析比较损失的梯度。为便于说明，我们以原始模型 $f(x;\ \boldsymbol{\theta})$ 和一个参数已被消融 n 次的模型 $f(x';\ \boldsymbol{\theta}')$ 为例，其他参数不同的模型对也类似。假设原始参数为 $\boldsymbol{\theta}=(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w})$，消融后的参数为 $\boldsymbol{\theta}'=(\boldsymbol{u}'\boldsymbol{v}'\boldsymbol{w}')=(\boldsymbol{u},\boldsymbol{v}/(1-p)^n,0)$，其中 $\boldsymbol{u}$ 是来自未应用随机失活层的参数，$\boldsymbol{w}$ 是经过 n 次 CmpDrop 消融的参数，$\boldsymbol{v}'$ 是经过 n 次随机失活后保留下来的缩放参数。那么，如果它们的任务特定损失违反比较原则，即 $l>l'$，则该模型对对比较损失的梯度贡献为

$$\nabla_{\boldsymbol{\theta}}(l-l')=(\nabla_{\boldsymbol{u}}(l-l'),\nabla_{\boldsymbol{v}}(l-l'),\nabla_{\boldsymbol{w}}l).$$

我们可以看到，相对于 $\boldsymbol{w}$ 的比较损失高于相对于其他参数的比较损失。这很直观，因为消融掉 $\boldsymbol{w}$ 后，模型的表现反而更好，这表明当前的$\boldsymbol{w}$ 效率低下，因此我们需要专注于更新 $\boldsymbol{w}$。

除了动态加权的视角外，比较损失也可以被视为训练过程中的一种“反向消融研究”。这是因为，与消融研究在验证期间确定被移除组件的贡献不同，比较损失认为被消融的神经元应当有所贡献，并以此为目标优化参数。

对于训练复杂度，在比较次数 c（即消融步骤数）通常较少的情况下，计算最终比较损失的开销小到可以忽略不计，而每次更新步骤增加的计算开销主要来自模型的多次前向传播和反向传播。具体而言，对于相同的批量大小，使用比较损失的训练步骤开销是常规训练的 $1+c$ 倍。然而，就推理复杂度而言，使用比较损失训练的模型在测试时与常规训练的模型相同。

4 实验

为了评估我们的方法在自然语言理解任务中的有效性和泛化能力，我们在 3 项具有代表性输出类型的任务上进行了实验，包括分类（8 个数据集）、抽取（2 个数据集）和排序（4 个数据集）。其中，分类任务要求为一段文本或一对文本预测一个类别；抽取任务要求预测一对边界位置，以抽取起始和结束边界之间的文本片段；排序任务要求预测相关性等级列表，以对候选项进行排序。具体而言，这三项不同的任务分别是文本分类（见 §4.1）、阅读理解（抽取，见 §4.2）和伪相关反馈（排序，见 §4.3）。我们在文本分类和阅读理解任务中评估仅使用 CmpDrop 的比较损失，在阅读理解和伪相关反馈任务中评估仅使用 CmpCrop 的比较损失，并在阅读理解任务中评估同时使用 CmpDrop 和 CmpCrop 的比较损失。对于每项任务，我们首先介绍所用的数据集，然后介绍我们模型及基线模型的实现，最后展示实验结果。

在开始每项实验之前，我们会说明一些常见的实验设置。对于算法 b 中的任务特定损失的基线值1，我们提供两种设置选项。一种是直接设为 $b = 0$，这相当于为所有完整模型/消融模型设定一个无法达到的目标值，从而促使它们的任务特定损失不断降低。然而，这会导致完整模型接触到所有训练数据，并可能加剧过拟合。因此，为减少对完整模型进行优化的次数，我们的第二种方案是将基线值设为完整模型的任务特定损失，即 $b = l(0)$。这样，只有当完整模型的表现不如其消融模型时，才会对其进行优化。实践中，我们倾向于采用设置 $b = 0$；如果发现模型容易在数据集上过拟合，则改用设置 $b = l(0)$。对于每个 CmpDrop 中的丢弃率 p，我们采用与基线模型相同的设置，在所有实验中均为 0.1。对于其他常规训练超参数，例如批量大小和学习率，除非特别说明，我们也沿用经过仔细调优的基线模型的设置。我们使用 PyTorch 和 HuggingFace Transformers [71] 实现模型和基线模型。所有模型均在 Tesla V100 GPU 上训练。在文本分类和阅读理解任务中，我们使用 5 个随机种子训练每个模型。在伪相关反馈任务中，我们使用固定的随机种子 42 训练所有模型。表中以均值±标准差形式呈现的结果，是根据五个随机种子的评估结果汇总得出的；否则，结果表示使用随机种子 42 训练的模型的性能。为方便起见，在表中我们用“Cmp”表示比较损失，并在括号中用“Drop”和“Crop”分别指代 CmpDrop 和 CmpCrop。

4.1 分类：文本分类的应用

文本分类是自然语言理解中的一项基础任务，旨在为一段或一组文本指定一个预定义类别。在许多文本分类数据集中，输入上下文的所有片段似乎都对文本类别发挥着重要作用，而且几乎没有关于最小支持上下文的标注，因此，我们很难在不改变分类标签的情况下，直接裁剪原始输入来构造输入消融模型。也就是说，这很可能违反比较原则中消融输入的标签保持不变这一约束，因此我们无法将 CmpCrop 应用于此任务。然而，许多当前的神经分类模型在训练过程中会使用随机失活，因此在此任务中，我们仅验证只使用 CmpDrop 的比较损失。

4.1.1 数据集。通用语言理解评估（GLUE）基准 [66] 由多种自然语言理解任务组成。遵循 [21], ，我们排除存在问题的 WNLI 数据集，并在 8 个数据集上进行实验：(1) 多体裁自然语言推理（**MNLI**） [70] 是一项句子对分类任务，旨在预测第二个句子对于第一个句子是蕴涵、矛盾还是中立。(2) 微软研究院释义语料库（**MRPC**） [22] 旨在预测一对句子在语义上是否等价。(3) 问句自然语言推理（**QNLI**） [66] 是一项二元句子对分类任务，旨在预测某个句子是否包含问题的正确答案。(4) Quora 问题对（**QQP**） [13]是一项二元句子对分类任务，旨在预测 Quora 上提出的两个问题在语义上是否等价。(5) 文本蕴涵识别（**RTE**） [4] 是一项与 MNLI 类似的二元蕴涵任务，但训练样本少得多。(6) 斯坦福情感树库（**SST-2**） [61] 是一项二元句子情感分类任务，包含从电影评论中提取的句子。(7) 语义文本相似度基准（**STS-B**） [9] 是一项句子对分类任务，旨在确定两个句子的语义相似程度。(8) 语言可接受性语料库（**CoLA**） [69] 是一项二元句子分类任务，旨在判断单个英语句子是否符合语言学规范。

表1。GLUE 语言理解基准开发集上的分类性能。

模型	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	CoLA	平均
BERT基础型 [21]	$84.2_{\pm 0.3}$	$85.9_{\pm 0.5}$	$91.0_{\pm 0.1}$	$91.0_{\pm 0.1}$	$68.2_{\pm 1.7}$	$92.2_{\pm 0.2}$	$88.9_{\pm 1.0}$	$61.9_{\pm 1.1}$	$82.92_{\pm 0.14}$
+R-Drop [40]	$84.3_{\pm 0.2}$	$86.5_{\pm 0.5}$	$91.6_{\pm 0.1}$	$91.4_{\pm 0.1}$	$68.8_{\pm 1.2}$	$92.2_{\pm 0.3}$	$89.4_{\pm 0.8}$	$62.9_{\pm 0.7}$	$83.38_{\pm 0.14}$
+Cmp（c 丢弃）	**84.8**$_{\pm 0.2}$	**87.1**$_{\pm 1.0}$	**91.9**$_{\pm 0.2}$	**91.5**$_{\pm 0.1}$	**70.5**$_{\pm 1.2}$	**93.1**$_{\pm 0.4}$	**89.7**$_{\pm 0.6}$	**63.4**$_{\pm 1.1}$	**83.96**$_{\pm 0.31}$
RoBERTa基础型 [44]	$87.4_{\pm 0.3}$	$89.9_{\pm 0.6}$	$92.8_{\pm 0.2}$	$91.4_{\pm 0.1}$	$76.5_{\pm 0.9}$	$95.1_{\pm 0.2}$	$90.8_{\pm 0.1}$	$62.7_{\pm 0.6}$	$85.82_{\pm 0.13}$
+R-Drop [40]	$87.6_{\pm 0.1}$	$90.0_{\pm 0.6}$	$92.9_{\pm 0.1}$	$91.5_{\pm 0.0}$	$78.5_{\pm 1.5}$	$95.4_{\pm 0.2}$	$91.1_{\pm 0.1}$	$64.0_{\pm 0.5}$	$86.37_{\pm 0.26}$
+Cmp（c 下降）	**88.0**$_{\pm 0.1}$	**90.5**$_{\pm 0.5}$	**93.3**$_{\pm 0.1}$	**91.9**$_{\pm 0.1}$	**79.3**$_{\pm 1.2}$	**95.5**$_{\pm 0.2}$	**91.3**$_{\pm 0.1}$	**65.4**$_{\pm 0.8}$	**86.91**$_{\pm 0.20}$
ALBERT基础型 [37]	$85.0_{\pm 0.3}$	$88.4_{\pm 0.6}$	$92.1_{\pm 0.3}$	$90.4_{\pm 0.1}$	$77.8_{\pm 0.8}$	$93.0_{\pm 0.4}$	$90.9_{\pm 0.2}$	$59.8_{\pm 1.7}$	$84.68_{\pm 0.18}$
+R-Drop [40]	$85.4_{\pm 0.4}$	$88.7_{\pm 0.8}$	$92.1_{\pm 0.2}$	$90.5_{\pm 0.1}$	$78.0_{\pm 1.4}$	$93.3_{\pm 0.3}$	$90.9_{\pm 0.2}$	$59.7_{\pm 0.4}$	$84.83_{\pm 0.14}$
+Cmp（c 丢弃）	**85.7**$_{\pm 0.1}$	**89.5**$_{\pm 0.4}$	**92.3**$_{\pm 0.1}$	**91.0**$_{\pm 0.1}$	**78.3**$_{\pm 1.4}$	**93.5**$_{\pm 0.2}$	**91.0**$_{\pm 0.2}$	**60.3**$_{\pm 0.9}$	**85.19**$_{\pm 0.24}$

4.1.2 模型与训练。遵循 R-Drop [40], 这一利用随机失活的最先进训练方法，我们在基于预训练语言模型（PLMs）[5]的热门分类模型上验证比较损失。具体而言，我们采用 BERT基础型 [21]、RoBERTa基础型 [44] 和 ALBERT基础型 [37] 作为骨干网络进行微调。STS-B 使用的任务特定损失为均方误差（MSE），其他数据集则使用交叉熵。我们针对每个数据集采用不同的训练超参数。对于基线模型和使用比较损失训练的模型，我们分别从 {1e-5, 2e-5, 3e-5, 4e-5} 中选择学习率，从 {0, 0.1} 中选择预热率，从 {16, 24, 32} 中选择批量大小，并将训练轮数设为 2 到 5。对于我们的模型，我们将消融步骤数 c（即 CmpDrop 的次数）调节为 1 到 4。遵循 R-Drop [40], 中的超参数设置，我们也为所有骨干模型实现 R-Drop，作为一种对比方法；它与 CmpDrop 一样，会多次执行随机失活。

4.1.3 结果。 我们在表1中展示了分类性能，其中 STS-B 的评估指标为皮尔逊相关系数，CoLA 的评估指标为马修斯相关系数，其他任务的评估指标为准确率。对于基于 BERT基础型的模型，可以看到，我们的模型（+Cmp）全面优于经过充分调优的基线 BERT基础型，平均提升了 1.04 个百分点，这证明了比较损失在分类任务中的有效性。此外，使用比较损失训练的模型也优于使用最先进的 R-Drop 训练的模型，平均提升了 0.58 个百分点，这体现了比较损失的优越性。对于基于其他更先进模型 RoBERTa基础型 和 ALBERT基础型的模型，我们也发现了稳定的提升。此外，由于 ALBERT 在多个层之间复用参数，其参数利用可提升的空间最小，这与我们的观察一致：比较损失为 ALBERT 带来的提升最小。

4.2 抽取：应用于阅读理解

抽取式阅读理解（RC） [43, 59] 是问答（QA） [11, 31,41, 60, 79]的一个重要技术分支。给定一个问题和一段上下文，抽取式阅读理解旨在从上下文中提取一个文本片段作为预测答案。目前主流的阅读理解模型基本采用预训练 Transformer [64] 架构，并在微调期间于许多层中使用随机失活。这使我们能够使用 CmpDrop 提升模型参数的效用。此外，给定的上下文通常很长，且包含许多干扰噪声片段，因此我们也可以通过随机删除标注的干扰段落，使用 CmpCrop 提高模型对上下文的利用率。因此，我们计划验证在此任务中使用 CmpDrop 和/或 CmpCrop 的比较损失的有效性。

[5]https://github.com/huggingface/transformers/blob/v4.19.2/examples/pytorch/text-classification/README.md提交给 ACM 的稿件

4.2.1 数据集。我们仅在 SuQAD 上使用 CmpDrop 评估比较损失 [59], 该数据集包含 10 万个单跳问题，其中 9832 个用于验证；我们还在 HotpotQA 上进行评估 [74], 该数据集包含 11.3 万个多跳问题，其中 7405 个用于验证。对于 HotpotQA，我们采用干扰项设置：每个问题的上下文包含 10 个段落，但只有其中 2 个有助于回答问题，其余 8 个是检索到的干扰段落。这些段落与问题相关，但不能支持答案。借此，我们可以评估在 HotpotQA 干扰项数据上使用 CmpCrop 的比较损失。

4.2.2 模型与训练。我们采用基于预训练语言模型的简单而有效的阅读理解模型 [3, 15, 21, 37, 44], ，其输入为问题与上下文的拼接，并使用线性层预测答案的起始和结束位置。按照 [21] ，我们使用答案边界的交叉熵作为任务特定损失函数，并在前 10% 的训练步数中进行学习率预热。对于 SQuAD，我们采用广受欢迎的 BERT [21], RoBERTa [44], ELECTRA [15]和 ALBERT [37] 作为骨干网络，最大序列长度设为 512；这些模型均曾在多个问答基准测试中先后取得顶尖排名 [58, 59, 74]。我们首先为基线模型调优学习率，取值范围为 {1e-5, 3e-5, 5e-5, 8e-5, 1e-4, 2e-4}；批量大小取{8, 12, 32}；训练轮数取{1, 2, 3}。然后，设定 $c = 2$后，我们采用这些超参数，并使用带有两个 CmpDrop 的比较损失训练模型。对于 HotpotQA，我们采用最先进的 Longformer [3] 作为骨干网络，最大序列长度设为 2048，其输入格式为 <s> [YES] [NO] [Q] 问题 </s> [T] 标题$_1$ [P] 段落$_1$ $\cdots$ [T] 标题$_{10}$ [P] 段落$_{10}$</s>。特殊标记 [YES]/[NO], [Q], [T], 和 [P] 分别表示是/否答案，以及问题、标题和段落的起始位置。类似地，我们为基线模型选择学习率{1e-5, 3e-5}、批量大小{6, 9, 12}和训练轮数{3, 5, 8}。随后，我们分别使用三种比较损失训练模型：前两种分别应用一次 CmpDrop/CmpCrop（$c = 1$），第三种则先应用一次 CmpCrop，再应用一次 CmpDrop（$c = 2$）。此外，在沿用通用超参数并搜索系数权重 α（取值为{0.1, 0.5, 1, 1.5}）的基础上，我们还实现了 R-Drop [40] ，将其作为 CmpDrop 的竞争方法。

4.2.3 结果。由于此处我们关注的是抽取，因此仅使用精确匹配（EM）和 F1 来衡量抽取出的答案，这与 HotpotQA 的官方设置略有不同，后者会同时评估支持事实的识别情况。从表 2 可以看出，我们直接使用任务特定损失公式（1）训练实现的基线模型，其结果大多优于原论文报告的结果。改用比较损失公式（5）训练后，即使不重新搜索训练超参数，我们的模型仍能显著优于这些经过精细调优的基线模型，证明了比较损失在抽取任务上的有效性。此外，基于三种不同预训练语言模型取得的稳定提升，也证明了比较损失具有模型无关性。进一步来看 HotpotQA 上的结果，尽管 CmpDrop 和 CmpCrop 均带来了显著提升，但 CmpCrop + CmpDrop 取得了最佳结果，这表明 CmpDrop 和 CmpCrop 可能会为训练后的模型带来不同的益处。

4.3 排序：应用于伪相关反馈

伪相关反馈（PRF） [1] 是一种有效的查询理解 [10] 技术，可提高排序准确率，旨在缓解查询与其潜在相关文档之间语言表达不匹配的问题。给定原始查询 q和文档集合 C，基础型排序模型会返回一个排序列表 $D = (d_1, d_2, \cdots d_{|D|})$。设 $D_{\leq k}$ 表示包含排名靠前的 k 篇文档的反馈集，其中 k 通常称为PRF深度。PRF的目标是利用 $D_{\leq k}$中的查询相关信息，将原始查询 q 改写为新的表示 $q^{(k)}$，即 $q^{(k)} = f((q, D_{\leq k}); \theta)$，其中 $q^{(k)}$ 预期能够带来更好的排序结果。尽管PRF方法确实

表2. SQuAD 和 HotpotQA 干扰项开发集上的问答性能。带 † 的结果是向论文作者询得的。

模型	EM	F1
SQuAD		
BERT基础型 [21]	80.8	88.5
BERT基础型 （我们的实现）	$81.3_{\pm 0.2}$	$88.5_{\pm 0.1}$
+ R-Drop [40]	$82.2_{\pm 0.1}$	$89.1_{\pm 0.1}$
+ Cmp（双重 Drop）	$\mathbf{82.3}_{\pm 0.2}$	$\mathbf{89.3}_{\pm 0.1}$
RoBERTa基础型 [44](我们的实现)	$85.8_{\pm 0.1}$	$92.2_{\pm 0.1}$
+ R-Drop [44]	$86.4_{\pm 0.1}$	$92.3_{\pm 0.1}$
+ Cmp（双重 Drop）	$\mathbf{86.5}_{\pm 0.2}$	$\mathbf{92.6}_{\pm 0.1}$
ELECTRA基础型 [15]	84.5	90.8
ELECTRA基础型 （我们的实现）	$85.9_{\pm 0.3}$	$92.3_{\pm 0.2}$
+ R-Drop [40]	$86.5_{\pm 0.1}$	$92.3_{\pm 0.1}$
+ Cmp（双重 Drop）	$\mathbf{86.6}_{\pm 0.1}$	$\mathbf{92.7}_{\pm 0.1}$
ALBERT基础型 [37]	82.3	89.3
ALBERT基础型 （我们的实现）	$83.6_{\pm 0.2}$	$90.6_{\pm 0.1}$
+ R-Drop [40]	$83.7_{\pm 0.2}$	$90.7_{\pm 0.2}$
+ Cmp（双重 Drop）	$\mathbf{84.4}_{\pm 0.1}$	$\mathbf{91.0}_{\pm 0.1}$
HotpotQA		
Longformer† 基础型 [3]	60.3	74.3
Longformer基础型（我们的实现）	$61.9_{\pm 0.4}$	$75.6_{\pm 0.3}$
+ R-Drop [40]	$62.0_{\pm 0.2}$	$76.0_{\pm 0.2}$
+ Cmp（1 次 Drop）	$63.0_{\pm 0.4}$	$77.0_{\pm 0.4}$
+ Cmp（1 次裁剪）	$62.6_{\pm 0.2}$	$76.4_{\pm 0.2}$
+ Cmp（1 次裁剪，1 次丢弃）	$\mathbf{63.5}_{\pm 0.3}$	$\mathbf{77.2}_{\pm 0.3}$

通常能平均提升排序表现 [16], , 但单个改写后的查询不可避免地会受到查询漂移 [50, 81] 的影响，这是因为反馈集中客观存在噪声，导致改写后的查询效果不如原始查询。因此，我们可以使用CmpCrop的比较损失来训练PRF模型，通过比较利用不同PRF深度的反馈集改写出的查询所产生的效果，抑制额外增加的噪声。

4.3.1 数据集。我们在 MS MARCO 段落 [51] 集合上进行实验，该集合包含 880 万个英文段落，采集自 Bing 的 100 万个真实查询的搜索结果。MS MARCO 训练集包含 53 万个查询（平均每个查询约有 1.1 个相关段落），开发集包含 6980 个查询，在线评测集包含 6837 个查询。除此之外，我们还考虑基于 MS MARCO 段落集合的 TREC DL 2019 [20], TREC DL 2020 [19], 和 DL-HARD [47],三个离线评测基准，它们分别包含 43、54 和 50 个细粒度标注查询（相关性等级为 0 到 3）。其中，DL-HARD [47] 是一个近期推出、专注于复杂查询的评测基准。我们使用 MS MARCO 训练集训练模型，并在 MS MARCO 开发集上评估训练好的模型，以调整超参数并选择模型检查点。最后，在在线 MS MARCO 评测集[6] 及另外三个离线基准上评估选定的模型。

4.3.2 模型与训练。 我们分别在两种基础检索模型上开展伪相关反馈实验：ANCE [72] （稠密检索）和 uniCOIL [42] （稀疏检索）。对于它们的 PRF模型，我们不会显式修改查询文本，而是

[6] https://microsoft.github.io/MSMARCO-Passage-Ranking-Submissions/leaderboard/提交

表3. 在基于 MS MARCO 段落集合构建的基准测试上的检索性能。ANCE 和 uniCOIL 是基础检索模型，+ PRF 表示 PRF 基线模型，+ Cmp 表示我们使用 1 CmpCrop 的比较损失训练的 PRF 模型，上标 (k) 表示测试时使用的 PRF 深度。上标 * 表示相较于 PRF 基线模型取得了具有统计显著性的提升，且 $p \leq 0.1$。

模型	MARCO 开发集			MARCO 评测集	TREC DL 2019		TREC DL 2020		DL-HARD	
	NDCG@10	MRR@10	R@1K	MRR@10	NDCG@10	R@1K	NDCG@10	R@1K	NDCG@10	R@1K
ANCE [72]	38.76	33.01	95.84	31.70	64.76	75.70	64.58	77.64	33.39	76.65
+伪相关反馈(3) [75]	40.10	34.40	95.90	33.00	68.10	79.10	69.50	81.50	36.50	76.10
+Cmp(3) (1 裁剪)	40.68*	34.84*	96.94*	-	68.42	80.10*	69.58	81.77	35.61	79.39*
+Cmp(5) (1 裁剪)	**41.01***	**35.14***	**97.03***	**34.17**	**69.58***	**80.81***	**70.44***	**82.77***	**37.44***	**79.55***
uniCOIL [42]	41.21	35.13	95.81	34.42	70.09	82.83	67.35	84.42	35.96	76.85
+ 伪相关反馈(3)	41.76	35.48	96.85	-	69.42	83.32	69.25	84.44	36.53	77.48
+Cmp(3) （1 裁剪）	**42.02***	**35.75***	**96.91**	**35.14**	**70.10***	**83.58**	**69.70***	**84.51**	**36.90***	**77.67**

直接按照当前最先进的方法 ANCE-PRF 生成新的查询向量用于检索 [75]。这样，我们就可以使用正样本文档 [34] 的负对数似然作为任务特定损失，端到端地直接优化改写后查询的检索效果：

$$L(\boldsymbol{q}^{(k)}) = -\log \frac{e^{\text{sim}(\boldsymbol{q}^{(k)}, \boldsymbol{d}^+)}}{e^{\text{sim}(\boldsymbol{q}^{(k)}, \boldsymbol{d}^+)} + \sum_{d^- \in D^-} e^{\text{sim}(\boldsymbol{q}^{(k)}, \boldsymbol{d}^-)}},$$

其中，$\boldsymbol{d}^+$ 是一个采样得到的、与 q 相关的文档向量，$\boldsymbol{q}^{(k)}$，$\text{sim}(\cdot,\cdot)$表示两个向量的点积，而 D^-是它们对应的负样本文档集合。由于只更新查询向量[7]，我们挖掘了一个轻量级集合（稠密检索包含 5.3M 个文档，稀疏检索包含 3.7M 个文档），其中包含所有训练查询的正样本文档和困难负样本文档。这样，对于每个查询，轻量级集合中除其正样本文档之外的所有文档都可用作其D^-。通常，我们的 PRF 模型由一个编码器、一个向量投影器和一个池化器组成。首先，原始查询 q 和 $D_{\leq k}$ 中的反馈文档按顺序拼接，并以 [SEP] 作为分隔符，然后输入编码器，以获取每个词元的上下文嵌入。接着，投影器将上下文嵌入映射为与文档向量维度相同的向量。最后，将所有词元向量池化为单个查询向量。对于稠密检索，编码器初始化自 ANCE$_{\text{FirstP}}$[8]，，投影器为线性层，池化器则对序列中的第一个向量（[CLS]）进行层归一化，与先前的工作 [75]相同。对于稀疏检索，编码器和投影器使用带掩码语言模型头的 BERT$_{\text{基础型}}$ 初始化，其中投影器是一个带 GeLU [28]激活函数和层归一化的多层感知机，池化器则由最大池化操作和 L2归一化[9]构成。我们对 PRF 基线模型进行微调，最多训练 12 轮，批量大小为 96，学习率从 {2e-5, 1e-5, 5e-6} 中选择，并为每个查询随机采样 0 到 5 的 PRF深度 k。随后，我们使用 $c = 1$ CmpCrop的比较损失对 PRF模型进行微调，最多训练 6 轮，批量大小为 48。这样，我们模型的最大训练步数与基线模型相同，即每个原始查询最多进行 12 次优化。由于使用多个随机种子的训练成本较高，我们使用配对t检验来计算检索性能的显著差异。

4.3.3 结果。 表 3报告了各模型在多个基准测试上的官方指标（MARCO 使用 MRR@10，其他数据集使用 NDCG@10）以及 Recall@1K。除了报告表现最佳的 PRF 深度下的结果

[7]文档的固定向量从以下工具预先构建的索引中恢复：https://github.com/castorini/pyserini。 [8] https://github.com/microsoft/ANCE [9]我们发现，L2归一化有助于模型稳定训练，而且不会改变文档对查询的相关性排序。

表4. 不同加权策略在 HotpotQA 干扰项开发集上的问答性能。Cmp 指 Longformer + CmpCrop + CmpDrop，它们通过比较损失对多个任务特定损失进行自适应加权。其他方法均为启发式方法：平均为所有任务特定损失赋予相同权重，FIRST 和 LAST 仅对第一个或最后一个赋予权重，而 MAX 则动态地仅对最大的一个赋予权重。

加权方法	EM	F1
Cmp	**63.5**$_{\pm 0.3}$	**77.2**$_{\pm 0.3}$
平均	63.2$_{\pm 0.3}$	76.7$_{\pm 0.3}$
FIRST	62.1$_{\pm 0.1}$	75.8$_{\pm 0.1}$
LAST	61.9$_{\pm 0.4}$	75.6$_{\pm 0.3}$
MAX	63.1$_{\pm 0.3}$	76.7$_{\pm 0.3}$

（上标括号中的数字），为与 ANCE-PRF(3)（第二行）进行公平比较，我们还展示了 ANCE-PRF + Cmp(3)的结果；这两者都使用前 3 篇文档作为反馈。可以看到，PRF 基线模型（+PRF）通常确实优于其基础检索模型，但 uniCOIL-PRF 在 TREC DL 2019 的 NDCG@10 上下降了 0.67 个百分点，这反映了查询漂移的存在。然而，采用比较损失训练的我们的 PRF 模型（+ Cmp）在所有方面都优于其基础检索模型。在同样使用 3 篇反馈文档的情况下，我们的 ANCE-PRF + Cmp 在除 DL-HARD 上的 NDCG@10 以外的所有指标上，也都优于已发表的最先进的 ANCE-PRF [75]。此外，使用 5 篇反馈文档时，ANCE-PRF + Cmp 在 DL-HARD 的 NDCG@10 上超过了 ANCE-PRF。对于稀疏检索，采用比较损失训练的我们的 PRF 模型（+Cmp）也超过了按照 ANCE-PRF 的方法实现的强基线 uniCOIL-PRF。以上结果均证明了比较损失在排序任务上的有效性。

5 分析

在本节中，我们进一步开展了几项实验，以进行更全面的分析。首先，根据§3.3中关于动态加权的观点，我们考察比较损失的自适应加权是否比其他加权策略更有效（§5.1）。接下来，我们尝试几种其他比较策略，以总结实践中选择消融数量和消融方法的一些指导经验（§5.2）。然后，为了验证比较损失对隐藏神经元和输入神经元效用的提升，我们考察了参数数量不同的模型的性能（§5.3）以及上下文长度不同的模型的性能（§5.4）。此外，我们将损失曲线可视化，以探究采用不同消融方法的比较损失对任务特定损失的影响（§5.5）。最后，我们详细展示比较损失实际产生的训练开销（§5.6）。

5.1 加权策略的效果

为了从动态加权的角度验证比较损失的作用，我们保持表2最后一行中 Longformer + CmpCrop + CmpDrop 的所有训练设置不变，仅用一些启发式方法替换任务特定损失的加权策略。表4展示了它们在 HotpotQA 开发集上的表现。AVERAGE、FIRST 和 LAST 是三种静态加权策略。AVERAGE 为所有任务特定损失分配相同权重，而 FIRST 和 LAST 分别只为第一个和最后一个任务特定损失分配权重，即 FIRST 优化完整模型在不使用随机失活时的 $l(0)$，而 LAST 优化模型在常规丢弃率 p 下的 $l(2)$（相当于表2中的基线 Longformer）。MAX 是另一种动态加权策略，它仅为最大的任务特定损失分配权重。可以看出，比较损失中的动态加权显著优于这些启发式加权策略，这证明比较损失能够更恰当地分配权重。提交给 ACM 的稿件

表5. 不同比较策略在 HotpotQA 干扰项开发集上的问答性能。 c 表示消融步骤的数量。x 2 表示某种消融方法重复两次，而 $A + B$ 表示先使用 A，再使用 B。

c	消融顺序	EM	F1
1	CmpDrop	**63.1**	**77.0**
	CmpCrop	63.1	76.8
2	CmpDrop x 2	63.4	77.1
	CmpCrop × 2	63.0	76.7
	CmpDrop + CmpCrop	63.2	76.8
	CmpCrop + CmpDrop	**63.6**	**77.4**

这证明比较损失能够更恰当地分配权重。此外，AVERAGE 优于后三种仅考虑单个任务特定损失的策略，表明考虑多个任务特定损失是有益的。更进一步，尽管后三种策略都只针对一个任务特定损失进行加权，但 MAX 优于另外两种，这表明动态分配优于静态分配。

值得注意的是，直接优化完整模型的 FIRST 优于采用随机失活进行训练的 LAST，这表明训练阶段与推理阶段的随机失活不一致 [82] 确实可能导致完整模型欠拟合。而 Cmp 远优于 FIRST 和 LAST，这表明比较损失能够自动在确保训练与推理一致性和防止过拟合之间取得平衡。

5.2 比较策略的影响

为研究比较策略的影响，即应使用多少个消融步骤进行比较，以及每一步应选择哪种消融方法，我们在 HopotQA 上尝试了多种比较策略，采用不同的比较次数和消融顺序。如表 5 所示，当我们重复进行两次 CmpDrop/CmpCrop 时，结果并未显著进一步提升；但先应用 CmpCrop，再应用 CmpDrop 时，结果则进一步提升。这表明，比较由相同方法消融得到的多个模型，即鼓励模型成为遗传输入高效 或遗传参数高效，似乎对完整模型的性能影响不大；但连续使用两种不同的消融方法，即鼓励模型变得高效(同时具备输入高效和参数高效)，则有助于提升性能。然而，先应用 CmpDrop 再应用 CmpCrop，效果不如仅应用 C mpDrop，这表明消融方法的顺序很重要，而且消融或许应按照模型中的信息流顺序进行。

为进一步确认消融步骤数量的影响 c，我们在图 4 中展示了模型在八个 GLUE 数据集上的平均指标与消融次数之间的关系。我们发现，使用不同数量的 CmpDrop 训练的模型，其平均性能差异很小。使用一个 CmpDrop 训练的模型表现明显最佳，主要是因为它在其中两个数据集上的巨大优势拉高了平均值。因此，如果没有极端的性能需求，通常无需调整超参数 c。

5.3 模型参数的影响

为探究模型参数的影响，我们研究了在不同规模的 BERT、RoBERTa 和 ELECTRA 模型上应用结合 CmpDrop 的比较损失。从表 6 可以看出，结合 CmpDrop 的比较损失在这些骨干模型上均优于基线模型，这表明比较损失能够通过提高参数效用来改善模型性能，而无需增加参数数量

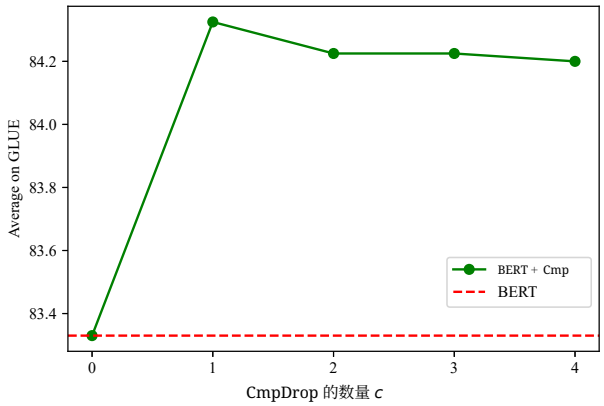


图 4. 随着消融步骤数量的变化，八个GLUE数据集上的平均结果。

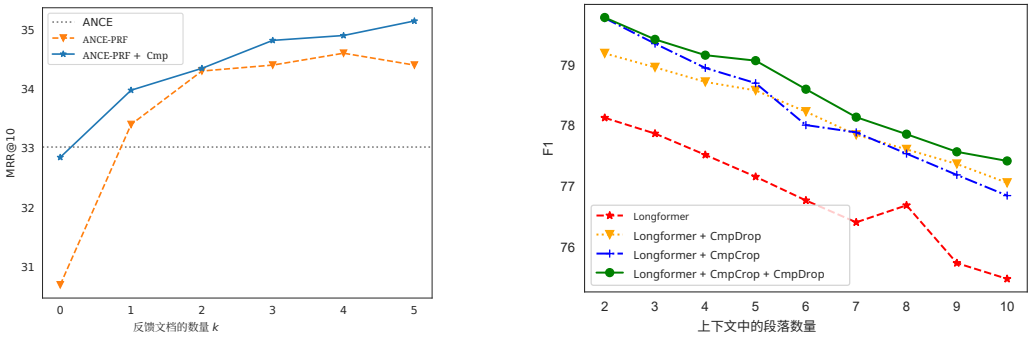
表6. SQuAD开发集（精确匹配/F1）上不同模型规模和初始化方式的基线模型评估结果，以及使用比较损失和CmpDrop训练的模型相对于基线模型的增益。

# 参数	BERT		ELECTRA		RoBERTa	
	基线	增益（%）	基线	增益（%）	基线	增益（%）
微型：4M	$41.5_{\pm 0.5}/54.2_{\pm 0.3}$	$2.2_{\pm 1.4}/1.7_{\pm 1.0}$	-	-	-	-
小型：14M	$74.2_{\pm 0.6}/82.7_{\pm 0.5}$	$2.2_{\pm 0.8}/1.6_{\pm 0.6}$	$78.1_{\pm 0.2}/85.9_{\pm 0.1}$	$1.6_{\pm 0.5}/1.2_{\pm 0.3}$	-	-
中型：42M	$78.0_{\pm 0.4}/85.8_{\pm 0.2}$	$1.6_{\pm 0.7}/1.3_{\pm 0.4}$	-	-	-	-
基础型：110M	$81.3_{\pm 0.2}/88.5_{\pm 0.1}$	$1.2_{\pm 0.2}/0.9_{\pm 0.2}$	$85.9_{\pm 0.3}/92.3_{\pm 0.2}$	$0.8_{\pm 0.4}/0.5_{\pm 0.3}$	$85.8_{\pm 0.1}/92.2_{\pm 0.1}$	$0.8_{\pm 0.2}/0.5_{\pm 0.1}$
大型：335M	$83.9_{\pm 0.2}/90.8_{\pm 0.1}$	$1.2_{\pm 0.2}/0.7_{\pm 0.1}$	$89.0_{\pm 0.1}/94.7_{\pm 0.0}$	$0.7_{\pm 0.2}/0.3_{\pm 0.0}$	$89.0_{\pm 0.3}/94.7_{\pm 0.0}$	$0.7_{\pm 0.3}/0.3_{\pm 0.1}$

。此外，除 BERT中型这一例外外，我们大致可以发现，模型参数越少，比较损失带来的相对增益越大。这是合理的，因为在容量较低的模型中，每个隐藏神经元发挥的作用更大，因此，提高隐藏神经元效用所带来的改进更能体现在最终性能上。相比之下，对于容量较高的模型，它更容易拟合较少的训练数据，也就是说，其任务特定损失已经较低，因此比较损失进一步降低任务特定损失的空间较小。此外，我们观察到，相比采用更复杂预训练的 RoBERTa 和 ELECTRA，结合 CmpDrop 的比较损失通常能为 BERT 带来更大的提升，这表明比较损失有助于模型摆脱由参数初始化导致的局部最优解。

5.4 输入上下文的影响

为了评估输入上下文（即输入神经元）对模型的效用，我们在图中绘制了各模型的性能趋势。5 展示了模型在不同上下文大小下的性能趋势。首先，在两个数据集中，使用比较损失训练的模型在所有上下文大小下都始终优于基线模型，这表明在输入上下文量相同的情况下，我们的模型能够更高效地利用输入神经元。其次，这表明通过上下文选择精简输入后，比较损失还能进一步提升模型性能。此外，我们注意到，图 5(a) 中的 ANCE-PRF +CmpCrop 按预期随着反馈文档数量的增加而提升检索性能，而提交给 ACM 的稿件中，



(a) (b)

图5. 使用不同上下文大小的性能曲线。(a) MARCO 开发集上的PRF模型，水平虚线表示基础检索模型。(b) HotpotQA开发集上的阅读理解模型。

ANCE-PRF 在 4 个反馈文档时达到峰值性能，随后性能下降，这说明我们的模型更稳健，能够从新增的反馈文档中挖掘并利用相关信息。与 PRF 相比，在 HotpotQA 中，图 5(b)显示，随着段落数量增加，所有阅读理解模型的性能都会下降。这可以理解，因为 HotpotQA 中只有 2 个段落包含支持事实，其余 8 个段落大多只是干扰项，因此理想的性能曲线应当是一条水平线，即不会随着段落数量增加而下降。有趣的是，我们发现，从预言机设置（2 个黄金段落）到干扰项设置（10 个段落），Longformer +CmpDrop（2.7%）和 Longformer + CmpCrop + CmpDrop（3.0%）的性能下降幅度低于基线 Longformer（3.4%）。这表明比较损失有助于模型抑制新增上下文中的噪声信息。尽管 Longformer + CmpCrop（3.7%）的性能下降幅度大于 Longformer，但我们认为这是因为 Longformer + CmpCrop 需要针对不同段落数量进行优化，而不使用 CmpCrop 的其他模型只需针对一种输入形式（即始终输入十个段落）进行学习。然而，这种输入形式的多样性使 Longformer + CmpCrop 在段落数量较少时的表现优于 Longformer +CmpDrop（≤ 5）。

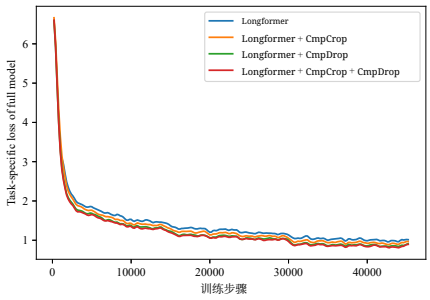
为进一步定量展示比较损失对PRF模型针对上下文大小的鲁棒性的帮助，我们在表7 中报告了 [18] ANCE-PRF + CmpCrop和ANCE-PRF在不同反馈文档数量下的鲁棒性指数。鲁棒性指数定义为 $\frac{N_+ - N_-}{|Q|}$，其中 $|Q|$是评估查询的总数，而N_+ 和 N_- 分别是使用更多~~一篇~~反馈文档时PRF模型性能提升或下降的查询数量。鲁棒性指数的取值范围为 [-1, 1], ，鲁棒性指数越高，模型就越稳健。可以看到，使用CmpCrop和比较损失训练的PRF模型比基线模型稳健得多。此外，从它们鲁棒性指数的差距来看（使用1篇或2篇文档时仅为0.03或0.02，而使用更多文档时为0.05），可以发现比较损失对长文本输入更有帮助。

5.5 损失可视化

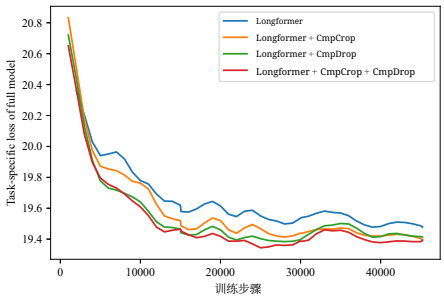
为了弄清比较损失对任务特定损失的影响，我们绘制了完整模型（即 $l_{(0)}$）的任务特定损失曲线，如图 6所示。从图 6(a) 和图 6(b) 可以看出，在批量大小相同的情况下，比较损失能够

表 7. 在 MARCO 开发集上， $q^{(k)}$ 相对于 $q^{(k-1)}$ 在各个 PRF 深度 k下的鲁棒性指数；其中， $q^{(k)}$ 和 $q^{(k-1)}$ 是由 PRF模型重新表述的查询向量，后者的输入上下文比前者少一个文档。

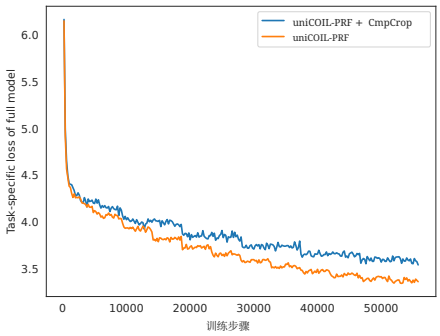
k	1	2	3	4	5
ANCE-PRF	0.51	0.54	0.58	0.58	0.61
ANCE-PRF +Cmp（1 次裁剪）	**0.54**	**0.56**	**0.63**	**0.63**	**0.66**



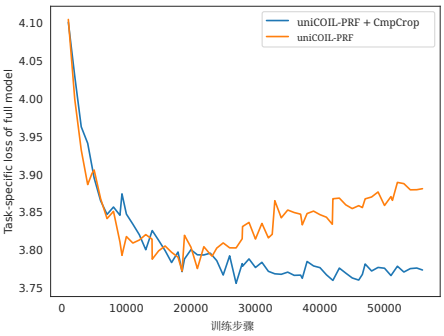
（a）HopotQA 训练集上的答案抽取损失



（b）HopotQA 开发集上的答案抽取损失



（c）MARCO 训练集上的检索损失



（d）MARCO 开发集上的检索损失

图 6。完整模型的任务特定损失曲线。

帮助我们的模型比基线 Longformer 拟合得更好。比较 Longformer + CmpDrop 和 Longformer + CmpCrop 可以发现，前者的训练损失显著更小，这表明带有 CmpDrop 的比较损失有助于模型更好地拟合训练数据。而 Longformer + CmpCrop 的评估损失在后期上升幅度较小，这表明带有 CmpCrop 的比较损失在一定程度上可以缓解过拟合。由于比较损失每个样本优化的任务特定损失数量是常规训练的 $1+c$ 倍，我们还绘制了图 6(c)和图 6(d)中的伪相关反馈模型任务特定损失曲线，其中我们的 uniCOIL-PRF + CmpCrop 的批量大小为基线 uniCOIL-PRF 的 $1/(1+c)$ 。这样，我们的模型和基线模型在一个批次中优化的任务特定损失数量相同，这有助于进一步阐明提交给 ACM 的稿件的作用

表 8. BERT + Cmp 在各 GLUE 数据集上的消融步骤数具体设置，以及相较于 BERT 的性能提升和训练计算开销增幅。

	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	CoLA
c	3	1	4	2	1	2	4	4
性能（%）↑	+1.3	+3.8	+0.9	+0.5	+4.1	+0.4	+0.6	+1.6
浮点运算次数 ↑	x3.5	x1.6	x3.5	x0.9	x2.1	x0.7	x4.8	x3.9

采用 CmpCrop 的比较损失。可以看到，虽然图 6(c)中我们模型的训练损失没有降到基线那么低，但图 6(d)中的评估损失降到了更低的水平，并显著缓解了过拟合。

5.6 训练效率

表 8 展示了 BERT$_{\text{基础型}}$ + Cmp 与 BERT$_{\text{基础型}}$相比的性能提升及训练浮点运算次数的相对变化，以及每个数据集所选用的具体比较次数（即消融步骤数 c）。我们发现，采用比较损失进行训练的实际开销通常不到常规训练的 $1+c$ 倍，甚至低于常规训练（例如在 QQP 上）。这是因为采用比较损失训练的模型往往比基线模型更早收敛。结合图 4中发现的比较损失对比较次数不敏感这一点，我们认为，当数据充足时，将 c 设为 1 或 2，即可实现高效快速的训练。

6 相关工作

本节介绍并讨论一些研究工作。它们的研究动机与我们不同，但在技术上与我们相关。我们将从通过比较来学习的对比学习 [38] 谈起，随后介绍近期同样多次使用随机失活的训练方法。

6.1 对比学习

近年来，对比学习在计算机视觉和自然语言处理领域的表征学习中取得了显著成功。对比学习的核心目标是通过拉近语义相似的邻居、推远不相似的邻居 [26]，来学习有效的表示。它并非逐个从单独的数据样本中学习信号，而是通过比较不同样本 [38]来学习。比较是在相似样本构成的正样本对与不相似样本构成的负样本对之间进行的。正样本对必须确保两个样本彼此相似，这既可以通过使用有监督的相似性标注来构造，也可以通过自监督来构造。在自监督对比学习中，正样本对可以由一个原始样本及其数据增强构成。例如，在计算机视觉领域，SimCLR [12] 将原始图像裁剪、翻转、扭曲或旋转，生成相似视图；在自然语言处理领域，SimCSE [25] 对输入句子应用两个随机失活掩码，以生成两个略有不同的句子嵌入，并将它们作为一对正样本句子嵌入。为共享更多计算并降低成本，负样本对通常由同一训练批次中的两个不相似样本构成。尽管两者都是通过比较来学习，对比学习旨在追求表示的对齐与均匀性 [68] ，而我们的比较损失则旨在追求完整模型及其消融模型的任务特定损失的有序性。此外，正如其词义所示，对比学习仅以二元方式对两个数据样本之间的关系（即相似或不相似）进行分类，而我们的比较损失则比较多个完整模型/消融模型

通过排序模型。然而，这两者并不冲突，我们的比较损失可以用于替代作为任务特定损失的对比损失。

6.2 基于 Dropout 的比较

随机失活是一类用于神经网络训练或推理的随机方法，已引起广泛的研究兴趣，并在实践中得到广泛应用。标准随机失活 [30] 旨在通过减少神经元的协同适应来避免网络过拟合。在协同适应中，单个神经元的输出只有与其他神经元的输出结合时才能提供有用信息。此后，一系列研究着重于采用其他神经元丢弃策略来改进标准随机失活，例如 DropConnect [65] 和变分 Dropout [35]。

与我们相关的一项研究方向是在训练过程中多次使用随机失活。SimCSE [25] 使用不同但丢弃率相同的 Dropout 掩码对模型进行两次前向传播，并通过对比损失约束模型输出在表示空间中的分布。现有文献揭示了随机失活可能产生的一种副作用 [46, 82]：模型的训练阶段与推理阶段之间存在不可忽视的不一致性，即训练时优化的是子模型，而推理时使用的是不含随机失活的完整模型。为解决这种不一致性，R-Drop [40] 通过使用不同的 Dropout 掩码多次运行模型，获得多个预测概率分布，并对它们施加 KL 散度以约束其一致性。与 R-Drop 独立采样多个 Dropout 掩码不同，我们的 CmpDrop 使用逐渐增大的多个丢弃率，且掩码是渐进生成的：后续掩码基于前一个掩码进一步随机丢弃元素而得。此外，我们在最后对任务特定损失施加约束，而非对上游的表示和概率施加约束。值得注意的是，使用 CmpDrop 训练时，通过比较损失也会及时优化完整模型。我们认为，这对于缓解训练与推理之间的不一致性至关重要。这是因为，虽然随机失活可以避免神经元的协同适应，但也会削弱神经元之间的协作（§5.1 提供了一些实证支持）。具体而言，当所有神经元都参与运算时，使用随机失活训练的完整模型并未学会如何让它们高效协同工作，因此在测试时无法充分发挥其作用。令人惊讶的是，我们的 CmpDrop 比较损失能够在促进神经元协作与防止其协同适应之间取得平衡。

7 结论

本文提出了跨模型比较损失这一简单的任务无关损失函数，用于提升自然语言理解模型中神经元的效用。比较损失本质上是一种排序损失，基于完整模型与其消融模型之间的比较原则，其预期是：消融程度越低，任务特定损失越小。为确保多个消融模型之间的可比较性，我们逐步对模型进行消融，并提出了两种基于随机失活和上下文裁剪的受控消融方法，适用于广泛的任务和模型。我们从理论上展示了比较损失的工作原理，表明它能够自适应地为多个任务特定损失分配权重。在来自 3 个不同自然语言理解任务的 14 个数据集上进行的大量实验和分析，证明了比较损失具有普适的有效性。有趣的是，我们的分析证实，比较损失确实能够更恰当地分配权重，并发现比较损失对于参数较少或输入较长的模型尤其有效。

未来，我们希望将比较损失应用于其他领域，例如自然语言生成和计算机视觉，并探索其在 Transformer 以外其他模型架构上的应用。此外，探索在预训练期间将比较损失与自监督损失（例如对比损失）结合使用，也可能很有意义。关于训练成本，如何通过复用更多共享计算来降低开销，是一个值得探究的方向。

此外，训练中更先进的消融方法，例如使用 DropConnect [65] 而非标准 Dropout，以及采用对抗式方法而非随机方法，或许值得在未来投入研究。

致谢

本研究得到国家重点研发计划（2022YFB3103700、2022YFB3103704）、国家自然科学基金委员会（NSFC）项目（编号：62276248、U21B2046）以及中国科学院青年创新促进会项目（编号：2023111）的资助。

参考文献

[1] R. Attar 和 A. S. Fraenkel。1977年。《全文检索系统中的局部反馈》。《ACM学报》24卷第3期（1977年7月），397–417。https://doi.org/10.1145/322017. 322021[2] Brian Bartoldson、Ari Morcos、Adrian Barbu 和 Gordon Erlebacher。2020年。《神经网络剪枝中的泛化-稳定性权衡》。载于《神经信息处理系统进展》第33卷。Curran Associates, Inc.，20852–20864。https://proceedings.neurips.cc/paper/2020/hash/ ef2ee09ea9551de88bc11fd7eeea93b0-Abstract.html[3] Iz Beltagy、Matthew E. Peters 和 Arman Cohan。2020年。《Longformer：长文档Transformer》。https://doi.org/10.48550/arXiv.2004.05150 arXiv:2004.05150[4] Luisa Bentivogli、Peter Clark、Ido Dagan 和 Danilo Giampiccolo。2009年。《第五届 PASCAL 文本蕴涵识别挑战赛》。载于文本分析会议。[5] Davis Blalock、Jose Javier Gonzalez Ortiz、Jonathan Frankle 和 John Guttag。2020年。《神经网络剪枝的现状如何？》《机器学习与系统会议论文集》2卷（2020年3月），129–146。https://proceedings.mlsys.org/paper/2020/hash/d2ddea18f00665ce8623e36bd4e3c7c5- Abstract.html[6] Sebastian Borgeaud、Arthur Mensch、Jordan Hoffmann、Trevor Cai、Eliza Rutherford、Katie Millican、George Bm Van Den Driessche、Jean-Baptiste Lespiau、Bogdan Damoc、Aidan Clark、Diego De Las Casas、Aurelia Guy、Jacob Menick、Roman Ring、Tom Hennigan、Saffron Huang、Loren Maggiore、Chris Jones、Albin Cassirer、Andy Brock、Michela Paganini、Geoffrey Irving、Oriol Vinyals、Simon Osindero、Karen Simonyan、Jack Rae、Erich Elsen 和 Laurent Sifre。2022年。《通过从数万亿个词元中检索来改进语言模型》。载于第39届国际机器学习会议论文集。PMLR，2206–2240。https://proceedings.mlr.press/v162/borgeaud22a.html[7] Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel Ziegler、Jeffrey Wu、Clemens Winter、Chris Hesse、Mark Chen、Eric Sigler、Mateusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。2020年。《语言模型是少样本学习者》。载于第34届神经信息处理系统国际会议论文集，第33卷。Curran Associates, Inc.，美国纽约州雷德胡克，1877–1901。https: // papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html[8] Alon Brutzkus 和 Amir Globerson。2019年。《为什么更大的模型泛化能力更强？通过异或问题给出的理论视角》。载于第36届国际机器学习会议论文集。PMLR，822–830。https://proceedings.mlr.press/v97/brutzkus19b.html[9] Daniel Cer、Mona Diab、Eneko Agirre、Iñigo Lopez-Gazpio 和 Lucia Specia。2017年。《SemEval-2017任务1：语义文本相似度的多语言与跨语言聚焦评测》。载于第11届语义评测国际研讨会论文集（SemEval-2017）。计算语言学协会，温哥华，加拿大，1–14。https://doi.org/10.18653/v1/S17-2001[10] Yi Chang 和 Hongbo Deng。2020年。《搜索引擎的查询理解》。施普林格。[11] Danqi Chen。2018年。《神经阅读理解及其他》。斯坦福大学。[12] Ting Chen、Simon Kornblith、Mohammad Norouzi 和 Geoffrey Hinton。2020年。《视觉表示对比学习的简单框架》。载于第37届国际机器学习会议论文集。PMLR，1597–1607。https://proceedings.mlr.press/v119/chen20j.html[13] Zihan Chen、Hongbo Zhang、Xiaoji Zhang 和 Leqi Zhao。2018年。《Quora问题对》。https://www.kaggle.com/c/quora-question-pairs[14] Christopher Clark 和 Matt Gardner。2018年。《简单而有效的多段落阅读理解》。载于计算语言学协会第56届年会论文集（第1卷：长篇论文）。计算语言学协会，墨尔本，澳大利亚，845–855。https://doi.org/10.18653/v1/P18-1078[15] Kevin Clark、Minh-Thang Luong、Quoc V. Le 和 Christopher D. Manning。2020年。《ELECTRA：将文本编码器预训练为判别器，而非生成器》。载于国际学习表征会议。https://openreview.net/forum?id=r1xMH1BtvB[16] Stéphane Clinchant 和 Eric Gaussier。2013年。《伪相关反馈模型的理论分析》。载于2013年信息检索理论会议论文集（ICTIR '13）。美国计算机协会，纽约，美国，6–13。https://doi.org/10.1145/2499178.2499179[17] Paul R. Cohen 和 Adele E. Howe。1988年。《评估如何指导人工智能研究：传达的信息仍比媒介更重要》。《人工智能杂志》9卷第4期（1988年12月），35–35。https://doi.org/10.1609/aimag.v9i4.952[18] Kevyn Collins-Thompson。2009年。《通过鲁棒约束优化降低查询扩展的风险》。载于第18届 ACM 信息与知识管理会议论文集（CIKM '09）。美国计算机协会，纽约，美国，837–846。https: // doi.org/10.1145/1645953.1646059

[19] Nick Craswell、Bhaskar Mitra、Emine Yilmaz 和 Daniel Campos。2021年。《TREC 2020 深度学习赛道概述》。arXiv:2102.*07662 [*cs*]*（2021年2月）。arXiv:2102.07662 http://arxiv.org/abs/2102.07662[20] Nick Craswell、Bhaskar Mitra、Emine Yilmaz、Daniel Campos 和 Ellen M. Voorhees。2020年。《TREC 2019 深度学习赛道概述》。arXiv:2003.*07820 [*cs*]*（2020年3月）。arXiv:2003.07820 http://arxiv.org/abs/2003.07820[21] Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。《BERT：用于语言理解的深度双向变换器预训练》。载于2019年计算语言学协会北美分会年会：人类语言技术会议论文集，第1卷（长篇与短篇论文）。计算语言学协会，明尼阿波利斯，明尼苏达州，4171–4186。https: //doi.org/10.18653/v1/N19-1423[22] William B. Dolan 和 Chris Brockett。2005 年。《自动构建句子改述语料库》。载于第三届国际改述研讨会论文集（ IWP2005 ）。https://aclanthology.org/I05-5002[23] Dheeru Dua、Cicero Nogueira dos Santos、Patrick Ng、Ben Athiwaratkun、Bing Xiang、Matt Gardner 和 Sameer Singh。2021年。《用于多跳问答的生成式上下文对选择》。载于2021年自然语言处理实证方法会议论文集。计算语言学协会，线上，蓬塔卡纳，多米尼加共和国，7009–7015。https://doi.org/10.18653/v1/2021.emnlp-main.561[24] Stella Frank、Emanuele Bugliarello 和 Desmond Elliott。2021年。《视觉与语言，还是面向语言的视觉？多模态变换器中的跨模态影响》。载于2021年自然语言处理实证方法会议论文集。计算语言学协会，线上，蓬塔卡纳，多米尼加共和国，9847–9857。https://doi.org/10.18653/v1/2021.emnlp-main.775[25] Tianyu Gao、Xingcheng Yao 和 Danqi Chen。2021年。《SimCSE：句子嵌入的简单对比学习》。载于2021年自然语言处理实证方法会议论文集。计算语言学协会，线上，蓬塔卡纳，多米尼加共和国，6894–6910。https://doi.org/10.18653/v1/2021.emnlp-main.552[26] Raia Hadsell、Sumit Chopra 和 Yann LeCun。2006年。《通过学习不变映射进行降维》。载于2006年 IEEE计算机学会计算机视觉与模式识别会议（CVPR'06），第2卷。IEEE，1735–1742。[27] Song Han、Jeff Pool、John Tran 和 William Dally。2015年。《为高效神经网络同时学习权重与连接》。载于《神经信息处理系统进展》第28卷。Curran Associates, Inc. https://papers.nips.cc/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6- Abstract.html[28] Dan Hendrycks 和 Kevin Gimpel。2016年。《高斯误差线性单元（GeLU）》。arXiv预印本 arXiv:1606.08415（2016年）。[29] Ralf Herbrich、Thore Graepel 和 Klaus Obermayer。2000年。《序数回归的大间隔排序边界》。《大间隔分类器进展》88卷第2期（2000年），115–132。[30] Geoffrey E. Hinton、Nitish Srivastava、Alex Krizhevsky、Ilya Sutskever 和 Ruslan R. Salakhutdinov。2012年。《通过防止特征检测器协同适应来改进神经网络》。https://doi.org/10.48550/arXiv.1207.0580 arXiv:1207.0580[31] L. Hirschman 和 R. Gaizauskas。2001年。《自然语言问答：我们的视角》。《自然语言工程》7卷第4期（2001年12月），275–300。https://doi.org/10.1017/S1351324901002807[32] Gautier Izacard、Patrick Lewis、Maria Lomeli、Lucas Hosseini、Fabio Petroni、Timo Schick、Jane Dwivedi-Yu、Armand Joulin、Sebastian Riedel 和 Edouard Grave。2022年。《通过检索增强语言模型进行少样本学习》。https://doi.org/10.48550/arXiv.2208.03299 arXiv:2208.03299[33] Jared Kaplan、Sam McCandlish、Tom Henighan、Tom B. Brown、Benjamin Chess、Rewon Child、Scott Gray、Alec Radford、Jeffrey Wu 和 Dario Amodei。2020年。《神经语言模型的缩放定律》。https://doi.org/10.48550/arXiv.2001.08361 arXiv:2001.08361[34] Vladimir Karpukhin、Barlas Oguz、Sewon Min、Patrick Lewis、Ledell Wu、Sergey Edunov、Danqi Chen 和 Wen-tau Yih。2020年。《用于开放域问答的密集段落检索》。载于2020年自然语言处理实证方法会议论文集（EMNLP）。计算语言学协会，线上，6769–6781。https://doi.org/10.18653/v1/2020.emnlp-main.550[35] Durk P Kingma、Tim Salimans 和 Max Welling。2015年。《变分丢弃与局部重参数化技巧》。载于《神经信息处理系统进展》第28卷。Curran Associates, Inc. https://papers.nips.cc/paper/2015/hash/bc7316929fe1545bf0b98d114ee3ecb8-Abstract.html[36] Alex Labach、Hojjat Salehinejad 和 Shahrokh Valaee。2019 年。《深度神经网络丢弃方法综述》。https://doi.org/10.48550/arXiv. 1904.13310 arXiv:1904.13310[37] Zhenzhong Lan、Mingda Chen、Sebastian Goodman、Kevin Gimpel、Piyush Sharma 和 Radu Soricut。2020年。《ALBERT：用于语言表示自监督学习的轻量版 BERT》。载于国际学习表征会议。https://openreview.net/forum?id=H1eA7AEtvS[38] Phuc H. Le-Khac、Graham Healy 和 Alan F. Smeaton。2020年。《对比表征学习：框架与综述》。《IEEE Access》8卷（2020年），193907–193934。https://doi.org/10.1109/ACCESS.2020.3031549[39] Yann LeCun、John Denker 和 Sara Solla。1989年。《最优脑损伤》。载于《神经信息处理系统进展》第2卷。Morgan- Kaufmann。https://papers.nips.cc/paper/1989/hash/6c9882bbac1c7093bd25041881277658-Abstract.html[40] xiaobo liang、Lijun Wu、Juntao Li、Yue Wang、Qi Meng、Tao Qin、Wei Chen、Min Zhang 和 Tie-Yan Liu。2021年。《R-Drop：神经网络的正则化丢弃》。载于《神经信息处理系统进展》第34卷。Curran Associates, Inc.，10890–10905。https://proceedings.neurips.cc/paper/2021/hash/5a66b9200f29ac3fa0ae244cc2a51b39-Abstract.html[41] Jimmy Lin。2002年。《网络作为问答资源：展望与挑战》。载于第三届国际语言资源与评测会议论文集（LREC'02）。欧洲语言资源协会（ELRA），拉斯帕尔马斯，加那利群岛，西班牙。http: //www.lrec-conf.org/proceedings/lrec2002/pdf/85.pdf[42] Jimmy Lin 和 Xueguang Ma。2021年。《关于 DeepImpact、COIL 及信息检索技术概念框架的几点简短说明》。arXiv:2106.*14807 [*cs*]*（2021年6月）。arXiv:2106.14807 http://arxiv.org/abs/2106.14807 提交给 ACM 的稿件

[43] Shanshan Liu、Xin Zhang、Sheng Zhang、Hui Wang 和 Weiming Zhang。2019年。《神经机器阅读理解：方法与趋势》。Applied Sciences 9, 18（2019年1月），3698。https://doi.org/10.3390/app9183698[44] Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Mandar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。2019年。RoBERTa：一种经过鲁棒优化的 BERT 预训练方法。arXiv:1907.*11692 [*cs*]*（2019年7月）。arXiv:1907.11692 http://arxiv.org/abs/1907.11692[45] Zhuang Liu、Jianguo Li、Zhiqiang Shen、Gao Huang、Shoumeng Yan 和 Changshui Zhang。2017年。通过网络瘦身学习高效卷积网络。载于《2017年 IEEE 国际计算机视觉会议（ICCV）》。2755–2763。https://doi.org/10.1109/ICCV.2017.298[46] Xuezhe Ma、Yingkai Gao、Zhiting Hu、Yaoliang Yu、Yuntian Deng 和 Eduard Hovy。2016年。采用期望线性正则化的随机失活。arXiv 预印本 arXiv:1609.08017（2016年）。[47] Iain Mackie、Jeffrey Dalton 和 Andrew Yates。2021年。你的学习有多深入：DL-HARD 标注深度学习数据集。载于《第44届 ACM SIGIR 信息检索研究与发展国际会议论文集》。美国计算机协会，纽约州纽约市，2335–2341。https://doi.org/10.1145/3404835.3463262[48] Richard Meyes、Melanie Lu、Constantin Waubert de Puiseau 和 Tobias Meisen。2019年。人工神经网络的消融研究。https: //doi.org/10.48550/arXiv.1901.08644 arXiv:1901.08644[49] Sewon Min、Victor Zhong、Richard Socher 和 Caiming Xiong。2018年。基于最少上下文从文档中高效、鲁棒地进行问答。载于《计算语言学协会第56届年会论文集（第1卷：长篇论文）》。计算语言学协会，澳大利亚墨尔本，1725–1735。https://doi.org/10.18653/v1/P18-1160[50] Mandar Mitra、Amit Singhal 和 Chris Buckley。1998年。改进自动查询扩展。载于《第21届 ACM SIGIR 信息检索研究与发展国际会议论文集（SIGIR ’98）》。美国计算机协会，纽约州纽约市，206–214。https://doi.org/10.1145/290941.290995[51] Tri Nguyen、Mir Rosenberg、Xia Song、Jianfeng Gao、Saurabh Tiwary、Rangan Majumder 和 Li Deng。2016年。MS MARCO：人工生成的机器阅读理解数据集。载于 CoCo@ NIPS。[52] Shuzi Niu、Jiafeng Guo、Yanyan Lan 和 Xueqi Cheng。2012年。Top-k 学习排序：标注、排序与评估。载于《第35届 ACM SIGIR 信息检索研究与发展国际会议论文集（美国俄勒冈州波特兰）（SIGIR ’12）》。美国计算机协会，纽约州纽约市，751–760。https://doi.org/10.1145/2348283.2348384[53] Liang Pang、Yanyan Lan 和 Xueqi Cheng。2021年。Match-Ignition：将 PageRank 融入 Transformer，以实现长文本匹配。载于《第30届 ACM 信息与知识管理国际会议论文集（线上会议，澳大利亚昆士兰州）（CIKM ’21）》。美国计算机协会，纽约州纽约市，1396–1405。https://doi.org/10.1145/3459637.3482450[54] Liang Pang、Yanyan Lan、Jiafeng Guo、Jun Xu、Lixin Su 和 Xueqi Cheng。2019年。HAS-QA：用于开放域问答的层次化答案片段模型。《AAAI人工智能会议论文集》33, 01（2019年7月），6875–6882。https://doi.org/10.1609/aaai.v33i01. 33016875[55] Liang Pang、Yanyan Lan、Jiafeng Guo、Jun Xu、Shengxian Wan 和 Xueqi Cheng。2016年。将文本匹配视为图像识别。载于《第三十届 AAAI人工智能会议论文集（亚利桑那州菲尼克斯）（AAAI’16）》。AAAI Press，2793–2799。[56] Liang Pang、Jun Xu、Qingyao Ai、Yanyan Lan、Xueqi Cheng 和 Jirong Wen。2020年。SetRank：学习用于信息检索的置换不变排序模型。载于《第43届 ACM SIGIR 信息检索研究与发展国际会议论文集（SIGIR ’20）》。美国计算机协会，纽约州纽约市，499–508。https://doi.org/10.1145/3397271.3401104[57] Colin Raffel、Noam Shazeer、Adam Roberts、Katherine Lee、Sharan Narang、Michael Matena、Yanqi Zhou、Wei Li 和 Peter J. Liu。2020年。探索统一文本到文本 Transformer 的迁移学习极限。《机器学习研究杂志》21, 140（2020年），1–67。http: // jmlr.org/papers/v21/20-074.html[58] Pranav Rajpurkar、Robin Jia 和 Percy Liang。2018年。了解自己不知道什么：SQuAD 中的不可回答问题。载于《计算语言学协会第56届年会论文集（第2卷：短篇论文）》。计算语言学协会，澳大利亚墨尔本，784–789。https://doi.org/10.18653/v1/P18-2124[59] Pranav Rajpurkar、Jian Zhang、Konstantin Lopyrev 和 Percy Liang。2016年。SQuAD：100，000+ 用于文本机器理解的问题。载于《2016年自然语言处理实证方法会议论文集》。计算语言学协会，美国得克萨斯州奥斯汀，2383–2392。https://doi.org/10.18653/v1/D16-1264[60] Pedro Rodriguez 和 Jordan Boyd-Graber。2021年。问答中的评估范式。载于《2021年自然语言处理实证方法会议论文集》。计算语言学协会，线上及多米尼加共和国蓬塔卡纳，9630–9642。https://doi.org/10.18653/v1/2021.emnlp-main.758[61] Richard Socher、Alex Perelygin、Jean Wu、Jason Chuang、Christopher D Manning、Andrew Y Ng 和 Christopher Potts。2013年。基于情感树库的语义组合性递归深度模型。载于《2013年自然语言处理实证方法会议论文集》。1631–1642。[62] Ming Tu、Kevin Huang、Guangtao Wang、Jing Huang、Xiaodong He 和 Bowen Zhou。2020年。选择、回答并解释：针对多篇文档的可解释多跳阅读理解。载于《AAAI人工智能会议论文集》，第34卷。9073–9080。https://doi.org/10.1609/aaai.v34i05.6441[63] Vladimir Vapnik。1991年。学习理论中的风险最小化原则。《神经信息处理系统进展》4（1991年）。[64] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。Attention Is All You Need。载于《神经信息处理系统进展》，第30卷。Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/

hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[65] Li Wan、Matthew Zeiler、Sixin Zhang、Yann Le Cun 和 Rob Fergus。2013年。使用 DropConnect 对神经网络进行正则化。载于《第30届国际机器学习会议论文集》。PMLR，1058–1066。https://proceedings.mlr.press/v28/wan13.html[66] Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R. Bowman。2019年。GLUE：自然语言理解多任务基准与分析平台。载于《国际学习表征会议》。https://openreview.net/forum?id=rJ4km2R5t7[67] Shuohang Wang、Yichong Xu、Yuwei Fang、Yang Liu、Siqi Sun、Ruochen Xu、Chenguang Zhu 和 Michael Zeng。2022年。训练数据比你想象的更有价值：一种通过从训练数据中检索实现的简单有效方法。载于《计算语言学协会第60届年会论文集（第1卷：长篇论文）》。计算语言学协会，爱尔兰都柏林，3170–3179。https: //doi.org/10.18653/v1/2022.acl-long.226[68] Tongzhou Wang 和 Phillip Isola。2020年。通过超球面上的对齐与均匀性理解对比表征学习。载于《第37届国际机器学习会议论文集》。PMLR，9929–9939。https://proceedings.mlr.press/v119/wang20k.html[69] Alex Warstadt、Amanpreet Singh 和 Samuel R Bowman。2019年。神经网络可接受性判断。《计算语言学协会汇刊》7（2019年），625–641。[70] Adina Williams、Nikita Nangia 和 Samuel Bowman。2018年。通过推理理解句子的广覆盖挑战语料库。载于《2018年计算语言学协会北美分会会议：人类语言技术论文集，第1卷（长篇论文）》。计算语言学协会，美国路易斯安那州新奥尔良，1112–1122。https://doi.org/10.18653/v1/N18-1101[71] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Remi Louf、Morgan Funtowicz、Joe Davison、Sam Shleifer、Patrick von Platen、Clara Ma、Yacine Jernite、Julien Plu、Canwen Xu、Teven Le Scao、Sylvain Gugger、Mariama Drame、Quentin Lhoest 和 Alexander Rush。2020年。Transformers：自然语言处理的最先进技术。载于《2020年自然语言处理实证方法会议论文集：系统演示》。计算语言学协会，线上，38–45。https://doi.org/10.18653/v1/2020.emnlp-demos.6[72] Lee Xiong、Chenyan Xiong、Ye Li、Kwok-Fung Tang、Jialin Liu、Paul N. Bennett、Junaid Ahmed 和 Arnold Overwijk。2020年。用于稠密文本检索的近似最近邻负对比学习。载于《国际学习表征会议》。https: //openreview.net/forum?id=zeFrfgyZln[73] Shicheng Xu、Liang Pang、Huawei Shen 和 Xueqi Cheng。2022年。Match-Prompt：通过提示学习提升神经文本匹配的多任务泛化能力。载于《第31届 ACM 信息与知识管理国际会议论文集（美国佐治亚州亚特兰大）（CIKM '22）》。美国计算机协会，纽约州纽约市，2290–2300。https://doi.org/10.1145/3511808.3557388[74] Zhilin Yang、Peng Qi、Saizheng Zhang、Yoshua Bengio、William Cohen、Ruslan Salakhutdinov 和 Christopher D. Manning。2018年。HotpotQA：用于多样化、可解释多跳问答的数据集。载于《2018年自然语言处理实证方法会议论文集》。计算语言学协会，2369–2380。https://doi.org/10.18653/v1/D18-1259[75] HongChien Yu、Chenyan Xiong 和 Jamie Callan。2021年。利用伪相关反馈改进稠密检索的查询表示。载于《第30届 ACM 信息与知识管理国际会议论文集（CIKM '21）》。美国计算机协会，纽约州纽约市，3592–3596。https://doi.org/10.1145/3459637.3482124[76] Zhi Zheng、Kai Hui、Ben He、Xianpei Han、Le Sun 和 Andrew Yates。2020年。BERT-QE：用于文档重排序的上下文化查询扩展。载于《计算语言学协会研究成果：EMNLP 2020》。计算语言学协会，线上，4718–4728。https://doi.org/10.18653/v1/2020.findings-emnlp.424[77] Ruiqi Zhong、Dhruba Ghosh、Dan Klein 和 Jacob Steinhardt。2021年。规模更大的预训练语言模型是否始终更好？实例级性能比较。载于《计算语言学协会研究成果：ACL-IJCNLP 2021》。计算语言学协会，线上，3813–3827。https://doi.org/10.18653/v1/2021.findings-acl.334[78] Yunchang Zhu、Liang Pang、Yanyan Lan 和 Xueqi Cheng。2020年。L2R^2：借助排序进行溯因推理。载于《第43届 ACM SIGIR 信息检索研究与发展国际会议论文集》。美国计算机协会，纽约州纽约市，1961–1964。https://doi.org/10.1145/3397271.3401332[79] Yunchang Zhu、Liang Pang、Yanyan Lan、Huawei Shen 和 Xueqi Cheng。2021年。面向开放域问答的自适应信息寻求。载于《2021年自然语言处理实证方法会议论文集》。计算语言学协会，线上及多米尼加共和国蓬塔卡纳，3615–3626。https://doi.org/10.18653/v1/2021.emnlp-main.293[80] Yunchang Zhu、Liang Pang、Yanyan Lan、Huawei Shen 和 Xueqi Cheng。2022年。LoL：针对伪相关反馈查询改写损失的比较正则化损失。载于《第45届 ACM SIGIR 信息检索研究与发展国际会议论文集（SIGIR '22）》。美国计算机协会，纽约州纽约市，825–836。https://doi.org/10.1145/3477495.3532017[81] Liron Zighelnic 和 Oren Kurland。2008年。鲁棒查询扩展中的查询漂移预防。载于《第31届 ACM SIGIR 信息检索研究与发展国际会议论文集（SIGIR '08）》。美国计算机协会，纽约州纽约市，825–826。https://doi.org/10.1145/1390334.1390524[82] Konrad Zolna、Devansh Arpit、Dendi Suhubdy 和 Yoshua Bengio。2017年。兄弟式 Dropout。arXiv 预印本 arXiv:1711.00066（2017年）。