南京大学

计算机系/重点实验室技术报告系列

学 科 门 类 ： 工学

学科、专业 ： 计算机科学与技术

研 究 方 向 ： 软件分析与测试

南 京 大 学 计 算 机 科 学 与 技 术 系

2024 年 3 月 1 日

博士在读期间科研成果列表

（隐去学生及导师信息）

1、CCF B类 期刊 第一作者 一篇（大修）。

南京大学

计算机系技术报告

复杂环境下的深度学习模型融合与测试技术研究

2024 年 3 月

目录

1 研究背景及意义

近年来，深度学习模型在许多安全攸关场景中得到广泛应用，如医疗 [43]、自动驾驶 [11] 和语音识别 [86] 等领域。在这些复杂应用环境下，虽然深度学习模型已达到甚至超越人类的能力，但深度学习模型在开发维护 [53] 和质量保障 [13] 等方面中仍然面临挑战。在深度学习模型开发方面，正如在文献 [56] 中所述，深度学习模型的训练过程需要付出巨大的努力，调整最优超参数需要专业知识和大量的尝试，这是一项繁琐且耗时的任务。在质量保障方面，例如，Tesla 的自动驾驶汽车事故就造成了人员财物上的巨大损失 [17]。因此，迫切需要相应的技术去提高深度学习模型开发维护效率和安全性以应对复杂的应用和开发环境。

深度学习模型开发与维护：参考之前的研究 [8]，其将模型开发维护过程分为了 9 个阶段，分别是（1）模型需求；（2）数据收集；（3）数据清洗；（4）数据标注；（5）特征工程；（6）模型训练；（7）模型评估；（8）模型部署和（9）模型监控。在这 9 个阶段中，数据标注和模型训练通常需要大量的人力资源和计算资源（如标记成本和参数训练）。更严峻的是，如果模型的需求发生了改变，就要先重新对数据进行标注，然后重新对模型进行训练。因此，如何提高模型随需求改变而进行演化的能力，在软件工程领域受到了广泛关注。

深度学习模型质量保障：模型质量涵盖多方面指标，包括正确性和安全性等 [93]。影响深度学习模型和传统软件正确性和安全性因素不同。传统软件主要依赖于程序员手动构建业务逻辑，深度学习模型主要通过数据驱动的编程范式来构建，其决策逻辑通过训练过程来获得，即模型的行为和质量可能会随着新的训练数据的加入而发生改变 [8]。因此对于深度学习模型，除源代码之外，数据集和模型架构是影响深度学习系统质量的重要因素。

对于传统软件，软件复用是提高软件开发维护的常用方法，其通过复用已经已有的代码来提高开发维护效率 [65]。深度学习模型复用技术，尝试复用在其他任务上训练的已有可用的模型来构建模型，这为提高模型随需求改变而进行演化的能力提供了巨大潜力，因为通过复用已有模型可以继承已有模型的内在知识，有助于为新模型奠定良好的基础，从而减少训练数据的需求量和训练时间。近年来，在深度学习模型复用领域，学者们进行了广泛研究。例如，文献 [99] 强调可重用性是学习软件新概念的一个关键特征。如果模型可以在与原始训练场景有很大不同的场景中得到重用，那该模型具有良好的可重用性。深度学习的快速发展，开越来越多的预训练模型被发者发布出来。经过训练的模型可以被视为相应任务的专家 [88]。如何复用预训练模型中的知识引起了学者们的关注。模型融合技术提供了模型知识复用的可能技术。例如，You 等人 [91] 提出了一种从多个教师模型中提取知识来训练学生模型的方法。在进行知识融合时，不仅要考虑实际教师模型的输出，还要考虑模型中间层不同样本的表示。Shen 等 [71] 提出通过知识合并完成综合分类任务，但限制教师模型必须是同构的。在进一步的研究中，Shen 等 [72] 提出通过知识合并从异构教师模型中获得学生模型。

软件测试是软件质量保障的常用方法，通过暴露和修复软件系统问题，进而提高软件系统的质量 [93]。深度学习模型的统计特性为深度学习模型测试和增强带来了新的问题和挑战。近年来，在深度学习测试领域，学者们进行了广泛的研究。深度学习模型的测试问题，可以被描述为测试的工作流程（如何去测试）、测试属性（为什么去测试）以及测试组件（去哪里测试和测什么）。例如，对于安全性的测试旨在尽可能高效的找出模型的错误，Pei 等 [64] 提

出的一种白盒测试技术 DeepXplore，通过筛选较高神经元覆盖率的输入样本，触发了自动驾驶学习系统中数以千计的错误驾驶行为。Tian 等 [79] 提出了 DeepTest，该方法使用九种不同的真实图像变换执行贪婪搜索，在 Udacity 自动驾驶数据挑战集上检查出 1,000 多个深度学习模型的错误。对于深度学习模型增强的研究主要集中在训练数据集精化和深度学习模型的权重数值方面 [73]。深度学习通常被认为是一种依赖"大"数据的方法，因此数据质量对模型的质量有着重要的影响：即使拥有再好的深度学习模型结构，没有好的训练数据也无法很好地执行。但深度学习模型的测试通常受到有限标记代价的限制，因此它必须采用具有统计效率的"小"数据方法 [42]。如何在有限标记代价下更好地增强模型的性能，引起学者们的广泛关注。例如，Feng 等 [25] 一种黑盒测试技术 DeepGini, 该方法选择重训练样本对模型进行重训练来增强模型。

综上所述，深度学习的模型融合与测试技术是保障开发维护效率和安全性的两个重要环节，当将深度学习模型视为一种新型软件时，是时候强调软件工程对于深度学习 [54] 的重要性了。软件工程可以帮助开发深度学习模型并提高其质量，例如，通过模型融合技术提高模型开发和维护效率，通过深度学习测试技术发现模型的缺陷。本文针对深度学习模型融合和测试开展了两项工作：

第一个研究工作：我们从一个新的应用场景来研究深度学习模型融合技术。如前所述，开发和维护深度学习模型的主要挑战之一是深度学习模型的训练过程和数据标注过程。特别是对于具有数十亿参数和大型数据集的模型 [41, 73]。解决这个问题的软件工程方法是借鉴传统软件重用的思想 [65]，即复用深度模型 [52, 61]。当前的模型重用研究包括两个方面：模型切片 [96] 和模型模块化 [60, 66]，两者都旨在通过简化缩小深度学习模型以适应给定任务的特定要求。隐藏的逻辑是，如果深度学习模型配备了过多的功能（即，可能只需要模型功能的一小部分来解决目标问题的要求），就会导致更长的推理时间和运行成本。因此，首选重用与特定问题相关的训练模型的某些组件 [67]。

与之前的软件工程研究不同，我们关注的是更复杂的重用场景，即需要重用多个深度学习模型。正如 [52] 中所述，随着模型构建和共享技术的成熟，产生了多个具有相似功能的深度学习模型，即功能上具有重叠性。面对如此多种深度学习模型，开发人员可能需要将它们组合起来，构建联合模型来实现统一的需求。具体来说，以我们研究的真实数据集为例，图 1 显示有两个深度学习模型 A 和 B，其功能都是是将输入分类为各种动物。如图 1 所示，模型 A 和 B 具有功能重叠，即两者都可以识别猫和狗的输入。此外，我们也注意到模型 A（B）有其独特的输出类别，熊猫（大象、马和狮子）。这个例子说明了一个更复杂的重用场景：当所有输出类别都被视为新深度学习模型的功能需求时，开发人员将致力于组合现有深度学习模型（即连接模型 A 和 B）来构建联合模型（即模型 C 如图 1 所示）

从软件工程的角度来看，我们指出结合深度学习模型来满足新需求时的两个局限性。

- 在实际应用中，我们通常只能获得多个模型（即模型文件）作为研究对象，而没有它们的训练数据集，这阻碍了从训练数据集构建新的联合模型
- 面对新的需求，需要对新数据进行标记。由于人工标记的成本很高，标记的数据是有限的，即仅应用中的一小部分样本可以被标记。

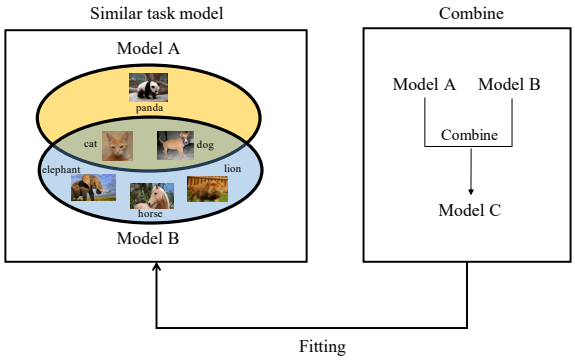


图 1: 这是一个展示我们研究场景的示例，开发人员旨在组合两个深度学习模型（模型 A 和 B）来构建联合模型（模型 C）。

为了解决上述限制，我们的研究密切关注“如何以有限的标记工作从应用程序上下文中提取信息”，具体的研究过程分为两个步骤。首先，我们从 Kaggle 平台收集数据集和模型，并进行实证研究，观察具有重叠功能的多个深度学习模型的性能，并提出了组合多个模型的需求。其次，我们提出了基于级联并行的模型融合方法，称为 MCCP。为了探讨模型组合在这种场景下的好处，以及 MCCP 是否具有更好的性能，我们通过回答以下两个研究点来探讨：

- **研究点 1（Why）：重叠类别数据中多个模型之间的性能是否存在明显差异?**

  实验结果显示两个重叠类别的数据分布存在差异（约 1% 到 38%）。通过模型组合性能可以满足全分类要求。

- **研究点 2（How）：我们的方法比基线更有效吗?**

  在收集到的数据集上，我们对学生模型（即组合模型）在不同采样率下进行了训练，评估了混合测试集上的分类精度，将 MCCP 与 3 个基线进行了比较，发现在大多数情况下（9 个场景中的 8 个场景）MCCP 优于基线。

本研究问题的主要贡献如下：

- 我们提出了一种简单直接的模型融合方法，称为 MCCP，它通过组合训练好的模型来生成组合模型，并以有限的标记代价更好地训练组合模型覆盖了整体分类。
- 我们对模型组合技术进行了实证研究，将 MCCP 与 9 个不同分类领域数据集上的 3 个基线进行了比较。实验结果表明，MCCP 非常有效，即可以在有限的标记成本下对总体混合数据进行分类，并且与基线相比具有更好的性能。

第二个研究工作，对于深度学习模型测试方面，我们主要关注深度学习模型后门防御研究。由于深度神经网络模型通常采用一些来自不可信第三方的外部训练数据，这些不可信的外部训练数据可能会含有被恶意攻击者投毒的样本（含有后门触发器的样本），然后通过训练给神经网络模型植入了后门。因此强大的后门防御策略非常重要。我们认为防御的核心是从中毒的训练集中将中毒样本识别出来。

一些恶意攻击者对深度学习模型植入后门 [37,40,47]，使得模型在面对中毒样本预测结果

与实际结果产生差异。由于深度学习模型通常被看做黑盒子，其内部逻辑很难被理解，因此深度学习模型的后门很难被发现与修复，对模型的准确性和安全性产生了很大的影响。学者们为了缓解后门攻击带来的影响提出了一系列防御方法。大体可以分为模型防御和输入防御两大类。在模型防御方面：Liu 等 [45] 提出名为 ABS 的技术，用于扫描深度学习模型以检测是否被植入了后门；在输入防御方面：Gao 等 [27] 提出名为 STRIP 技术，它是一个基于强烈的故意扰动的运行时木马攻击检测系统。具体来说，其通过故意扰乱被检测的输入，例如叠加各种图像模式，并观察预测的扰动输入的类别，预测类别中的低熵意味着存在木马输入的特征。基于以上的研究，我们也是从输入防御角度出发提出基于熵的木马输入检测 ETID（Entropy-based Trojaned Input Detection）我们的方法与先前的检测技术有明显的区别：1）我们的假设是木马样本对模型决策边界更不敏感；2）我们采用变异测试技术对被植入后门的模型进行变异来修改其决策边界。我们在 3 种广泛应用的数据集和 6 种攻击操作的场景下对 ETID 进行了评估，实验结果表明，我们提出的检测方法对木马输入样本检测是有效的。

本文后续内容按如下结构进行组织：第二章介绍深度学模型融合和深度学习模型测试的相关工作；第三章描述待解决的两个问题；第四章提供了对两个问题的解决思路和方案；最后一章对已完成的研究工作进行总结并规划未来的研究计划。

2 相关工作

本章介绍与深度学习模型融合和测试技术相关的工作，主要从深度学习模型模型融合技术和深度学习模型测试技术两个部分进行介绍。

2.1 深度学习模型融合

2.1.1 知识复用

随着深度学习的快速发展，开发者发布了越来越多的预训练模型。一个经过训练的模型可以被视为其相应任务的专家 [88]。如何重用模型中包含的知识引起了学者们的关注。对此，我们从以下三个方面介绍相关工作：模型重用、知识蒸馏和知识融合。

2.1.2 模型重用

模型复用的关键一步是从模型池中选择对当前任务有帮助的模型进行复用。Ding 等人 [20] 研究了从模型池中选择模型子集来对未标记的数据集进行分类，而无需访问模型的原始训练数据。Meng 等人 [52] 提出了一种可以以有限的标记成本对源模型进行排序的方法，以从具有相同功能的多个模型中选择更适合目标任务的模型。Zhang 等人 [94] 研究了模型池中的模型如何在不完全覆盖目标任务的情况下识别出对目标任务有价值的模型。Wu 等人 [85] 人研究了如何在不访问模型池中模型的原始训练数据的情况下从模型池中选择满足当前任务的模型。基于此，Tan 等人 [78] 进行了进一步的研究，特别是考虑到各种预训练模型通常是从现实场景中的不同特征空间获得的。这是使多路复用模型能够处理异构特征空间的首次尝试。

Ding 等人 [19] 研究了迁移学习领域开源预训练模型的可重用性评估，提出了适用于一般学习任务的协同学习算法和模型。

相应地，近年来也提出了一些模型重用方法。Wu 等人 [84] 提出了一种称为 HMR 的通用异构模型重用，用于有限通信成本（即有限的标记训练样本）下的多方学习。Zhao 等人 [98] 提出了一种通过模型重用来处理概念漂移的方法。Shao 等人 [70] 提出了使用多潜在域方法的模型重用，以解决来自目标任务中潜在未知域的数据问题。Pan 等人 [61] 提出了模型分解的方法，提高了模型开发的可重用性、可替换性和模块化程度。Qi 等人 [66]。通过卷积神经网络的模块化和组合来增强神经网络。

同时，模型复用也会带来一些问题。Ji 等人 [34] 研究了模型重用带来的安全问题，并提出了一种广泛的模型重用攻击方法。分析表明，复用模型前所未有的复杂性是安全问题产生的原因，并给出了潜在的对策和相应的挑战。模型复用还可能导致漏洞传播，因此 Li 等人 [41] 提出了模型复用可能导致漏洞传播的问题并提出了一种基于测试的深度学习模型相似度比较方法，称为 ModelDiff，用于模型重用检测。减轻迁移学习中模型重用导致的漏洞或后门的继承。Zhang 等人 [96] 提出了模型切片技术来继承更多相关权重以减少缺陷继承。Qi [67] 等人提出了模型重组的概念，设计了提高模型复用性的工具，以缓解模型复用带来的缺陷继承。

2.1.3 知识融合

知识融合的主要目的是整合教师模型中包含的暗知识，从而获得多功能的学生模型。You 等人 [91] 提出了一种从多个教师模型中提取知识来训练学生模型的方法。在进行知识融合时，不仅考虑实际教师模型的输出，还考虑模型中间层不同样本的表示。Shen 等人 [71] 提出通过知识融合来完成综合分类任务，但限制了教师模型的同质性。在进一步的研究中，Shen 等人 [72] 提出通过异构教师模型的知识融合来获得学生模型。Ye 等人 [90] 通过融合来自教师模型的过滤知识，使学生模型掌握教师模型的全部或子集经验。Ye 等人 [89] 个融合了两个不同侧重点的教师模型的知识，用于训练多功能学生模型。Liu 等人 [44] 提出了从教师模型到学生模型的知识流，性能超过微调。Jing 等人 [36] 提出了一种图神经网络中知识融合的方法。Yu 等人 [92] 通过知识融合使神经检索模型对新旧数据更加准确。Xu 等人 [87] 提出了双重判别特征对齐（DDFA）框架来改善类外类别的知识融合。Carvalho 等人 [18] 通过知识融合缓解了神经网络的灾难性遗忘问题。

2.1.4 知识蒸馏

知识蒸馏与知识融合类似，但它侧重于通过蒸馏过程将教师模型的知识转移到学生模型。近年来模型蒸馏的工作大致可以分为以下几个方面。多模型蒸馏 [63, 97]，缓解教师模型和学生模型在结构和数据层面的匹配问题 [24, 55, 100] 等。回顾文献 [30, 83] 并提出一些新的知识蒸馏方法 [14, 39].

2.2 深度学习模型测试

目前深度学习模型测试旨在通过实施不同的深度学习测试技术来检测模型的质量。Pei 等 [64] 提出了一种名为 DeepXlpore 的白盒测试技术，是基于神经元覆盖的测试。揭示了自动驾驶学习系统中数以千计的错误转弯行为。Moosavi-Dezfooli 等 [57] 提出名为 DeepFool 的测试鲁棒性技术，通过添加愚弄深度学习模型的噪音去评估该模型的鲁棒性。Carlini and Wagner 等 [12] 使用距离度量量化相似性的来生成对抗样本。Papernot 等 [29, 62] 设计了一个库来标准化来生成对抗样本。Zhang 等 [95] 通过实证研究发现发现 175 个模型错误中有 9 个（5.1%）属于效率问题。这个比例不高。原因可能是效率问题很少发生，或者这些问题很难检测。AEQUITAS [68] 是一种公平性测试技术，它的目标是在数据集中发现存在偏见的实例，并据此生成更多的测试用例来帮助模型进行再训练。Doshi-Velez 等 [22] 给出了可解释性评估（测试）方法的分类：基于应用程序、基于人和基于功能。Ding 等 [21] 将程序视为灰色框，并通过统计测试来检测不同的隐私侵犯行为。Ahuja 等 [5] 对多种深度学习模型测试技术进行了比较。

传统的变异测试技术也被拓展用于深度学习测试。Shen 等人最先提出了 MuNN 技术 [75]，首次将变异测试技术应用于不同类型的深度学习模型，并提出了五种变异算子。Ma 等在 DeepMutation 方法 [49] 中提出了更丰富的变异算子，包括模型级和源码级，并通过实验证明了变异测试技术在深度学习软件中的有效性。Hu 等人 [32] 推出了一个升级的变异测试工具 DeepMutation++，它不仅支持正常的深度神经网络，还支持有状态的循环神经网络（RNN）。Hildebrandt 等人认为目前的变异算子无法模拟实际故障的影响，他们通过对深度学习软件实际故障的实证研究，在工具 DeepCrime [33] 中实现了 24 个深度学习变异算子，这是基于真实错误的源代码级预训练变异工具。他们的实验也显示 DeepCrime 对测试数据质量变化的敏感性优于 DeepMutation++。

3 问题描述

本文对深度学习模型融合与测试技术展开研究，重点关注深度学习模型融合和后门样本检测两个问题，并对两个问题给予了解决方案。下文将对本文研究的问题进行描述。

3.1 模型融合

由第一章所述，我们的研究方向是，在图像分类任务中我们将两个分类功能具有重叠的模型进行融合，来增强神经网络模型，使融合模型可以满足全分类任务需求。在此我们对此研究问题进行具体描述。

如下图 2所示，给定两个分类任务，分别记为 Task $\mathcal{T}_\mathcal{A}$ 和 Task $\mathcal{T}_\mathcal{B}$，其中 $\mathcal{T}_\mathcal{A}$ 的类别空间为 $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ 记为 $\mathcal{C}_\mathcal{A}$ ，$\mathcal{T}_\mathcal{B}$ 的类别空间为 $\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6$ 记为 $\mathcal{C}_\mathcal{B}$ ，并且它们是重叠的，即 $\mathcal{C}_\mathcal{A} \cap \mathcal{C}_\mathcal{B} = \mathcal{C}_3, \mathcal{C}_4$。$\mathcal{M}_\mathcal{A}$ 是在 $\mathcal{T}_\mathcal{A}$ 上训练的模型，$\mathcal{M}_\mathcal{B}$ 是在 Task $\mathcal{T}_\mathcal{B}$. 上训练的模型。通过对 $\mathcal{M}_\mathcal{A}$ 和 $\mathcal{M}_\mathcal{B}$ 进行融合生成 $\mathcal{M}_\mathcal{C}$ 可以满足新的分类任务 Task $\mathcal{T}_\mathcal{C}$ ，其应用上下文有大量未标记数据，其类别空间 $\mathcal{C}_\mathcal{C} = \mathcal{C}_\mathcal{A} \cup \mathcal{C}_\mathcal{B}$。

模型	类别空间						任务
	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	
$\mathcal{M}_\mathcal{A}$	✓	✓	✓	✓	×	×	$\mathcal{T}_\mathcal{A}$
$\mathcal{M}_\mathcal{B}$	×	×	✓	✓	✓	✓	$\mathcal{T}_\mathcal{B}$
$\mathcal{M}_\mathcal{C}$	✓	✓	✓	✓	✓	✓	$\mathcal{T}_\mathcal{C}$

图 2: 模型融合的例子。✓和 × 代表了是否包含分类。

由于数据隐私的保护或其他一些场景限制，我们假设无法获取 $\mathcal{T}_\mathcal{A}$ 和 $\mathcal{T}_\mathcal{B}$ 的训练数据集。只能得到训练好的模型，即 $\mathcal{M}_\mathcal{A}$ 和 $\mathcal{M}_\mathcal{B}$。另外由于人工标注的成本，只能标记少量数据。因此，如何在有限的标记数据下重用 $\mathcal{M}_\mathcal{A}$ 和 $\mathcal{M}_\mathcal{B}$ 更好地解决 $\mathcal{T}_\mathcal{C}$ 的分类任务是我们研究的主要问题。

很多应用场景都满足上述假设。以医院对一组特定疾病的分类为例。$\mathcal{C}$ 医院，拥有大量未标记的疾病图像，迫切需要对这些图像进行分类。这些图像类别空间被记作 $\mathcal{C}_\mathcal{C}$。另外还有两家医院：医院 $\mathcal{A}$ 建立了 $\mathcal{M}_\mathcal{A}$，可以用类别空间 $\mathcal{C}_\mathcal{A}$ 对分类法的部分进行分类，医院 $\mathcal{B}$ 有 $\mathcal{M}_\mathcal{B}$，可以对疾病的分类法和类别空间的部分进行分类 $\mathcal{C}_\mathcal{B}$。出于保护患者隐私的考虑，$\mathcal{C}$ 医院只能获取 $\mathcal{A}$ 医院和 $\mathcal{B}$ 医院的模型，而无法获取相应的训练数据。而且，由于标注疾病图片既需要专业医生，也需要病人的信息甚至是身体代价，故标注成本较高。因此，在标记样本有限的情况下，如何复用 $\mathcal{M}_\mathcal{A}$ 和 $\mathcal{M}_\mathcal{B}$ 来满足医院 $\mathcal{C}$ 任务是一个非常现实的需求。

3.2 后门模型木马输入检测

由第一章所述，深度学习测试技术普遍用于揭露模型的问题。可以利用测试技术来对被植入后门的模型进行木马样本检测。在此我们给出三个定义来说明要解决的问题。

Definition 1 (深度学习模型). *深度神经网络结构实现的深度学习模型可以被看做一个复杂分类器。该分类器实现了特征域 $\mathcal{F}_d$ 到分类域 $\mathcal{C}_d$ 的映射，$\mathcal{M}: \mathcal{F}_d \rightarrow \mathcal{C}_d$，当一个入 $x \in \mathcal{F}_d$ 是特征向量，那么对应的输出 $y \in \mathcal{C}_d$，其中 $\mathcal{C}_d = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_m\}$。*

Definition 2 (后门攻击和目标类). *后门攻击通过将隐藏的后门植入到原始深度学习模型 $\mathcal{M}_\mathcal{O}$ 的模型结构中来感染模型。判断后门攻击成功与否，要看是否满足下面两点：* ***(a)*** *在干净（正常样本）x^c 输入下，被植入的深度学习模型 $\mathcal{M}_\mathcal{I}$ 行为不改变，即，$\mathcal{M}_\mathcal{I}(x^c) = \mathcal{M}_\mathcal{O}(x^c)$；* ***(b)*** *当后门被攻击触发器激活后，$\mathcal{M}_\mathcal{I}$ 预测木马输入 x^t 为攻击指定的分类目标类别 $C_\mathcal{T}$，即，$\mathcal{M}_\mathcal{I}(x^t) = C_\mathcal{T}$。*

Definition 3 (木马样本检测). *给定一个被感染深度学习模型 $\mathcal{M}_\mathcal{I}$ 和一个包含干净和恶意样本输入的训练集合 $\{x_1, x_2, \ldots, x_n\}$，假定当前模型不知道被植入后门的攻击和目标类信息。我们要研究的问题如下：对于每一个训练样本 x_i，可以检测出 x_i 是否是木马样本。即，$TID(x_i) \in \{clean, trojaned\}$。*

我们将采用软件测试的思路去研究木马输入检测的问题，具体为采用深度学习变异测试

表 1: The datasets and models used in this experiment.

领域	数据集	模型	类别数	重叠数	性能		数据量	
					训练集准确率	测试集准确率	训练集数据量	测试集数据量
Car	Car Body Style Dataset [16]	Xception	7	5	0.964	0.8125	2670	672
	Car Body Style [23]	Xception	8		0.9675	0.815	1599	400
Flower	Flower Classification | 10 Classes | [81]	ResNet50V2	5	3	0.983	0.948	6765	1693
	Flowers Recognition [6]	DenseNet201	5		0.9565	0.9272	3452	865
Food	food-11 Image Classification Dataset [10]	EfficientNetB3	11	2	0.981	0.9155	9900	1100
	Food Recognition - Burger, Pizza & Coke [50]	VGG19	3		0.9465	0.9009	4320	1080
Fruit	10 fruit [3]	VGG19	10	3	0.9714	0.8538	664	171
	Vegetables & Fruits fresh and Stale [7]	Xception	9		0.9936	0.9589	7472	4040
Sport	100 Sports Image Classification [28]	EfficientNetB3	100	13	0.989	0.9442	11615	2957
	Sports Image Dataset [38]	VGG16	22		0.9465	0.9453	11436	2870
Weather	Weather Classification [76]	VGG19	5	3	0.9235	0.8202	3697	929
	Weather Image Recognition [9]	ResNet152V2	11		0.9737	0.8788	5484	1378
Animal_1	African Wildlife [26]	VGG19	4	2	0.9533	0.8322	1200	304
	Danger Of Extinction / Animal Image Set [2]	Xception	11		0.9732	0.8916	5183	1301
Animal_2	Animal Image Dataset(DOG, CAT and PANDA) [69]	ResNet50	3	2	0.9708	0.9133	2400	600
	Animal -5 Mammal [4]	EfficientNetB0	5		0.9877	0.9383	11996	3000
Animal_3	wild_cats [1]	EfficientNetB3	5	2	0.944	0.8347	2463	617
	Wildlife Animals Images [51]	EfficientNetB3	6		0.9985	1.0000	1377	346

技术来生成后门模型的变异体，通关观察变异体在木马输入和干净输入的差异来识别木马输入。

4 思路及解决方案

本章中将对第三章提出的两个问题提供思路及解决方案。

4.1 基于级联并行的模型融合技术

数据集：我们从 Kaggle 平台收集了 9 对 18 个数据集进行实证实验。我们收集的数据集具有以下 3 个特征。首先，我们关注图像分类数据集。其次，分类任务的领域广泛，涵盖不同的应用领域，例如汽车、花卉、水果等。每对数据集对应一个领域，其中包含 $\mathcal{T}_{\mathcal{A}}$ 和 $\mathcal{T}_{\mathcal{B}}$。最后，类别重叠。例如，在汽车领域，$\mathcal{T}_{\mathcal{A}}$ 有 7 个类别，$\mathcal{T}_{\mathcal{B}}$ 有 8 个类别，并且有 5 个类别重叠。

数据集的具体信息如表 1所示。表 1中的“领域”列表示数据集的领域。“数据集”列表示 $\mathcal{T}_{\mathcal{A}(\mathcal{B})}$ 的名称。“类别数”列表示数据集的分类数。“重叠数”列表示重叠类别的数量。两个数据集都分为训练集和测试集。训练集用于训练教师模型，测试集混合得到混合测试集，用于学生模型的训练和评估。具体性能和数量量分别见表 1“性能”和“数据量”列。

教师模型：每个领域下，分别使用数据集 $\mathcal{A}$ 和数据集 $\mathcal{B}$ 的训练集来训练 $\mathcal{M}_{\mathcal{A}}$ 和 $\mathcal{M}_{\mathcal{B}}$。使用了 5 种常见的卷积神经网络，如 ResNet、Xcepiton、VGG19 等。如表 1所示，“模型”列列出了数据集对应的模型结构。

学生模型生成：融合教师模型生成的学生模型记为 $\mathcal{M}_{\mathcal{C}}$。直观上，$\mathcal{M}_{\mathcal{C}}$ 可以融合教师模型

（$\mathcal{M}_{\mathcal{A}}$ 和 $\mathcal{M}_{\mathcal{B}}$）的知识，从而在混合测试集上有更好的性能。）

学生模型的生成对于我们的方法至关重要。我们提出了基于级联并行的模型融合方法，称为 MCCP（Model Combination method based on Concatenation-Parallel）。我们的方法结合了以下两点技术思想。

- 固定模型重用（FMR）[88]：FMR 的主要目的是利用预训练模型的固定特征来训练新模型，使新模型能够获得预训练模型的内部知识，将模型的隐藏层充分连接起来训练和预训练模型的固定特征与分类输出层。我们采用类似的连接策略来融合两个教师模型，即完全连接两个教师模型的某些层来构建联合模型。
- 迁移学习（TL）[77]：我们的任务场景与 TL 类似，迁移在新旧分类任务上训练的模型。在 TL 中，如果新任务的训练数据有限，它会冻结预训练模型中某些层的权重，以便只训练其他分类层的权重。考虑到人工标记的成本较高（即只能标记一小部分测试样本），我们将权重冻结在“合并”层之前，即仅训练“合并”层和分类层之间的权重。。

详细介绍如图 3所示，图中第一列的 $\mathcal{M}_{\mathcal{A}(\mathcal{B})}$ 分别代表两个教师模型。其中，$\mathcal{M}_{\mathcal{A}(\mathcal{B})}$ 的输出类别为 y_1、y_2 和 y_3，$\mathcal{M}_{\mathcal{B}}$ 的输出类别为 y_3、y_4 和 y_5，其中 y_3 是重叠分类类别。MCCP 可以分为三步。第一步：连接教师模型的最后隐藏层。具体来说，我们分别切断 $\mathcal{M}_{\mathcal{A}(\mathcal{B})}$ 的输出层，然后连接它们的最后一个隐藏层以形成组合特征。第二步：输出层权值的并行化，其中 $\mathcal{W}_{\mathcal{A}(\mathcal{B})}$ 分别代表 $\mathcal{M}_{\mathcal{A}(\mathcal{B})}$ 的输出层权值。输出层神经元的数量由所有类别的数量决定，输出层连接到连接隐藏层，从而形成学生模型结构。然后新模型继承了教师模型的神经元权重，即 $\mathcal{W}_{\mathcal{A}(\mathcal{B})}$。图 3中的虚线表示连接权重初始化为 0。至此，学生模型（$\mathcal{M}_{\mathcal{C}}$）的结构和初始权重就完成了。第三步：$\mathcal{M}_{\mathcal{C}}$ 训练，我们只训练最终输出层的权重，也就是图中的黑色连接线。

形式上，设 $\mathcal{L}_{\mathcal{A}(\mathcal{B})}$ 分别为 $\mathcal{M}_{\mathcal{A}(\mathcal{B})}$ 的最后一个隐藏层，$\mathcal{M}_{\mathcal{C}}$ 的最后一个隐藏层标记为 $\mathcal{L}_{\mathcal{S}} = concate(\mathcal{L}_{\mathcal{A}}, \mathcal{L}_{\mathcal{B}})$。$\mathcal{M}_{\mathcal{A}(\mathcal{B})}$ 的类别数为 $\mathcal{N}_{\mathcal{A}(\mathcal{B})}$，重叠类别数为 $\mathcal{N}_{\mathcal{O}}$，因此 $\mathcal{M}_{\mathcal{C}}$ 的输出层神经元数记为 $\mathcal{N}_{\mathcal{S}} = \mathcal{N}_{\mathcal{A}} + \mathcal{N}_{\mathcal{B}} - \mathcal{N}_{\mathcal{O}}$。具体权重继承规则如下。

$\mathcal{M}_{\mathcal{C}}$ 输出层的神经元索引记为 i。神经元 i 的权重表示为 $\mathcal{W}_i$。$\mathcal{M}_{\mathcal{C}}$ 输出层神经元索引集可分为三部分：$\mathcal{A}$ 的唯一神经元索引集 $unique_{\mathcal{A}}$,$\mathcal{B}$ 的唯一神经元索引集 $unique_{\mathcal{B}}$ 和重叠神经元索引集 $overlap$。注 $\mathcal{M}_{\mathcal{A}(\mathcal{B})}$ 输出层神经元索引分别为 $i_{\mathcal{A}(\mathcal{B})}$ 根据类别的字母顺序，索引 i 与索引 $i_{\mathcal{A}}$ 和 $i_{\mathcal{B}}$ 具有映射关。具体地，当 $i \in unique_{\mathcal{A}}$, 时，映射关系 $i \rightarrow i_{\mathcal{A}}$ 表示全局分类标签索引 i 对应 $\mathcal{M}_{\mathcal{A}}$ 的局部分类标签索引 $i_{\mathcal{A}}$。因此，总结出权重继承公式 1，其中 $concat$ 的作用就是将两个向量连接起来

简而言之，MCCP 就是通过连接教师模型最后的隐藏层，并通过继承教师输出层神经元的权重来继承教师模型的知识，从而完成特征的串联。因此，学生模型有两个输入层，它们由两个教师模型的输入层组成。注意学生模型的两个输入层数据都经过预处理。

训练学生模型：学生模型（即 $\mathcal{M}_{\mathcal{C}}$）的训练分为两个步骤。第一步是通过采样生成带标记的训练样本，第二步是训练学生模型 $\mathcal{M}_{\mathcal{C}}$ 并在混合测试集上对其进行评估。

第一步（采样）：我们通过从混合测试集中采样来生成训练样本。采样比例设置为 1%，3%，5%，10%，15% 和 20% [74]。抽样策略为随机抽样。为了消除训练样本带来的随机性，我们对每个采样比例重复实验 5 次。

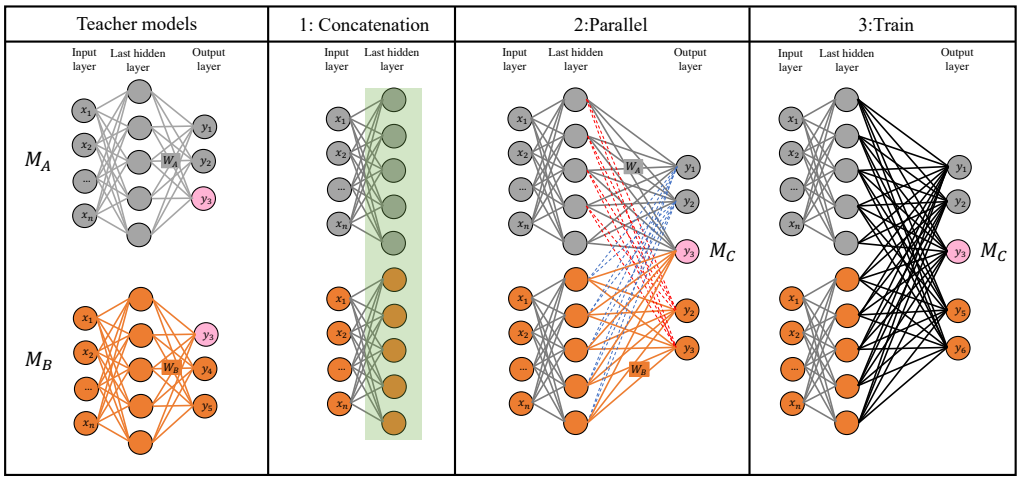


图 3: 我们的 MCCP 方法的流程图。$\mathcal{W}_\mathcal{A}$ 和 $\mathcal{W}_\mathcal{B}$ 分别表示最后一层 $\mathcal{M}_\mathcal{A}$ 和 $\mathcal{M}_\mathcal{B}$ 的权重。绿色阴影层代表最后一个隐藏层的组合。虚线的权重值为 0。

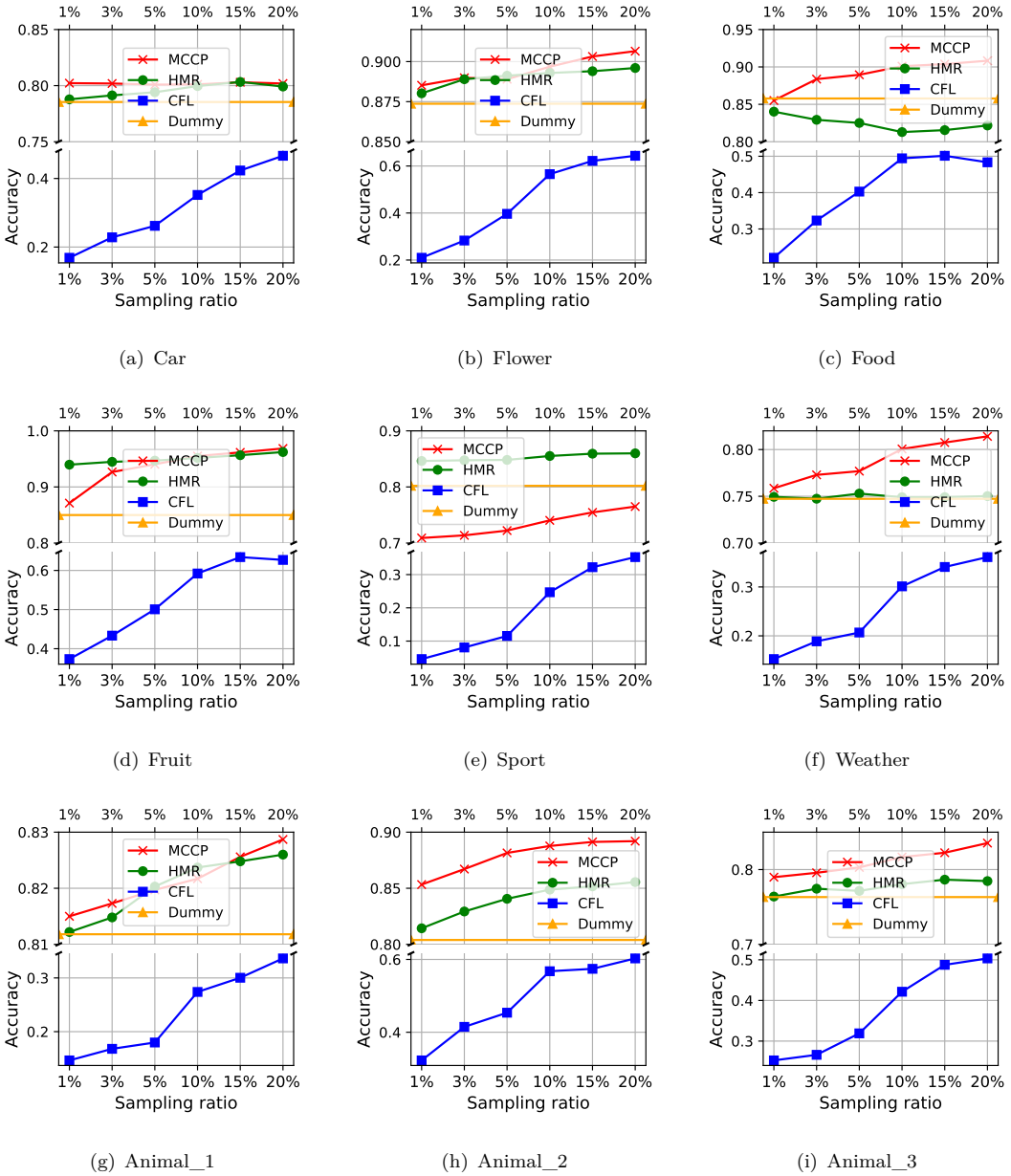
第二步（训练）：对于 MCCP，将训练样本输入 $\mathcal{M}_\mathcal{C}$ 并训练 5 个 epoch。请注意，由于 $\mathcal{M}_\mathcal{C}$ 有两个输入层，因此有必要对每个输入层进行相应的数据预处理。同时，在训练过程中，MCCP 冻结了教师模型组合层之前的所有权重，即只训练 $\mathcal{M}_\mathcal{C}$ 的组合层和分类层之间的权重。在基线方法中，对于 HMR，训练样本分别输入 $\mathcal{M}_\mathcal{A}$ 和 $\mathcal{M}_\mathcal{B}$ ，并添加虚拟保留类进行再训练，训练 5 个 epoch。对于 CFL，将训练样本分别输入 $\mathcal{M}_\mathcal{A}$、$\mathcal{M}_\mathcal{B}$ 和 $\mathcal{M}_\mathcal{C}$，并将 $\mathcal{M}_\mathcal{C}$ 的输出与 $\mathcal{M}_\mathcal{A}$ 和 $\mathcal{M}_\mathcal{B}$ 的输出进行拟合，训练 30 个 epoch。对于 Dummy 来说，没有训练过程。最后，对于所有模型组合方法，在混合测试集上进行评估。

$$\mathcal{W}_i = \begin{cases} concat(\mathcal{W}_{i_\mathcal{A}}, 0) & i \in unique_\mathcal{A}\,; \\ concat(0, \mathcal{W}_{i_\mathcal{B}}) & i \in unique_\mathcal{B}\,; \\ concat(\mathcal{W}_{i_\mathcal{A}}, \mathcal{W}_{i_\mathcal{B}}) & i \in overlap\,. \end{cases} \tag{1}$$

实验方法：我们引入了三种方法作为基线，包括 ICML'2019 [84] 的 HMR、IJCAI'2019 [48] 的 CFL）和一个通用基线（称为 Dummy）。选择 HMR 和 CFL 的原因是这两种方法可以很好地迁移到我们的研究场景中。即教师模型的训练集不可访问，教师模型可以是同构的，也可以是异构的。Dummy 是最直观的基线，它通过整合教师模型的输出来直接进行预测。

我们的实验可以分为两个步骤。第一步是从混合测试集中采样以进行模型训练。采用随机抽样策略进行抽样，抽样比例为 1%、3%、5%、10%、15%、20% [74]。第二步是训练学生模型，训练轮数为 5。在混合测试集上评估学生模型的分类精度。

实验结果：实验结果如图 4所示。每个子图对应一个领域，横轴表示采样比例，纵轴表示学生模型在混合测试集上的分类精度。Dummy 方法是一条横线，因为 Dummy 方法没有训练过程，只使用教师模型置信度最高的标签作为预测标签。从图中可以看出，随着训练样本数量



(a) Car (b) Flower (c) Food

(d) Fruit (e) Sport (f) Weather

(g) Animal_1 (h) Animal_2 (i) Animal_3

图 4: 训练后学生模型的准确性。横轴表示用于训练的手动标注的采样比例，纵轴表示学生模型在混合测试集中的分类准确率。

的增加，模型的性能将会提高。其中 CFL 的性能提升最快，这是由于 CFL 在训练前没有结合教师模型，而是使用新的未经训练的模型作为学生模型, 即 CFL 方法在初始阶段并没有融合教师模型的知识（初始精度较低）。只是在后续的训练过程中，通过知识蒸馏的方法将教师模型的判别能力蒸馏到学生模型中。综合来看，可以看出，在大多数情况下，9 个领域中的 8 个，MCCP 的性能都优于其他方法，CFL 的准确率最低，但其性能提升最快。

实验发现 MCCP 在训练后在混合测试集上表现更好，在大多数情况下超过了基线。（已

投稿至 JSME 期刊并获得审稿人的 major revision 意见）

4.2 基于熵的木马输入检测技术

随着将传统软件测试中的变异测试 [35] 引入到深度学习软件 [49, 75] 之后，深度学习变异测试已广泛应用于许多新的场景中。变异测试的核心技术是变异算子。Ma 等 [49] 提出的 DeepMutation 涵盖了 16 种（8 种代码级和 8 种模型级）深度学习变异算子。在后门攻击场景下，代码级的变异操作是不适用的，因为我们仅仅只能访问被植入后门的模型，没有训练集和训练代码。Wang 等 [82] 应用深度学习变异测试去检测对抗样本，他们的核心观察是对抗样本往往更接近决策边界，这使得利用变异技术来检测对抗样本成为可能。

与 Wang [82] 等提出的视角相反，我们假设木马样本对模型的决策边界并不敏感。我们提出了基于熵的木马输入检测 ETID（Entropy-Based Trojaned InputDetection）该方法大体上分成两部分，第一步，确定出哪一个分类是攻击的目标分类；第二步，根据模型的预测信息的熵来确定某一个输入样本是不是木马样本。我们的方法结合了两种思想：

- 决策边界 [73]：通过变异操作对被植入后门的模型进行变异，改变模型的决策边界，通过统计各个分类子集在变异模型上的精度精度，根据我们的假设，精度分布越高则越有可能是攻击目标类别。
- 预测信息熵 [27]：在确定攻击类别后，我们根据一个输入在不同变异模型上输出的结果，计算该样本的预测熵。因为有很高的概率变异模型对后门样本的预测功能不变，但是对普通干净样本而言，变异模型对其预测标记会发生改变。而且由于我们的变异操作是随机不可控的，所以不同的变异输出类别很可能是不同的。也就是干净样本的熵会比较大，同时后门样本的熵会比较小甚至接近于 0，因为针对某个后门样本每一个变异模型的输出类别都是一样的。因此根据信息熵就可以检测出某个输入样本是否是木马输入样本。

图 5展示了一个被注入后门的多分类模型 $\mathcal{M}_{\mathcal{I}}$ 的例子。具体来看该模型是一个 3 分类模型，类别 1: ★, 类别 2: ♦, 和类别 3 ▲。这些样本被原始模型 $\mathcal{M}_{\mathcal{O}}$ 自带的原始边界（实线）分成了三类。不带框的形状（★，♦，▲）代表普通干净样本，颜色（红，绿，蓝）代表被 $\mathcal{M}_{\mathcal{I}}$ 预测的结果，框子表明木马样本。正如图 5所看到的干净的样本都被 $\mathcal{M}_{\mathcal{I}}$ 正确分类了，但是被攻击者注入后门的带框样本都被 $\mathcal{M}_{\mathcal{I}}$ 分类成了类别 2。综上所述，我们获得了以下观察发现：

1）木马样本与原始的模型的决策边界没有明显的关系，即这些样本与普通干净的样本相比并没有更接近决策边界。

2）深度神经网络模型被变异后，其决策边界发生改变。木马样本没有很高的概率跨越改变后的决策边界，因此这些样本的预测结果可能保持不变。

3）攻击者设计的后门触发是隐蔽的（很难去被察觉）同时也很难去消除。

方法介绍：

生成变异模型：变异算子是变异测试的核心技术，用于变异原始深度学习模型。最近，Shen 等人提出了一种称为 MuNN [75] 的深度学习变异方法，由 5 个模型级变异算子组成。之后，

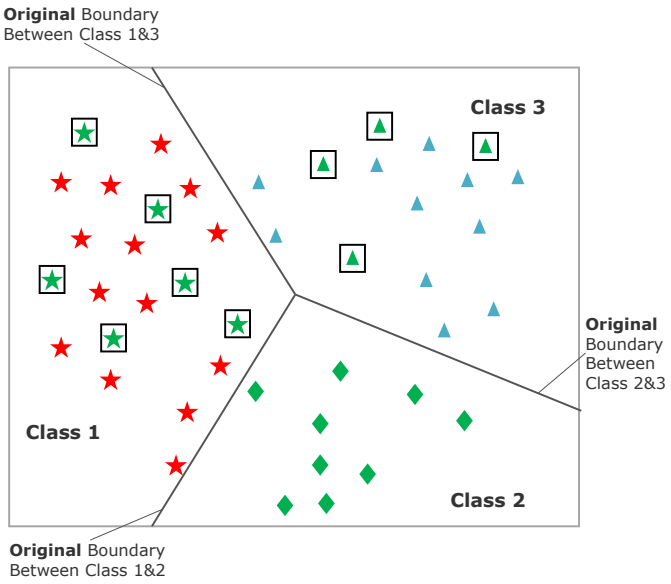


图 5: 一个被植入后门的多分类深度学习模型 $\mathcal{M}_{\mathcal{I}}$

Ma 等人提出了一种更全面的深度学习变异技术，称为 DeepMutation [49]，它包含 16 个（即 8 个源级和 8 个模型级）深度学习变异算子。在后门攻击场景中，源级变异算子不适合，因为我们只能获取后门注入的深度学习模型，而不能获取训练数据或训练代码。具体来说，在本研究中，我们在 DeepMutation [49] 中采用了 5 个模型级深度学习特定变异算子如下表 2所示，表中第一列表示变异算子的名称，表中第二列表示算子作用域，表中第三列描述变异算子的作用。它们在语法上与 MuNN [75] 中的变异算子相似。没有考虑算子“Layer Deactivation(M)”和“Layer Addition(M)”，因为我们研究的深度学习模型不满足结构条件。此外，我们进一步删除了算子“Activation Function Removal(M)”，因为该算子只能生成很少数量的变异体（与深度学习模型中的层数一致），这不适合我们的工作。

第一步：确定攻击分类：

我们在实验中观察到一个现象：注入后门的深度学习模型的变异体对于目标（受攻击）类别的预测非常稳健，可用于在检测中首先识别目标类别。我们使用 3 个图片分类数据集，分别是 MNIST、CIFAR10 和 GTSRB，其大小分别为 10,000、10,000 和 12,630。我们使用 6 个先进的后门攻击方法，分别是 BadNets [31],Blended [15],IAD [59],LabelConsistent [80],Refool [46],WaNet [58]。每个攻击我们只设置数据集中的一个类别作为目标攻击类，且攻击比例设计为 10%。

首先，我们计算了每个变异体在每个类别的原始训练集 $\mathcal{T}$ 上的准确性。例如，对于类别 $\mathcal{C}$，我们首先选择 $\mathcal{T}$ 中真实标签为 $\mathcal{C}$ 的所有测试样本组成子集 $\mathcal{T}_{\mathcal{C}}$。$\mathcal{C}$ 类在变异模型、$\mathcal{M}_m$ 上的准确率为 $\mathcal{ACC}(\mathcal{M}_{\mathcal{C}}, \mathcal{T}_{\mathcal{C}})$。变异模型的变异率从 1%,3%,5%,10%,20%,30%,40%,50%,60%,70%,80% 逐渐增加，直到精度的平均值和中位值均为最大的分类上的精度分布与其他类有显著性差异，则停止变异，确定当前变异率为自适应的变异率，该类别为预测的攻击目标类别。如前所述，我们总共使用了 5 个变异算子，详情见表 2。显然，对于每个数据集和攻击场景，我们可以在

表 2: 采用的 5 种模型级变异算子

变异算子	级别	详情
Gaussian Fuzzing	Weight	Fuzz weight by Gaussian Distribution
Weight Shuffling	Neuron	Shuffle weights of selected neurons
Neuron Effect Blocking	Neuron	Block selected neurons
Neuron Activation Inverse	Neuron	Reverse the activation of selected neurons
Neuron Switch	Neuron	Switch two selected neurons in one layer

每个输出类别中获得 250 个变异精度值

第二步：检测后门样本：

在检测后门样本阶段我们分为 2 步进行，我们先对生成的变异模型（共有 250 个）进行排序，选择出合适的变异模型来用于后门样本检测。然后我们在用被选择的变异模型计算数据集样本的预测熵，然后基于熵来检测后门样本。下面我们对这 2 步分别进行介绍。

（1）变异模型排序。首先，人工从攻击类别中选择少量干净样本。计算变异模型在干净样本和剩余样本的精度差。差值越小的变异模型优先级越高，以此对变异模型进行排序。

例如，如果在第一步中确定攻击目标类为类 1，我们就人工从类别为 1 的样本中，选择少量的干净样本作为干净种子集合记为 $Seed_{clean}$，则类 1 中剩余的样本集合记为 $RemainSet$。然后计算变异模型 $\mathcal{M}_m$ 在 $Seed_{clean}$ 上的准确度，记为 $\mathcal{ACC}(\mathcal{M}_m, Seed_{clean})$，计算变异模型 $\mathcal{M}_m$ 在 $RemainSet$ 上的准确度，记为 $\mathcal{ACC}(\mathcal{M}_m, RemainSet)$。表示变异模型 $\mathcal{M}_m$ 的优先级值可用 $\mathcal{ACC}(\mathcal{M}_m, Seed_{clean}) - \mathcal{ACC}(\mathcal{M}_m, RemainSet)$ 表示，值越小则优先级越高，以此优先级生成变异模型的优先级队列。

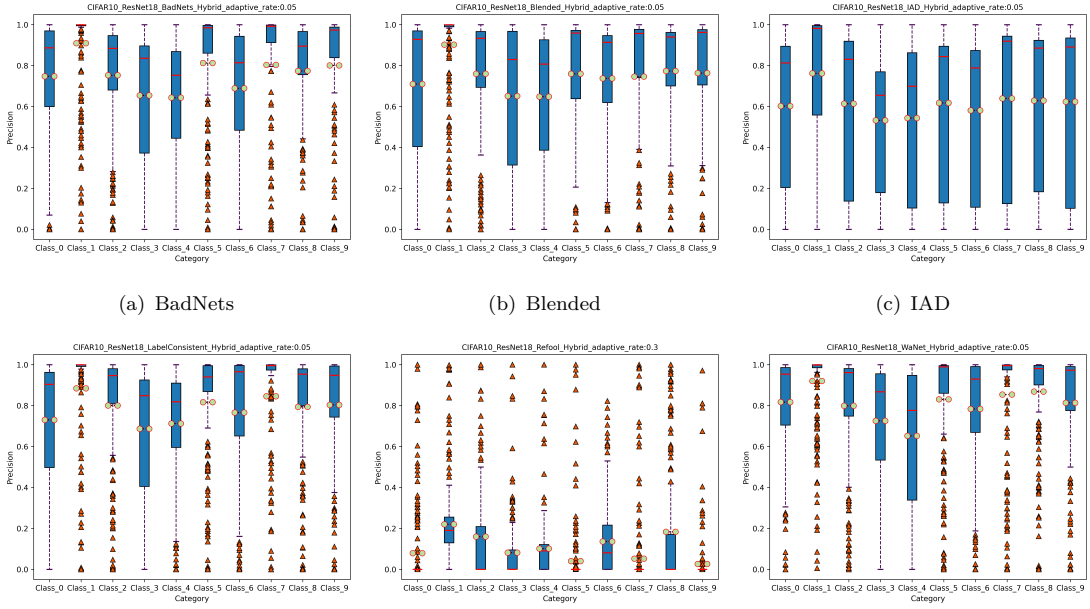
（2）基于熵的后门检测。从变异模型的优先级队列中选择前 50 的变异模型，在攻击目标类中计算样本的预测分类结果的熵。熵越大则该样本越可能为后门样本。

具体来说，紧接上一步，我们从变异模型优先级队列中取出前 50 个变异模型作为我们变异模型组。将攻击目标类（如类别 1）中的每个样本都喂给每个变异模型，即获得每个待测样的 50 个预测标签列表，进而计算每个预测标签列表的信息熵。根据我们的见解，干净样本对模型的变异更为敏感，故熵越大则该样本越可能为后门样本。将熵值作为优先级，我们可以获得待测样本的优先级队列（即越靠近队头越有可能为后门样本）。

具体的实现算法如算法 1所示。第 1 行，我们先预先定义从小到大的变异率。第 2 行至第 10 行，完成确定攻击目标类，即变异率从小到大对后门模型进行变异，当可以确定出攻击目标类时终止，不再进行变异否则继续变异。第 12 行至第 19 行，利用种子样本对变异模型

进行排序。第 21 行至 24 行，计算待测样本熵，并生成样本的优先级队列。

我们在 3 个图像数据集（MINST、CIFAR-10、GTSRB）和 6 个后门攻击 (BadNets [31],Blended [15],IAD [59],LabelConsistent [80],Refool [46],WaNet [58]。) 上进行了实验。其中在 CIFAR-10 数据集,ResNet18 模型的第一步（确定目标攻击类）和第二步（木马样本检测）实验结果分别如图 6和图 7所示。从结果上来看，在第一步中，可以明显看出攻击目标类（Class 1）在变异模型上的精度分布高于其他类，具有明显差异 (Wilcoxon 有符号秩检验 p 值小于 0.05)。在第二步中，如图 7所示，每个子图对应一个后门攻击，横轴为待测样本优先级队列的前截取比例。例如横轴 0.1 表示，将队列的前 10% 样本为检测为木马样本。纵轴为检测性能指标分别计算了精度 (precision) 和召回率 (recall)。从结果上来看，对于每个后门攻击，检测木马样本的效果都非常好。



(a) BadNets (b) Blended (c) IAD

(d) LabelConsistent (e) Refool (f) WaNet

图 6: 确定攻击目标类

5 之前工作总结和未来工作展望

目前我们的工作专注于深度学习模型融合与测试技术研究，解决了两个相关的研究问题: 1）模型融合; 2）木马输入检测。对于模型融合研究，考虑了一种新的应用场景。如何复用功能重叠的模型，在受限的标记代价下如何能够提高模型的分类性能，提出了一种新的基于级联并行的模型合并技术，通过复用已有模型，来提高开发效率。对于深度学习模型的测试技术研究，我们利用变异测试技术，提出一种新的测试方法用于检测木马输入。

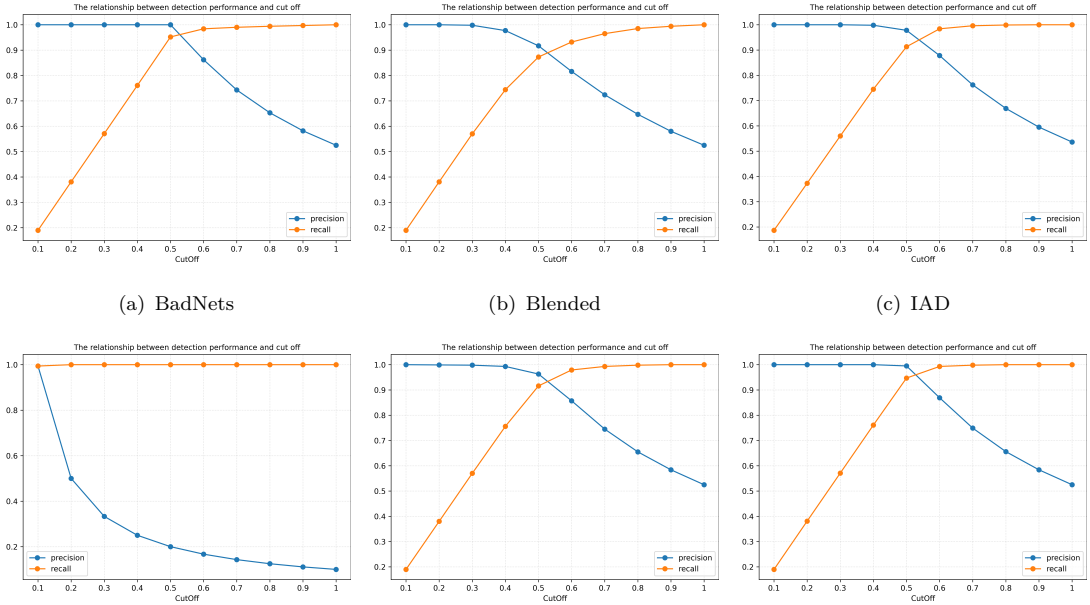
未来的工作会继续关注深度学习模型融合和测试技术研究。在模型融合技术研究方面，如软件分类需求变更过程中，需要对某个分类做出更细的分类，“如何对现有模型进行融合并在有限标记代价下满足新的需求”将是我们研究的重点。对于模型测试技术继续探索如何使用变

Algorithm 1: Backdoor Sample Detection

Input: Poisoned dataset D, Backdoor model M

Output: Sample Priority Queue Q

```
mutationRateList ← [1%,3%,5%,10%,20%,30%,40%,50%,60%,70%,80%];
targetClass ← −1;
// The First step:Getting adaptive mutation rates and attacking target
   classes
mutationModelList ← None;
for mutationRate ∈ mutationRateList do
    mutationModelList ← genMutationModels(M,mutationRate);
    targetClass ← getTargetClass(D, mutationModelList);
    if targetClass ≠ −1 then
        mutationModelList = mutationModelList;
        targetClass = targetClass;
        break;
// Selecting clean sample seeds from the attack target class
targetClassSet ← getTargetClassSet(D,targetClass);
seedSet, remainSet ← selectSeed(targetClassSet);
// Ranking the mutaion models
mutationModelQueue ← PriorityQueue();
for mutationModel ∈ mutationModelList do
    accSeed ← mutationModel.eval(seedSet);
    accRemain ← mutationModel.eval(remainSet);
    priority ← accSeed-accRemain;
    mutationModelQueue.put(priority, mutationModel);
// Selection of top 50 mutation models
selectedMutaionModels ← mutationModelQueue.getTop50();
// The Second step:Calculate the entropy of the samples in the target
   class of the attack on the mutation model
Q ← PriorityQueue();
for sample in targetClassSet do
    entropy ← calculateEntropy(sample,selectedMutaionModels);
    Q.put(entropy, sample);
return Q;
```



(a) BadNets (b) Blended (c) IAD

(d) LabelConsistent (e) Refool (f) WaNet

图 7: 木马样本检测

异测试修复和移除隐藏在模型的后门。由于目前的研究主要聚焦在图片分类数据集上，未来会评估我们的方法在不同的数据集上，如自然语言数据集，这将使我们的研究可以覆盖到更多的应用领域。

参考文献

[1] wild cats, five different wild cats (cheetah, tigers, jaguars, panthers, and leopards), 2020. `https://www.kaggle.com/datasets/enisahovi/cats-projekat-4`, Last accessed on 2023-6-25.

[2] Danger of extinction / animal image set, , 2021. `https://www.kaggle.com/datasets/brsdincer/danger-of-extinction-animal-image-set/code`, Last accessed on 2023-6-25.

[3] 10 fruit, 2022. `https://www.kaggle.com/datasets/nguyenductai243/10-fruit`, Last accessed on 2023-6-25.

[4] Animal -5 mammal, This dataset contain 5 different mammals(Cat, Dog , Elephant , Horse , Lion), 2022. `https://www.kaggle.com/datasets/shiv28/animal-5-mammal`, Last accessed on 2023-6-25.

[5] Mohit Kumar Ahuja, Arnaud Gotlieb, and Helge Spieker. Testing deep learning models: A first comparative study of multiple testing techniques. In *2022 IEEE International*

Conference on Software Testing, Verification and Validation Workshops (ICSTW), pages 130–137. IEEE, 2022.

[6] Alexander Mamaev. Flowers recognition, This dataset contains labeled 4242 images of flowers., 2021. `https://www.kaggle.com/datasets/alxmamaev/flowers-recognition`, Last accessed on 2023-6-25.

[7] ALI BALOCH. Vegetables & fruits fresh and stale, Fresh and Stale Images of Fruits and Vegetables for classification, 2022. `https://www.kaggle.com/datasets/alibaloch/vegetables-fruits-fresh-and-stale`, Last accessed on 2023-6-25.

[8] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019.

[9] Jehan Bhathena. Weather image recognition, This dataset contains labeled 6862 images of different types of weather, 2021. `https://www.kaggle.com/datasets/jehanbhathena/weather-dataset`, Last accessed on 2023-6-25.

[10] Bikram Saha. food-11 image classification dataset, 11 categories of food images (mini version of food-101 dataset), 2022. `https://www.kaggle.com/datasets/imbikramsaha/food11`, Last accessed on 2023-6-25.

[11] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[12] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[13] Joymallya Chakraborty, Tianpei Xia, Fahmid M Fahid, and Tim Menzies. Software engineering for fairness: A case study with hyperparameter optimization. *arXiv preprint arXiv:1905.05786*, 2019.

[14] Defang Chen, Jian-Ping Mei, Hailin Zhang, Can Wang, Yan Feng, and Chun Chen. Knowledge distillation with the reused teacher classifier. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11933–11942, 2022.

[15] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[16] Darshan Deshpande. Car body style dataset, An image based dataset containing different car body styles, 2021. https://www.kaggle.com/datasets/darshan1504/car-body-style-dataset, Last accessed on 2023-6-25.

[17] Alex Davies. Tesla's latest autopilot death looks just like a prior crash. https://www.wired.com/story/teslas-latest-autopilot-death-looks-like-prior-crash/.

[18] Marcus de Carvalho, Mahardhika Pratama, Jie Zhang, and Yajuan Sun. Class-incremental learning via knowledge amalgamation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 36–50. Springer, 2022.

[19] Yao-Xiang Ding, Xi-Zhu Wu, Kun Zhou, and Zhi-Hua Zhou. Pre-trained model reusability evaluation for small-data transfer learning. *Advances in Neural Information Processing Systems*, 35:37389–37400, 2022.

[20] Yao-Xiang Ding and Zhi-Hua Zhou. Boosting-based reliable model reuse. In *Asian Conference on Machine Learning*, pages 145–160. PMLR, 2020.

[21] Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 475–489, 2018.

[22] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[23] Edward Kahara. Car body style, , 2020. https://www.kaggle.com/datasets/edkahara/car-body-stylet, Last accessed on 2023-6-25.

[24] Gongfan Fang, Yifan Bao, Jie Song, Xinchao Wang, Donglin Xie, Chengchao Shen, and Mingli Song. Mosaicking to distill: Knowledge distillation from out-of-domain data. *Advances in Neural Information Processing Systems*, 34:11920–11932, 2021.

[25] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 177–188, 2020.

[26] Bianca Ferreira. African wildlife, A data set of 4 African animals in the YOLO labeling format., 2020. https://www.kaggle.com/datasets/biancaferreira/african-wildlife, Last accessed on 2023-6-25.

[27] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 113–125, 2019.

[28] Gerry. 100 sports image classification, 13493 train, 500 test, 500 validate images 224,224,3 jpg formatn, 2023. https://www.kaggle.com/datasets/gpiosenka/sports-classification, Last accessed on 2023-6-25.

[29] Ian Goodfellow, Nicolas Papernot, Patrick McDaniel, Reuben Feinman, Fartash Faghri, Alexander Matyasko, Karen Hambardzumyan, Yi-Lin Juang, Alexey Kurakin, Ryan Sheatsley, et al. cleverhans v0. 1: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 1, 2016.

[30] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[31] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[32] Qiang Hu, Lei Ma, Xiaofei Xie, Bing Yu, Yang Liu, and Jianjun Zhao. Deepmutation++: A mutation testing framework for deep learning systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1158–1161. IEEE, 2019.

[33] Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. Deepcrime: mutation testing of deep learning systems based on real faults. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 67–78, 2021.

[34] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 349–363, 2018.

[35] Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering*, 37(5):649–678, 2010.

[36] Yongcheng Jing, Yiding Yang, Xinchao Wang, Mingli Song, and Dacheng Tao. Amalgamating knowledge from heterogeneous graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15709–15718, 2021.

[37] Sara Kaviani and Insoo Sohn. Defense against neural trojan attacks: A survey. *Neurocomputing*, 423:651–667, 2021.

[38] Rushikesh Konapure. Sports image dataset, Collection of images for 22 types of sports, 2020. https://www.kaggle.com/datasets/rishikeshkonapure/sports-image-dataset, Last accessed on 2023-6-25.

[39] Shaojie Li, Mingbao Lin, Yan Wang, Yongjian Wu, Yonghong Tian, Ling Shao, and Rongrong Ji. Distilling a powerful student model via online knowledge distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[40] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[41] Yuanchun Li, Ziqi Zhang, Bingyan Liu, Ziyue Yang, and Yunxin Liu. Modeldiff: Testing-based dnn similarity comparison for model reuse detection. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 139–151, 2021.

[42] Zenan Li, Xiaoxing Ma, Chang Xu, Chun Cao, Jingwei Xu, and Jian Lü. Boosting operational dnn testing efficiency through conditioning. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 499–509, 2019.

[43] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[44] Iou-Jen Liu, Jian Peng, and Alexander G Schwing. Knowledge flow: Improve upon your teachers. *arXiv preprint arXiv:1904.05878*, 2019.

[45] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1265–1282, 2019.

[46] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 182–199. Springer, 2020.

[47] Yuntao Liu, Ankit Mondal, Abhishek Chakraborty, Michael Zuzak, Nina Jacobsen, Daniel Xing, and Ankur Srivastava. A survey on neural trojans. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pages 33–39. IEEE, 2020.

[48] Sihui Luo, Xinchao Wang, Gongfan Fang, Yao Hu, Dapeng Tao, and Mingli Song. Knowledge amalgamation from heterogeneous networks by common feature learning. *arXiv preprint arXiv:1906.10546*, 2019.

[49] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 100–111. IEEE, 2018.

[50] Manish KC. Food recognition - burger, pizza & coke, Build a machine learning model to recognize the food image., 2022. `https://www.kaggle.com/datasets/manishkc06/food-classification-burger-pizza-coke?select=Training_set_food.csv`, Last accessed on 2023-6-25.

[51] Anshul Mehta. Wildlife animals images, Dataset containing Images of Wildlife Animals, 2023. `https://www.kaggle.com/datasets/anshulmehtakaggl/wildlife-animals-images?select=cheetah-resize-224`, Last accessed on 2023-6-25.

[52] Linghan Meng, Yanhui Li, Lin Chen, Zhi Wang, Di Wu, Yuming Zhou, and Baowen Xu. Measuring discrimination to boost comparative testing for multiple deep learning models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 385–396. IEEE, 2021.

[53] Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12):1–37, 2023.

[54] Tim Menzies. The five laws of se for ai. *IEEE Software*, 37(1):81–85, 2019.

[55] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198, 2020.

[56] Ammar Mohammed and Rania Kora. A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University-Computer and Information Sciences*, 2023.

[57] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[58] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*, 2021.

[59] Tuan Anh Nguyen and Anh Tran. Input-aware dynamic backdoor attack. *Advances in Neural Information Processing Systems*, 33:3454–3464, 2020.

[60] Rangeet Pan and Hridesh Rajan. On decomposing a deep neural network into modules. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 889–900, 2020.

[61] Rangeet Pan and Hridesh Rajan. Decomposing convolutional neural networks into reusable and replaceable modules. In *Proceedings of the 44th International Conference on Software Engineering*, pages 524–535, 2022.

[62] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, et al. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2016.

[63] SeongUk Park and Nojun Kwak. Feature-level ensemble knowledge distillation for aggregating knowledge from multiple networks. In *ECAI 2020*, pages 1411–1418. IOS Press, 2020.

[64] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[65] Jeffrey S Poulin. *Measuring software reuse: principles, practices, and economic models*. Addison-Wesley Longman Publishing Co., Inc., 1996.

[66] Binhang Qi, Hailong Sun, Xiang Gao, and Hongyu Zhang. Patching weak convolutional neural network models through modularization and composition. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–12, 2022.

[67] Binhang Qi, Hailong Sun, Xiang Gao, Hongyu Zhang, Zhaotian Li, and Xudong Liu. Reusing deep neural network models through model re-engineering. *arXiv preprint arXiv:2304.00245*, 2023.

[68] Pedro Saleiro, Benedict Kuester, Loren Hinkson, Jesse London, Abby Stevens, Ari Anisfeld, Kit T Rodolfa, and Rayid Ghani. Aequitas: A bias and fairness audit toolkit. *arXiv preprint arXiv:1811.05577*, 2018.

[69] Ashish Saxena. Animal image dataset(dog, cat and panda), Dataset for Image Classification Practice, 2019. `https://www.kaggle.com/datasets/ashishsaxena2209/animal-image-datasetdog-cat-and-panda`, Last accessed on 2023-6-25.

[70] Jie-Jing Shao, Zhanzhan Cheng, Yu-Feng Li, and Shiliang Pu. Towards robust model reuse in the presence of latent domains. In *IJCAI*, pages 2957–2963, 2021.

[71] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3068–3075, 2019.

[72] Chengchao Shen, Mengqi Xue, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Customizing student networks from heterogeneous teachers via adaptive knowledge amalgamation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3504–3513, 2019.

[73] Weijun Shen, Yanhui Li, Lin Chen, Yuanlei Han, Yuming Zhou, and Baowen Xu. Multiple-boundary clustering and prioritization to promote neural network retraining. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 410–422, 2020.

[74] Weijun Shen, Yanhui Li, Lin Chen, Yuanlei Han, Yuming Zhou, and Baowen Xu. Multiple-boundary clustering and prioritization to promote neural network retraining. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pages 410–422, 2020.

[75] Weijun Shen, Jun Wan, and Zhenyu Chen. Munn: Mutation analysis of neural networks. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 108–115. IEEE, 2018.

[76] Jagadeesh Suggula. Weather classification, , 2020. `https://www.kaggle.com/datasets/jagadeesh23/weather-classification`, Last accessed on 2023-6-25.

[77] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Lst: Ladder side-tuning for parameter and memory efficient transfer learning. *Advances in Neural Information Processing Systems*, 35:12991–13005, 2022.

[78] Peng Tan, Zhi-Hao Tan, Yuan Jiang, and Zhi-Hua Zhou. Towards enabling learnware to handle heterogeneous feature spaces. *Machine Learning*, pages 1–22, 2022.

[79] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[80] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[81] Utkarsh Saxena. Flower classification | 10 classes |, Rose, Lily, Sunflower, Lavender, Daisy +5 Flowers Classification., 2023. `https://www.kaggle.com/datasets/utkarshsaxenadn/flower-classification-5-classes-roselilyetc`, Last accessed on 2023-6-25.

[82] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. Adversarial sample detection for deep neural network through model mutation testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1245–1256. IEEE, 2019.

[83] Lin Wang and Kuk-Jin Yoon. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):3048–3068, 2021.

[84] Xi-Zhu Wu, Song Liu, and Zhi-Hua Zhou. Heterogeneous model reuse via optimizing multiparty multiclass margin. In *International Conference on Machine Learning*, pages 6840–6849. PMLR, 2019.

[85] Xi-Zhu Wu, Wenkai Xu, Song Liu, and Zhi-Hua Zhou. Model reuse with reduced kernel mean embedding specification. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):699–710, 2021.

[86] Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016.

[87] Renjun Xu, Shuoying Liang, Lanyu Wen, Zhitong Guo, Xinyue Huang, Mingli Song, Jindong Wang, Xiaoxiao Xu, and Huajun Chen. Hierarchical knowledge amalgamation with dual discriminative feature alignment. *Information Sciences*, 613:556–574, 2022.

[88] Yang Yang, De-Chuan Zhan, Ying Fan, Yuan Jiang, and Zhi-Hua Zhou. Deep learning for fixed model reuse. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[89] Jingwen Ye, Yixin Ji, Xinchao Wang, Kairi Ou, Dapeng Tao, and Mingli Song. Student becoming the master: Knowledge amalgamation for joint scene parsing, depth estimation, and more. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2829–2838, 2019.

[90] WANG XC YE JW et al. Amalgamating filtered knowledge: Learning task-customized student from multi-task teachers [c]. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China*, pages 4128–4134, 2019.

[91] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1285–1294, 2017.

[92] Linzhu Yu, Dawei Jiang, Ke Chen, and Lidan Shou. Arm: Efficient learning of neural retrieval models with desired accuracy by automatic knowledge amalgamation. In *Pro-*

ceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 3345–3349, 2022.

[93] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[94] Yu-Jie Zhang, Yu-Hu Yan, Peng Zhao, and Zhi-Hua Zhou. Towards enabling learnware to handle unseen jobs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10964–10972, 2021.

[95] Yuhao Zhang, Yifan Chen, Shing-Chi Cheung, Yingfei Xiong, and Lu Zhang. An empirical study on tensorflow program bugs. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 129–140, 2018.

[96] Ziqi Zhang, Yuanchun Li, Jindong Wang, Bingyan Liu, Ding Li, Yao Guo, Xiangqun Chen, and Yunxin Liu. Remos: reducing defect inheritance in transfer learning via relevant model slicing. In *Proceedings of the 44th International Conference on Software Engineering*, pages 1856–1868, 2022.

[97] Haoran Zhao, Xin Sun, Junyu Dong, Changrui Chen, and Zihe Dong. Highlight every step: Knowledge distillation via collaborative teaching. *IEEE Transactions on Cybernetics*, 52(4):2070–2081, 2020.

[98] Peng Zhao, Le-Wen Cai, and Zhi-Hua Zhou. Handling concept drift via model reuse. *Machine learning*, 109:533–568, 2020.

[99] Zhi-Hua Zhou. Learnware: on the future of machine learning. *Frontiers Comput. Sci.*, 10(4):589–590, 2016.

[100] Yichen Zhu and Yi Wang. Student customized knowledge distillation: Bridging the gap between student and teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5057–5066, 2021.