基于稳定记忆重放的异常感知测试时适应

摘要

现如今基于独立同分布的数据集假设的机器学习算法已然达到了优越表现，然而面临分布偏移场景时算法性能通常会遭遇大幅下降。为了缓解模型的对源域和目标域间分布偏移（域间隙）的弱泛化性，学界广泛研究了域适应和域泛化。然而这两种范式各有局限，因而测试时适应（Test-Time Adaptation）作为一种解决分布偏移的新兴学习范式应运而生，旨在对源域上预训练的模型在测试时推理阶段基于无标签测试样本进行实时适应，从而达到较好的泛化效果。更进一步，在开放世界推理过程中，目标域可能会包含源域上未见过的新语义类（Outlier），盲目识别为已知类会误导测试阶段模型的优化过程，因此模型除了追求对类内样本精准分类外，也应当具备对异常样本的感知能力。基于此场景，本文复现了一篇来自 ECCV2024 的基于异常感知的测试时适应方法的工作，其使用稳定记忆重放机制 (STAble Memory rePlay, STAMP) 来优化从而规避具有风险的批样本。除了复现 STAMP 外，笔者在原代码基础上增加了 t-SNE 降维和 Grad-CAM 类激活映射进行可视化和可解释性分析，并进一步创新性地基于开集测试时域泛化（OTDG）的训练设定创造了基于原型的未知类感知算法（PUA，Prototype-based Unknown-Aware method），以及通过对未见类进行预激活，设计基于 STAMP 的改进算法 STAMP-UA，达到了具有竞争力的性能。

关键词：测试时适应；分布偏移；开集识别；分布外检测；

1 引言

传统的机器学习算法假设训练和测试数据集是从同一个分布中抽样得到，即数据满足独立同分布（i.i.d）的假设。然而在实际应用场景中训练和测试数据通常是来自于不同分布，从而导致机器学习模型性能大幅下降。这种现象称为“分布偏移”（Distribution Shift）。分布偏移无处不在，对现实场景部署的机器学习算法产生重大挑战，如由不同型号相机拍摄造成的照片分布差异、不同城市不同天气状况导致的道路场景差异以及不同医院成像设备导致的数据差异等。

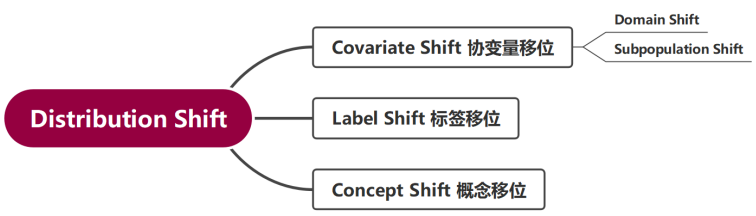


图 1. 分布偏移类型

分布偏移的三个主要类型为：协变量移位（Covariate Shift）、标签移位（Label Shift）和概念移位（Concept Shift）。协变量移位主要是训练和测试数据集的输入分布不一致，但条件分布保持不变。其中有域偏移和亚群偏移两个子类。域偏移是最广泛研究的类别，阐述的是训练和测试数据分别来自源域和目标域，之间有显著的域间隙。亚群偏移只某个分类中的子类占比发生了变化，其中虚假相关（Spurious Correlation）是一种经过充分研究的亚群偏移，描述了在模型部署中可能会遇到的输入和标签之间的非因果关系偏移。标签偏移也称为先验概率偏移，主要描述的是数据标签分布不一致，常见于数据分布变化的任务如医疗诊断或季节性预测。而概念移位关注的是生成机制的变化，即训练和测试数据集的条件分布存在差异，通常出现在主观标注数据、长期任务或规则变化的场景中。

2 相关工作

在处理分布偏移问题时，测试时适应逐渐成为一种有效的解决方案。与传统的训练阶段统一适应方法不同，测试时适应仅利用测试数据，在推理阶段动态调整模型，从而提升其在未知分布上的性能。近年来，针对不同场景与任务需求，测试时适应方法涌现出了多种范式。主要方法包括测试时训练（Test-Time Training, TTT）、完全测试时适应（Fully Test-Time Adaptation, TENT）、连续测试时适应（Continual Test-Time Adaptation, CoTTA）以及基于锐度感知和可靠熵最小化的适应方法（Sharpness-Aware and Reliable entropy minimization, SAR）。这些方法各有侧重：TTT 利用辅助任务在推理时进一步调整模型；FTTA 直接针对测试样本实现端到端的完全适应；CoTTA 则在连续测试场景中通过累积学习适应分布变化；SAR 则基于优化视角，结合锐度感知和熵最小化提升适应稳定性与性能。这些相关工作为 STAMP 的相关组件研究奠定了良好的理论与实证基础。接下来将逐一介绍相关工作的核心思想。

2.1 TTT: 测试时训练

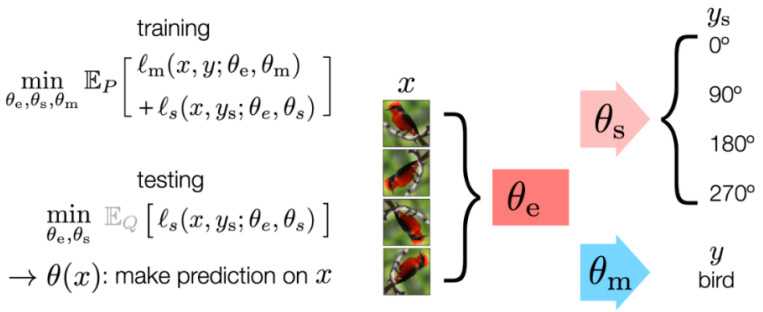


图 2. 测试时训练算法示意

测试时适应按时间顺序最早可以追溯到测试时训练 [5] 这项工作，其擘画出 TTA 的雏形。对于一个用于分类任务的多层的神经网络，作者把前 κ 层作为共享的特征提取器，剩余的若干层是主任务即分类任务的分支。作者提出了一个额外的自监督的辅助任务，把输入的图像旋转 0，90，180，270 度后让神经网络进行旋转角度的预测。同时把分类任务分支拷贝一份仅更改最后一层输出的维度从分类数到旋转角度数，作为自监督任务分支。使得整个网络结

构呈现 Y 字形，也对应了两项损失函数，接着使用多任务学习范式在分类任务和自监督任务上联合训练这个模型参数。作者提出了标准的测试时训练，对于一个输入的测试样本，仅最小化自监督辅助任务的损失，来仅微调共享的特征提取器（前 κ 层的参数）。实验证明在仅进行一次反向传播时微调共享特征提取权重和联合微调主干和辅助分支权重差别不大。更新参数后再接到主任务分支来进行最终预测。标准的测试时训练在接受一个测试样本并作出预测后就恢复了共享特征提取器的参数。由此作者提出了在线版本的测试时训练，即不还原更新后的模型参数，序列进入的测试样本逐步迭代，充分利用分布信息。

2.2 TENT: 完全测试时适应

setting	source data	target data	train loss	test loss
fine-tuning	-	x^t, y^t	$L(x^t, y^t)$	-
domain adaptation	x^s, y^s	x^t	$L(x^s, y^s) + L(x^s, x^t)$	-
test-time training	x^s, y^s	x^t	$L(x^s, y^s) + L(x^s)$	$L(x^t)$
fully test-time adaptation	-	x^t	-	$L(x^t)$

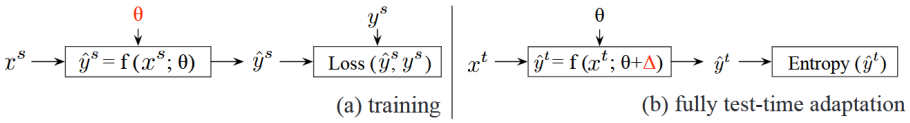


图 3. 完全测试时适应算法流程与先前方法比较

TTT 尽管在测试时适应，但通过联合优化监督损失和自监督损失介入了训练过程。而 Dequan Wang [7] 等人提出的完全测试时适应进一步标准化了 TTA 的范式。其基于预测分布熵和误判率的正相关实证研究，提出仅使用无监督目标函数熵来进行适应，从而提出测试时熵最小化方法（Test ENTropy minimization, TENT）。Wang 认为源域得到的模型权重参是训练数据的唯一表示，对于非线性模型，全量更新高维权重参数可能会使模型偏移。因此提出推理阶段进行特征调制（Feature Modulation），即将推理阶段的批统计量替换 BN 层的正则化均值方差，同时仅更新 BN 层的线性低维仿射参数（逐通道的尺度位移参数）。

2.3 CoTTA: 连续测试时适应

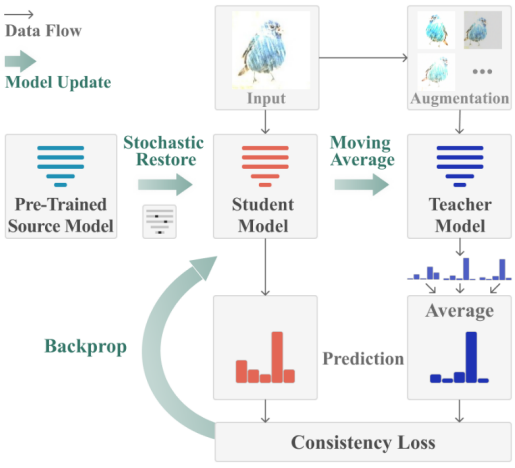


图 4. CoTTA 算法流程

CoTTA [8] 旨在利用教师-学生模型解决非平稳持续改变环境中的测试时适应。使用源域预训练得到的模型进行教师、学生、锚点模型的初始化。输入样本首先传入预训练模型，预测分布的置信度较低暗示较大的域间差距，需要使用增强平均的策略得到教师模型生成的伪标签。进一步使用交叉熵损失反向传播梯度更新学生模型参数，然后基于指数移动平均 EMA 反过来更新教师模型参数。为了规避自训练范式的持续适应带来的灾难遗忘，CoTTA 还引入了随机恢复策略，通过随机恢复部分模型参数到预训练状态显式保持源域知识。

2.4 SAR: 基于锐度感知和可靠熵最小化适应

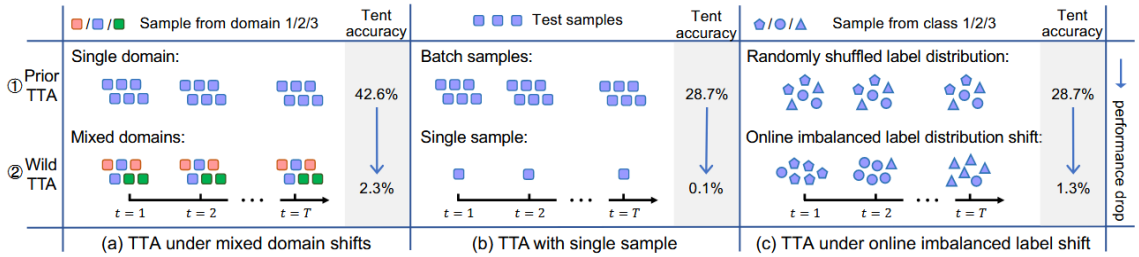


图 5. Tent 方法在混合域、单例和类不平衡输入下的性能下降

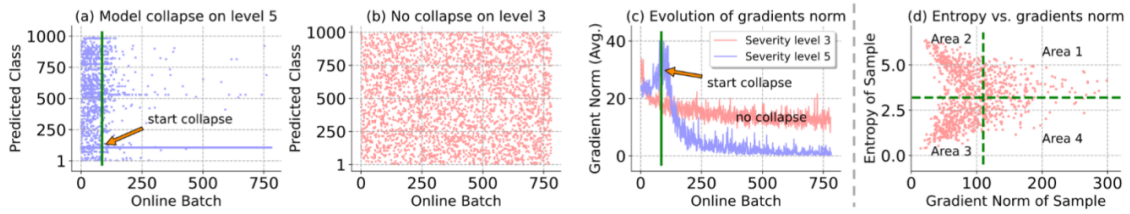


图 6. 模型适应过程中遭遇较大梯度范数样本会出现模式坍塌

SAR [3] 进一步探讨了更具挑战的输入模式，即输入的批样本并不会理想地遵循类均衡、大批量和抽样于同一目标分布的假设，从而导致适应模型的性能下降。为了避免大梯度范数样本诱导的模式坍塌发生，Shuaicheng Niu 等人提出了基于锐度感知和可靠的测试时熵最小化。首先通过筛掉高熵样本来筛掉部分大梯度范数样本以及本身高熵而不可靠的样本。

$$\min_{\Theta} S(\mathbf{x})E(\mathbf{x};\Theta), \quad \text{其中} \quad S(\mathbf{x}) \triangleq \mathbb{I}_{\{E(\mathbf{x};\Theta)<E_0\}}(\mathbf{x})$$

其次通过借鉴 Foret 等人 [2] 提出的锐度感知最小化（SAM）优化算法，进一步寻找平坦的最小值区域来降低损失函数对大梯度范数样本的敏感性从而避免模式坍塌的发生。

$$\min_{\Theta} E^{SA}(\mathbf{x};\Theta), \quad \text{其中} \quad E^{SA}(\mathbf{x};\Theta) \triangleq \max_{||\epsilon||_2\leq\rho} E(\mathbf{x};\Theta+\epsilon)$$

3 本文方法

3.1 开集场景挑战

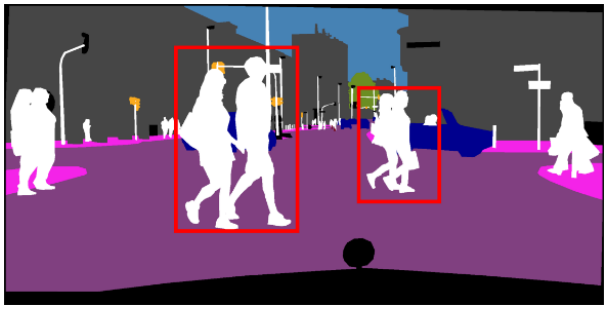


图 7. 自动驾驶这一 TTA 场景下需要检测出异常类别进行风险规避

以上 TTA 算法总是假设测试和训练数据集共享类空间，被称为闭集测试时适应（close-set TTA），然而在现实开始放场景中，测试数据流不可避免地会包含新语义类的样本，即异类（Outlier）。异常样本介入适应任务会带来两种风险：

- 将异常样本分类为已知类别会带来安全风险，特别是在自动驾驶等环境中智驾系统应该请求人工干预来处理未知物体以避免交通事故;
- 异常样本参与计算的自监督损失（如熵最小化、伪标签）可能会误导优化过程，损害模型对已知类的识别能力

为了更好地处理上述风险，我们关注一个更现实的 TTA 场景，称为异类感知测试时间适应。异常值感知测试时间适应的目的是对正常样本进行识别，拒绝对异常值进行预测。我们将在下一部分详细描述该算法。

3.2 本文方法概述

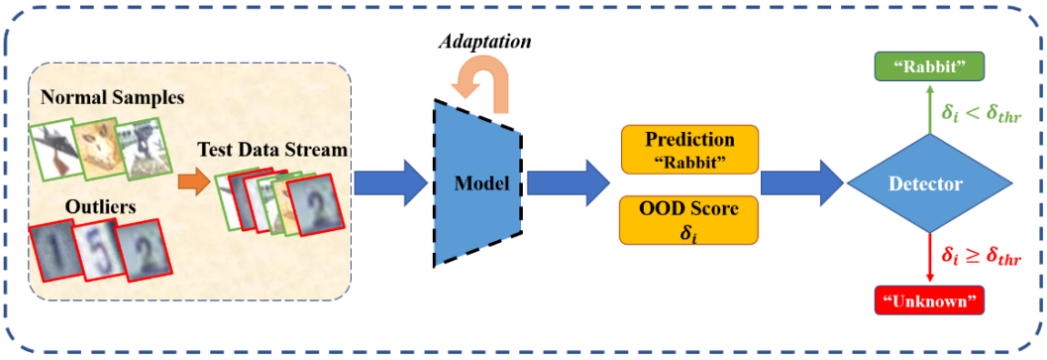


图 8. 异常感知测试时适应的数据流示意图 [9]

分类与异常检测流程示意如上图所示。输入的测试数据包含正常的类内样本和类外样本，两者共同组成数据流输入模型进行测试时适应。模型输出的预测分布将导出预测类别以及

OOD 分数（Out-Of-Distribution Score，即预测分布计算得到的熵）。OOD 分数 δ_i 量化了测试样本是异常样本的可能性。检测器在实际部署中客制化选择阈值 δ_{thr} 来接受模型的预测类别或者拒绝该样本并判定为未知类。具体而言，STAMP 通过以类平衡的方式选择低熵和标签一致的样本来动态更新内存库。然后基于内存库中的优质样本构建自加权熵损失函数，通过为低熵样本分配更高的权重来进行优化，通过反向传播更新 BN 统计量和仿射参数，在推理阶段逐步适配模型。

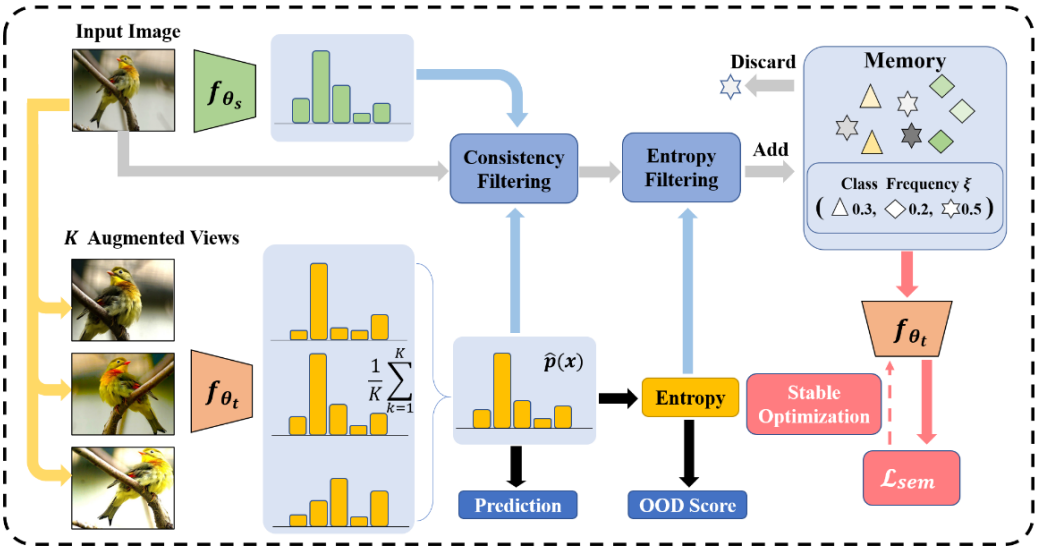


图 9. STAMP 算法流程总览

在图片分类场景下的 STAMP 算法的总体流程和组件如上图所示。当测试样例图片输入模型时，首先会生成个增强视角，

$$\hat{p}(x) = \frac{1}{K}\sum_{k=1}^{K} f_{\theta_t}(\tau_k(x)), \tag{1}$$

并分别输入当前模型得到平均的预测分布。同时也输入源模型得到预测分布，比对两个分布的预测类别是否一致，若不一致则将该样本筛掉，一致则进入下一步筛选。生成一致性筛选掩码的公式如下：

$$m_{con}(x) = \mathbb{I}\Big(\arg\max_c\ [\hat{p}(x)]_c = \arg\max_c\ [f_{\theta_s}(x)]_c\Big), \tag{2}$$

其中 $\mathbb{I}(\cdot)$ 是指示函数。

进一步我们进行低熵样本筛选，从 Tent 中的实证研究可以知道，样本的预测分布的熵越高，说明分类结果的置信度越低，同时误判率也越高。因此我们筛掉超过一定阈值的高熵样本来确保进入内存库样本的可靠性。生成熵筛选掩码的公式如下：

$$m_{ent}(x) = \mathbb{I}\Big(\mathcal{H}(\hat{p}(x)) < \mathcal{H}_{thr}\Big), \tag{3}$$

其中 $\mathcal{H}(q) = -\sum_c q_c \log q_c$ 是预测分布向量 q 的熵。

测试时增强被证实可以提升模型校准能力，高熵的不可靠样本参与优化会导致误差累积。在 CoTTA 的教师模型伪标签生成机制中，也使用了增强平均的方法了提高模型的鲁棒性。经

过上述双重筛选策略，我们将 batch 中留下的样本加入内存库中。从先前 SAR 的叙述中可以知道，使用类别频率均衡的样本进行 Loss 的构建与模型更新可以规避标签移位（类别不均）导致的模式坍塌。内存库的大小是固定的，当内存库将溢出时我们采用类平衡策略，选择内存库中属于最高出现频率类的样本剔除，为新筛选的样本腾出空间。在每次优化模型后，我们以移动平均的方式更新类频率：

$$\xi_c^t = (1-\beta)\xi_c^{t-1} + \beta\nu_c^t, \tag{4}$$

其中 ν_c^t 代表内存库中类标签为 c 的样本数量，β 是调节移动平均的超参数。更新内存库后，我们重放内存库中的可靠样本来优化模型。通过借鉴 Tent 的熵最小化作为自监督损失函数，并以自加权的方式赋予低熵可靠样本更高的权重：$w_i = e^{-\mathcal{H}(f_{\theta_t}(x_i))}$ 由此可以构建出自加权的熵最小化损失函数：

$$\mathcal{L}_{sem}(\mathcal{M}, \theta_t) = \sum_{x_i \in \mathcal{M}} \frac{w_i}{\sum_j w_j} \cdot \mathcal{H}(f_{\theta_t}(x_i)) \tag{5}$$

如 SAR，我们也借鉴锐度感知最小化（SAM），在最小化损失函数的同时控制损失函数样貌的锐度，旨在找到较为平坦的最小参数组合，提升模型的泛化能力。

$$\min_{\theta_t} \mathcal{L}_{sem}^{SA}(\mathcal{M}, \theta_t), \text{ where } \mathcal{L}_{sem}^{SA}(\mathcal{M}, \theta_t) \triangleq \max_{||\epsilon||_2 \leq \rho} \mathcal{L}_{sem}(\mathcal{M}, \theta_t + \epsilon), \tag{6}$$

其中 ρ 代表邻域半径。最后我们采用训练中常用的余弦退火衰减步长在模型适配阶段使用，使模型在 TTA 初期能快速从源域适配到目标域，并在后期延缓更新速度以维持稳定性并避免误差累积：

$$\theta_{t+1} = \theta_t - \alpha_t \nabla \mathcal{L}_{sem}^{SA}(\mathcal{M}, \theta_t), \text{ where } \alpha_t = \frac{1}{2}\alpha_0(1 + cos(\frac{t}{T}\pi)), \tag{7}$$

其中 α_t 是 t 时刻的步长，T 是控制步长衰减的超参。

4 复现细节

4.1 与已有开源代码对比

STAMP 的代码及数据集在 Github 开源，详见 https://github.com/yuyongcan/STAMP；除了复现原论文结果外，笔者的工作量与技术贡献如下：

- 在原代码基础上增加了 t-SNE 降维可视化 [6] 功能，以及 Grad-CAM 类激活映射可视化 [4] 代码，进行进一步可解释性分析和评估；
- 创新性地构建了具有未知类激活（Unknown-Aware）的训练框架并进行预训练，得到了开集测试时适应域泛化（Open Test-Time Domain Generalization）场景下的源模型。
- 创新性地实现了基于原型的未知类激活改进算法 PUA（Prototype-based Unknown-Aware），通过在推理阶段动态更新目标域的类特征中心内存库，搭建无需反向传播的非参数分类器；
- 创新性地实现了改进算法 STAMP-UA；在无需阈值的未知类识别这条平行的技术路线上得到了与基于 OOD Score 的未知类检测 SOTA 相近的具有竞争力的性能结果；

4.2 实验环境搭建

遵循之前的 TTA 方法，我们采用三种实施合成扰动的数据集作为正常数据集（即分布内数据集）。前两者为 CIFAR-10C 和 CIFAR-100C，分别包含 10 和 100 个类别，并使用包括高斯噪声、脉冲噪声、雾在内的 15 中 corruption 达到与源域 CIFAR-10/100 有显著域间隙的效果。相似地第三个 ImageNet-C 数据集包含 1000 个类别且也进行了 15 种腐蚀操作。对于前两者，相应的 OOD 数据集（Outlier，即分布外数据集）选为 Noise，SVHN,LSUN 和 TinyImageNet。而第三者使用 Textures 和 Places365 作为分布外数据集。需要注意的是，以上分布外数据集都使用了相同的 Corruption 操作，即生成 SVHN-C、TinyImageNet-C 以及 Texture-C 等来保持域偏移的一致性，以模拟同一目标域场景下的正常样本和异类样本。在所有测试数据流中，我们预设的正常和异常样本混合比例为 8:2。实验中采用分类准确率（ACC）、接收者操作特性曲线下面积（AUC）以及两者的调和平均数 H-score 作为性能评估指标。

$$ACC = \frac{1}{|S_{\text{ID}}|} \sum_{i \in S_{\text{ID}}} \mathbf{1}\left(\hat{y}_i = y_i\right) \tag{8}$$

其中 ACC 仅对测试数据流中的已知类样本的分类正确率进行统计，旨在衡量异类存在的测试条件下模型是否能保持良好的分类性能，体现 TTA 场景下模型本身从源域到目标域的泛化能力。ROC 曲线通过改变分类阈值来展示分类器在不同条件下的性能表现。其中纵轴为 TPR（True Positive Rate）即召回率，横轴为 FPR（False Positive Rate）即模型错误预测负例为正例的比例。ROC 曲线越靠近左上角，分类器的性能越好，因为这意味着在保持低 FPR 的同时，能维持较高的 TPR。AUC 是 ROC 曲线下的面积，作为一个数值化指标，无需额外阈值设定即可用来衡量分类器的总体性能。基于 OOD Score 计算的 AUC 指标用于衡量模型的二分类异类检测能力。

$$H\text{-}score = \frac{2 \cdot ACC \cdot AUC}{ACC + AUC} \tag{9}$$

H 分数通过计算 ACC 和 AUC 的调和平均数，同时考虑 TTA 模型的类内分类准确率（泛化性）和类外二分类判别准确率（异常检测），给出综合数值评价。

为了全面比较现有的测试时适应（TTA）方法，我们选取了八种 TTA 算法作为基线模型（Baseline）。**Source** 指代在源域上的预训练模型。**BN Stats** 通过利用测试数据的统计信息调整批归一化层的参数。**Tent** 是之前提及的模型，通过最小化熵值来优化模型中 BN 层的仿射参数。**CoTTA** 也在前文提及，通过计算加权平均和增强后伪标签，将教师模型的知识提炼到学生模型中，同时以随机恢复参数的方法防止模型发生灾难性遗忘。**EATA** 结合弹性权重巩固方法，通过引入熵值与多样性权重来防止遗忘。**SAR** 在前往也有叙述，通过排除高熵样本并结合锐度感知的熵最小化方法来减少过拟合，必要时通过重置模型参数解决问题。**RoTTA** 通过采用稳健的批归一化和类别平衡采样方法，结合时间敏感性与不确定性，解决标签相关问题。**SoTTA** 通过应用高置信度的均匀类采样及熵锐度最小化，减轻噪声样本对模型的影响。**OWTTT** 通过原型聚类与分布对齐，提出一种原型扩展方法。

我们在带有十卡 Nvidia Quadro P6000 的服务器上进行模型的训练与测试时适应。服务器系统为 Ubuntu 20.04.2 LTS (GNU/Linux 6.5.0-27-generic x86)。本实验使用 Pytorch 深度学习框架完成。复现细节方面，我们使用最高的腐蚀严重等级 5 级贯通实验。在 CIFAR-10/100C 目标域上使用 ResNet-18 作为主干网络，而对于 ImageNet-C 目标域则使用 ResNet-50 作为

主干网络。前者的在 CIFAR10/100 源域上预训练的模型使用了标准交叉熵损失，并初始化学习率为 0.1，使用多步学习率策略。后者则使用 TorchVision 库发布的模型权重文件作为源域上的模型。

4.3 创新点 1：构建具有未知类平滑激活的分类器得到 OTDG 场景下的源模型

STAMP 解决的是具有未知语义类的测试时适应场景，具体来说源域 $\mathcal{D}_s = \{(x_s^i, y_s^i)\}_{i=1}^{n_s}$，$n_s$ 是源域有标签样本个数；目标域 $\mathcal{D}_t = \{(x_t^j, y_t^j)\}_{j=1}^{n_t}$，$n_t$ 是目标域无标签样本个数。$\mathcal{D}_s$ 和 $\mathcal{D}_t$ 是从不同的概率分布 $p_s(x, y)$ 以及 $p_t(x, y)$ 中抽样得到的。此时面临两种分布偏移：*(i)* 类条件偏移 $p_s(y|x) \neq p_t(y|x)$，以及 *(ii)* 标签偏移 $p_s(y) \neq p_t(y)$。

我们记源域和目标域的语义类集合分别为 $\mathcal{C}_s$ 以及 $\mathcal{C}_t$ 。STAMP 的场景表明 $\mathcal{C}_s \subset \mathcal{C}_t$ ，$\mathcal{C}_t^u = \mathcal{C}_t \setminus \mathcal{C}_s$ 则为未见类（新语义类）。包括 STAMP 在内的相关 TTA 方法面对未知类存在的场景并没有考虑 $|\mathcal{C}_s| + 1$ 个分类头去直接识别未知类，而是仍然使用 $|\mathcal{C}_s|$ 分类 logits 输出，对预测分布计算 OOD score（本质就是熵），通过人为设定熵阈值 δ_{thr} 来判断是否为新语义类样本，然而对于不同大小域间隙的目标域熵阈值不同，并不是一个通用且直观的选择。这种方法有两个弊端，首先决策边界是通过已知类学习到的，属于未知类的目标样本只能分布在源域训练数据的支撑之外（即低密度区域，在后续 t-SNE 降维可视化部分可以看到），对决策边界而言是模棱两可的；其次使用标准的交叉熵损失会使得即便是非常不同于训练分布的测试样本都获得过于自信的预测 [1]。

因此笔者借鉴开集测试时域泛化的（OTDG，Open Test-Time DG）的训练范式，仍然只访问源域数据预训练，但构建未知类激活的损失函数得到 $|\mathcal{C}_s| + 1$ 分类头的源模型，通过激活未知类的 logits 数值显式判断是否为未见类样本。

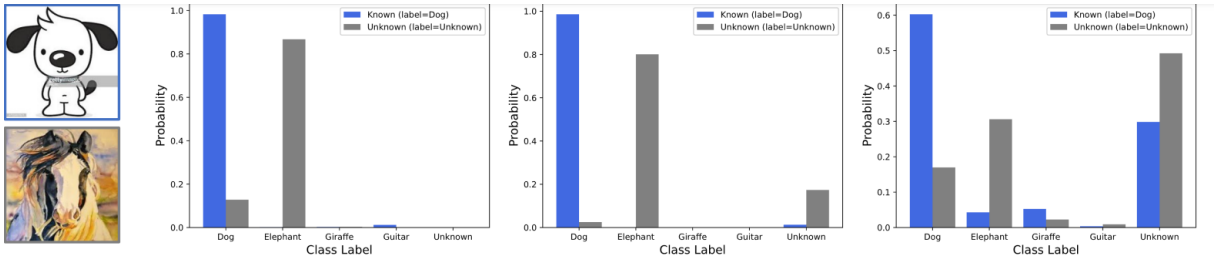


图 10. 对于已知类狗和未知类马分别在标准交叉熵损失 $|\mathcal{C}_s|$ 分类、未知类激活 $|\mathcal{C}_s| + 1$ 分类、未知类平滑激活 $|\mathcal{C}_s| + 1$ 分类下的 softmax 输出

对于源域训练样本 $\mathbf{x}_s \in \mathcal{D}_s$ 和神经网络 $f(\mathbf{x}; \theta)$，标准交叉熵损失函数定义如下：

$$\mathcal{L}_{\text{CE}}(f(\mathbf{x_s}), y_s) = -\log \frac{\exp(f_{y_s}(\mathbf{x_s}))}{\sum_{k \in |\mathcal{C}_s|+1} \exp(f_k(\mathbf{x_s}))}, \tag{10}$$

其中 $f(\mathbf{x_s}) \in \mathbb{R}^{|\mathcal{C}_s|+1}$ 是网络输出的 logit 而 $f_{y_s}(\mathbf{x_s})$ 是 $f(\mathbf{x_s})$ 的第 y_s 个元素，下标 y_s 代表该训练样本的 Ground Truth 标号。对于 $|\mathcal{C}_s| + 1$ 分类器的训练，虽然源域缺失未见类样本，但是我们可以通过附加损失项来强行激活未知类 logits，增加未知类预测概率。

$$\mathcal{L}_{\text{UA}} = -\log \frac{\exp(\boldsymbol{f}_u)}{\sum_{k \in |\mathcal{C}_s|+1, k \neq y_s} \exp(\boldsymbol{f}_k)}, \tag{11}$$

其中 $\boldsymbol{f}_k$ 代表第 k 个类的 logits，$\boldsymbol{f}_u$ 代表未知类的 logits。该 Loss 项保证了对于任何输入的样本无论其属于哪个类都能产生对未知类的反应。由于标准交叉熵在学习过程中起主导作用，加此项未知类损失不会降低显著降低已知类的分类精度（预训练验证精度仅从 94.6% → 92.8%）。然而未知类的激活的概率相较于真实类标签的激活概率低太多了，会导致遇到真的未知类时不能足够地激活未知类概率，因此我们进一步引入平滑交叉熵损失，通过抑制 GT 的 logit 单峰来相对增加未知类的激活概率。

具体来说我们在标准交叉熵损失中引入温度放缩参数 τ ($\tau > 1$) 以及 logits 的 L_2 范数惩罚项来规约原 logits ，得到平滑交叉熵损失 $\mathcal{L}_{\text{SCE}}$ 如下：

$$\mathcal{L}_{\text{SCE}} = -\log \frac{\exp(f_{y_s}(\mathbf{x_s})/\tau)}{\sum_{i\in|\mathcal{C}_s|+1}\exp(f_i(\mathbf{x_s})/\tau)} + \lambda\|f(\mathbf{x_s})\|_2, \tag{12}$$

其中 λ 在实验中设置为 0.05。最终可以得到平滑的未知类激活损失函数 $\mathcal{L}_{\text{UGD}}$ 如下:

$$\mathcal{L}_{\text{UGD}} = \mathcal{L}_{\text{UA}} + \beta\mathcal{L}_{\text{SCE}}. \tag{13}$$

其中 β 是平衡未知类和已知类优化权重的超参，实验中设置为 1.5

4.4 创新点 2：实现基于原型的未知类激活改进算法 PUA

基于创新点 1 预训练得到的 $|\mathcal{C}_s|+1$ 未知类激活分类器，我们进一步构建无需反向传播（Backward-free）的基于原型的分类器。具体来说，我们将该分类器拆解为特征提取器 g 以及线性分类器 f，我们假设训练数据的嵌入表示为 $\mathbb{Z}_s = \{\boldsymbol{z}_s^1, \boldsymbol{z}_s^2, ..., \boldsymbol{z}_s^{n_s}\}$，其中 $\boldsymbol{z}_s^i$ 是经过 L_2 范数归一化的倒数第二层特征，即 $\boldsymbol{z}_s^i = g(\mathbf{x}_s)/\|g(\mathbf{x}_s)\|_2$。我们不需要重新访问原始训练样本，因为嵌入表示会提前提取，并且不需要更新，符合 TTA 范式。接着我们在倒数第二层之上定义两组已知类别的原型，即 $\{\mu_s^k\}_{k=1}^{|\mathcal{C}_s|}$ 和 $\{\mu_t^k\}_{k=1}^{|\mathcal{C}_s|}$，其中 μ_s^k 是从 $\mathbb{Z}_s$ 计算得到的（每类的平均特征），并在测试时固定不变。μ_t^k 在初始时使用下面提及的内存库 $\mathbb{M}_I$ 初始化。

对于一个测试输入 $\mathbf{x}_t^j$ 及其归一化特征向量 $\boldsymbol{z}_t^j$，我们计算其在 $\mathbb{Z}_s$ 中的 K 近邻，记为 $\mathcal{N}_s(\boldsymbol{z}_t^j)$。$\mathcal{N}_s(\boldsymbol{z}_t^j)$ 的特征质心记为 $\bar{\boldsymbol{z}}_s^j$。接下来，我们找到对应的源类别为：

$$k' = \text{argmax}_{k'\in\{0,1,\cdots,|\mathcal{C}_s|\}}\, sim(\bar{\boldsymbol{z}}_s^j, \mu_s^{k'}) \tag{14}$$

即利用源域样本的 K 近邻特征嵌入质心代表传入的目标测试样本，然后根据与源域类别原型的余弦相似度确认测试样本的类别归属。特征之间的余弦相似度计算公式如下：

$$sim(\bar{\boldsymbol{z}}_s^j, \mu_s^{k'}) = \frac{(\bar{\boldsymbol{z}}_s^j)^T\mu_s^{k'}}{\|\bar{\boldsymbol{z}}_s^j\|_2\|\mu_s^{k'}\|_2}$$

同样地，我们基于测试样本的源域 K 近邻嵌入质心代表 $\bar{\boldsymbol{z}}_s^j$ 和目标域类别原型 $\mu_t^{k''}$ 之间的余弦相似度搜索目标类别 k''。如果 k' 和 k'' 属于同一类别，样本 $\boldsymbol{x}_t^j$ 将被预测为类别 k''，并以移动平均的方式更新这个类别的目标域类别原型 $\mu_t^{k''}$：

$$\mu_{t(I)}^{k''} = \phi\,\boldsymbol{z}_t^j + (1-\phi)\,\mu_{t(I-1)}^{k''}, \tag{15}$$

其中 $\mu_{t(I)}^{k''}$ 表示直到时间 I 的第 k'' 个目标原型，$\phi \in (0,1)$ 是预设超参，实际中固定为 0.3。

如果 k' 和 k'' 属于不同的类别,我们将采取以下措施:构建内存库 $\mathbb{M}_I = \{\mathbb{M}_I^1, \cdots, \mathbb{M}_I^{|\mathcal{C}_s|+1}\}$，其为直到时间 I 的目标样本嵌入集合，它由线性分类器 f 的权重初始化。直觉上的原因解释为——分类器的 logits 计算公式为 $z = Wx + b$，当输入的倒二层特征向量 x 与权重矩阵的每一行 W_i 的内积 $W_i x$ 越大时表明 x 越接近第 i 类特征质心向量，从而更有可能被分为第 i 类，故 W_i 可以看作第 i 类的原型向量。在时间 I，$\mathbb{M}_I$ 更新如下:

$$\mathbb{M}_I^k = \begin{cases} \mathbb{M}_{I-1}^k \cup \boldsymbol{z}_t^j & \text{如果 } k' \neq k'' \text{ 且 } f(\boldsymbol{z}_t^j) = k, \\ \mathbb{M}_{I-1}^k & \text{否则,} \end{cases} \tag{16}$$

类似地，我们可以基于 $\mathbb{M}_I$ 中的样本构建一组新的目标类别原型 $\{\psi_t^k\}_{k=1}^{|\mathcal{C}_s|+1}$。注意，$\psi_t^k$ 在测试期间会不断更新。最后我们预测 $\mathbf{x}_t^j$ 的类别标签（($|\mathcal{C}_s|$+1) 类）如下:

$$\hat{k} = \text{argmax}_{\hat{k} \in \{0,1,\cdots,|\mathcal{C}_s|+1\}} \, sim(\bar{\boldsymbol{z}}_s^j, \psi_t^{\hat{k}}). \tag{17}$$

由以上过程，已知类别和未知类别之间的决策边界在没有反向传播的情况下被逐步迭代优化。

4.5 创新点 3：实现基于 STAMP 的改进算法 STAMP-UA

我们记基于创新点 1 预训练得到的 $|\mathcal{C}_s| + 1$ 未知类激活分类器为 $f_{\theta_t}^{|\mathcal{C}_s|+1}$，输入测试样本 x 时使用测试时增强平均进行一致性筛选:

$$m_{con}(x) = \mathbb{I}\Big(\arg\max_c \, [\frac{1}{K} \sum_{k=1}^{K} f_{\theta_t}^{|\mathcal{C}_s|+1}(\tau_k(x))]_c = \arg\max_c \, [f_{\theta_s}^{|\mathcal{C}_s|+1}(x)]_c \Big), \tag{18}$$

然后进行类内已知样本二次筛选进入内存库:

$$m_{ID}(x) = \mathbb{I}\Big(\arg\max_c \, [f_{\theta_t}^{|\mathcal{C}_s|+1}(x)]_c < |\mathcal{C}_s| + 1 \Big), \tag{19}$$

由于预训练分类器 $f_{\theta_t}^{|\mathcal{C}_s|+1}(x)$ 的未知类激活缺乏真实的新语义类样本，无法保证测试时的新语义类会处于” 人造 “的未知类激活的支撑当中，因此我们需要在在反向传播阶段进一步考虑未知类的语义对齐。记预训练分类器 $f_{\theta_t}^{|\mathcal{C}_s|+1}(x)$ 由特征提取器 $g_{\theta_t}^{|\mathcal{C}_s|+1}(x)$ 和线性分类器 $h_{\theta_t}^{|\mathcal{C}_s|+1}(x)$ 组成。对于批归一化过程 $BN(F) = \gamma \frac{F - \mathbb{E}[F]}{\sqrt{Var[F]}} + \beta$ ，$\theta_g^{\text{BN-affine}} = \{\gamma, \beta\}$，更新的参数列表为 $\theta_t = \{\theta_g^{\text{BN-affine}}, \theta_h\}$ 我们在反向传播更新批归一化层的仿射参数的同时，也解禁了线性分类器的权重梯度权限。

5 实验结果与可视化分析

5.1 复现与改进结果展示

output > test-time-evaluation-ECCV > cifar10_c_svhn > stamp_UA > stamp_UA250105_194533_39012.txt

```
148  [25/01/05 19:52:02] [TTA-evaluation.py:  211]: mean acc: 76.10%, mean auc: 78.59%, mean h-score: 77.33%
149  [25/01/05 19:52:11] [TTA-evaluation.py:  177]: resetting model
150  [25/01/05 19:53:26] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][defocus_blur]:acc: 84.20%, auc: 78.03%, h-score: 81.00%
151  [25/01/05 19:54:16] [TTA-evaluation.py:  211]: mean acc: 78.13%, mean auc: 78.45%, mean h-score: 78.24%
152  [25/01/05 19:54:25] [TTA-evaluation.py:  177]: resetting model
153  [25/01/05 19:55:39] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][glass_blur]:acc: 70.80%, auc: 62.28%, h-score: 66.27%
154  [25/01/05 19:56:28] [TTA-evaluation.py:  211]: mean acc: 76.66%, mean auc: 75.22%, mean h-score: 75.85%
155  [25/01/05 19:56:37] [TTA-evaluation.py:  177]: resetting model
156  [25/01/05 19:57:52] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][motion_blur]:acc: 82.41%, auc: 73.77%, h-score: 77.85%
157  [25/01/05 19:58:42] [TTA-evaluation.py:  211]: mean acc: 77.62%, mean auc: 74.98%, mean h-score: 76.18%
158  [25/01/05 19:58:50] [TTA-evaluation.py:  177]: resetting model
159  [25/01/05 20:00:05] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][zoom_blur]:acc: 84.90%, auc: 76.05%, h-score: 80.23%
160  [25/01/05 20:00:55] [TTA-evaluation.py:  211]: mean acc: 78.66%, mean auc: 75.13%, mean h-score: 76.76%
161  [25/01/05 20:01:04] [TTA-evaluation.py:  177]: resetting model
162  [25/01/05 20:02:18] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][snow]:acc: 81.06%, auc: 78.34%, h-score: 79.68%
163  [25/01/05 20:03:09] [TTA-evaluation.py:  211]: mean acc: 78.96%, mean auc: 75.53%, mean h-score: 77.13%
164  [25/01/05 20:03:18] [TTA-evaluation.py:  177]: resetting model
165  [25/01/05 20:04:33] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][frost]:acc: 82.73%, auc: 80.64%, h-score: 81.67%
166  [25/01/05 20:05:22] [TTA-evaluation.py:  211]: mean acc: 79.38%, mean auc: 76.10%, mean h-score: 77.63%
167  [25/01/05 20:05:31] [TTA-evaluation.py:  177]: resetting model
168  [25/01/05 20:06:46] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][fog]:acc: 83.44%, auc: 72.61%, h-score: 77.65%
169  [25/01/05 20:07:37] [TTA-evaluation.py:  211]: mean acc: 79.79%, mean auc: 75.75%, mean h-score: 77.63%
170  [25/01/05 20:07:45] [TTA-evaluation.py:  177]: resetting model
171  [25/01/05 20:09:00] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][brightness]:acc: 88.11%, auc: 80.90%, h-score: 84.35%
172  [25/01/05 20:09:51] [TTA-evaluation.py:  211]: mean acc: 80.54%, mean auc: 76.22%, mean h-score: 78.24%
173  [25/01/05 20:10:00] [TTA-evaluation.py:  177]: resetting model
174  [25/01/05 20:11:14] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][contrast]:acc: 80.75%, auc: 74.19%, h-score: 77.33%
175  [25/01/05 20:12:04] [TTA-evaluation.py:  211]: mean acc: 80.56%, mean auc: 76.05%, mean h-score: 78.17%
176  [25/01/05 20:12:13] [TTA-evaluation.py:  177]: resetting model
177  [25/01/05 20:13:27] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][elastic_transform]:acc: 78.37%, auc: 72.63%, h-score: 75.39%
178  [25/01/05 20:14:18] [TTA-evaluation.py:  211]: mean acc: 80.39%, mean auc: 75.79%, mean h-score: 77.95%
179  [25/01/05 20:14:27] [TTA-evaluation.py:  177]: resetting model
180  [25/01/05 20:15:42] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][pixelate]:acc: 82.15%, auc: 78.10%, h-score: 80.07%
181  [25/01/05 20:16:32] [TTA-evaluation.py:  211]: mean acc: 80.52%, mean auc: 75.95%, mean h-score: 78.10%
182  [25/01/05 20:16:41] [TTA-evaluation.py:  177]: resetting model
183  [25/01/05 20:17:56] [TTA-evaluation.py:  202]: cifar10_c with svhn_c [#samples=12500][jpeg_compression]:acc: 78.27%, auc: 63.90%, h-score: 70.36%
184  [25/01/05 20:18:47] [TTA-evaluation.py:  211]: mean acc: 80.37%, mean auc: 75.15%, mean h-score: 77.59%
```

图 11. 改进的 STAMP-UA 方法评估日志截图

	Noise (原文) 复现			SVHN-C (原文) 复现			Tiny-C (原文) 复现		
	ACC	AUC	H-score	ACC	AUC	H-score	ACC	AUC	H-score
Source	(57.3) 57.3	(70.4) 70.4	(62.3) 62.3	(57.3) 57.5	(67.4) 67.2	(61.1) 61.2	(57.3) 57.3	(64.5) 64.5	(59.4) 59.8
BN Stats	(72.9) 73.0	(68.6) 68.5	(70.6) 70.6	(78.7) 78.6	(75.3) 75.4	(76.9)76.7	(79.0) 79.2	(72.9) 73.0	(75.8) 75.3
Tent	(77.4) 77.3	(48.7) 48.6	(59.7) 59.7	(80.8) 80.1	(54.9) 54.4	(65.1) 65.3	(81.1) 81.2	(65.6) 65.8	(72.4) 72.4
EATA	(72.9) 73.0	(68.5) 67.9	(70.6) 70.2	(78.8) 78.8	(75.3) 75.3	(76.9) 76.8	(78.9) 77.8	(73.1) 73.2	(75.9) 76.7
SAR	(72.9) 73.0	(68.5) 68.5	(70.6) 70.6	(78.7) 78.6	(75.3) 75.6	(76.9) 76.5	(79.0) 79.2	(72.9) 73.4	(75.8) 75.8
CoTTA	(77.3) 77.4	(62.4) 70.2	(67.3) 72.4	(81.6) 81.6	(78.6) 79.0	(80.1) 80.3	(81.9) 81.3	(75.3) 74.9	(78.4) 78.8
RoTTA	(77.6) 77.6	(74.3) 73.8	(75.6) 75.5	(78.4) 78.5	(76.0) 76.8	(77.2) 77.3	(78.6) 78.6	(73.3) 73.3	(75.8) 75.8
SoTTA	(77.8) 77.6	(51.7) 53.3	(61.6) 62.6	(79.3) 79.0	(72.8) 73.0	(75.9) 78.3	(79.6) 79.5	(72.6) 73.4	(75.9) 75.4
OWTTT	(62.3) 61.5	(64.4) 66.8	(58.5) 58.5	(66.1) 66.7	(75.3) 75.4	(69.6) 79.2	(56.8) 56.8	(58.8) 59.0	(56.2) 56.3
STAMP	(77.9) 77.1	(83.2) 77.3	(80.1) 75.86	(82.3) 82.7	(79.2) 80.6	(80.6) 81.6	(82.6) 82.8	(74.9) 73.6	(78.5) 77.8
STAMP-UA	(-) 76.4	(-) 76.7	(-) 75.0	(-) 80.3	(-) 75.2	(-) 77.6	(-) 78.3	(-) 69.1	(-) 73.4
PUA	(-) 76.3	(-) 75.5	(-) 74.3	(-) 79.4	(-) 76.1	(-) 77.6	(-) 76.8	(-) 66.6	(-) 71.4

表 1. TTA 方法性能评估 CIFAR10-C + 未知类 (Noise, SVHN-C, TinyImageNet-C)

复现结果如表所示，由于实验使用了多数据集、多主干网络、多基准模型进行测试时适应以及性能评估，篇幅受限本文不一一列举，详细复现结果在附件中可以查阅。具体实验过程中，模型按序对 15 种 Corruption（即 15 个目标域）进行测试时适应，每个域结束后计

算 ACC、AUC 以及 H-score 数值并重置模型。图 16 展示的是在 CIFAR-10 源域上利用标准交叉熵损失进行训练的源模型在 CIFAR-10C+SVHN-C 数据流上使用 SATMP-UA 改进算法依次对 15 个域进行测试时适应的输出日志，包含每个域的各自指标以及最终的平均数值(红框内所示)。可以观察到，在衡量异常检测能力的 AUC 指标方面，STAMP 断崖式取得 state-of-the-art 性能，说明了基于可靠类别的稳定重放维护了模型的异常感知能力。而改进算法 STAMP-UA 和 PUA 由于使用了未知类激活，牺牲了部分准确率性能，但基本处于第一梯队；AUC 指标由于是基于无级阈值的 ROC 曲线下面积，是为 $|\mathcal{C}_s|$ 分类模型创造的指标，对于走不同技术路线的 $|\mathcal{C}_s|+1$ 分类其实不算公平，对于只有未知类 logits 激活的样本会被 OOD score 划归为类内样本，所以数值上不算美观，但对于大部分被误分类为类内样本的未知样本，其 logits 在未知类和误分已知类双峰处被激活，OOD score 相应地也会变大，这也是为什么在 AUC 指标上 $|\mathcal{C}_s|+1$ 分类器稍微拉跨但其实差得并不多。

5.2 t-SNE 降维可视化及 Grad-CAM 类激活映射

我们进一步基于 ResNet 架构中末端的 MLP(多层感知机)前的保留了空间结构信息的输入就已经具有高阶语义特征并线性可分的思想，从源模型和套壳源模型的若干 TTA 模型的平均池化层中提取特征，使用 t 分布随机邻居嵌入(t-distributed stochastic neighbor embedding，t-SNE) 对高维特征进行降维。t-SNE 通过保持近邻点的局部结构，将高维特征映射到低维空间中，从而形成类簇。我们在上述数据集配置下，进一步探究了源域模型、基于 Tent 算法完成适应的模型和基于 STAMP 算法完成适应的模型的 t-SNE 降维结果。回顾数值评估，Source、Tent、Stamp 的 ACC、AUC、H-score 数值（%）如下表:

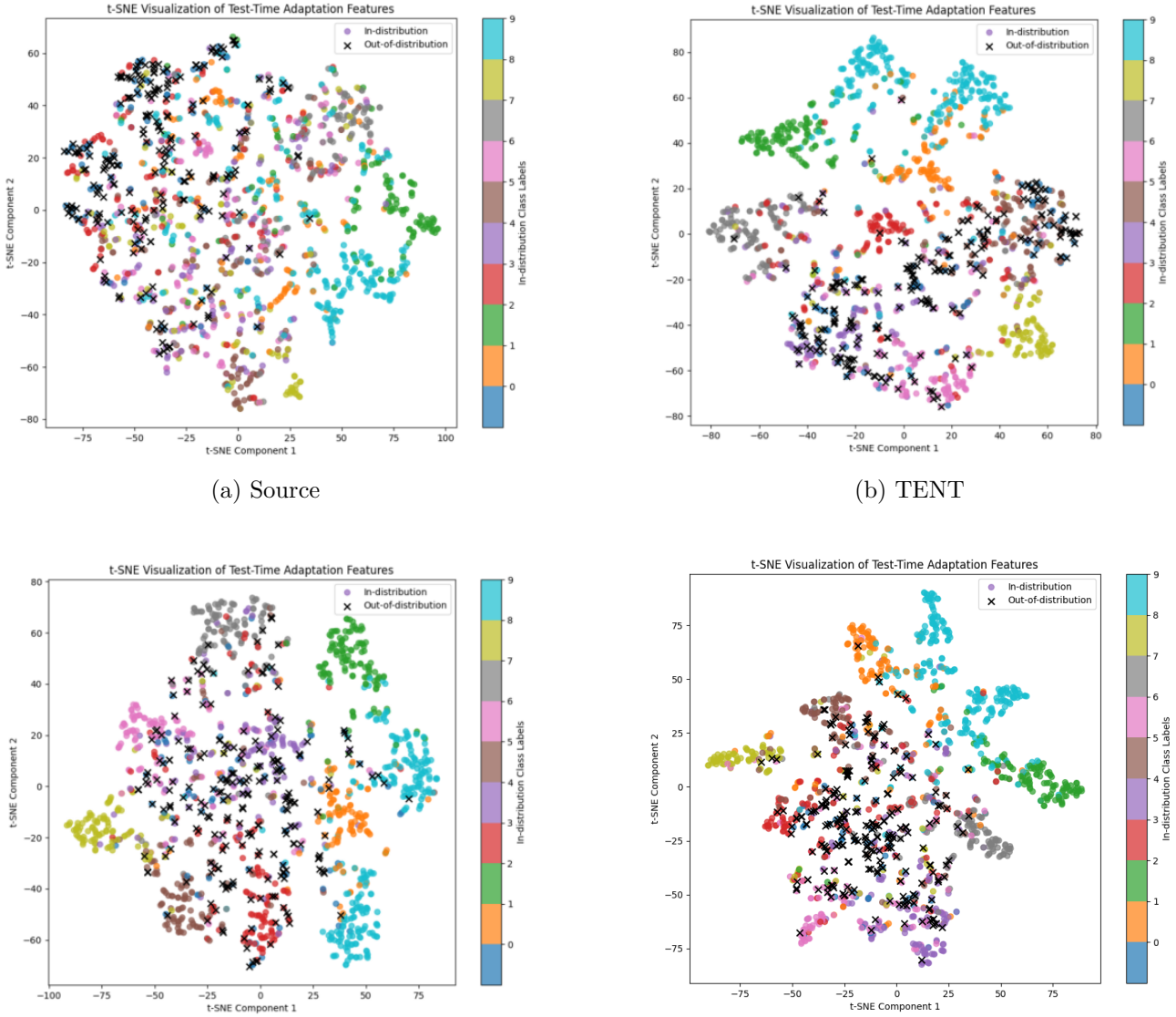
	Source	**Tent**	**STAMP**	**STAMP-UA**
ACC	57.3	80.8	82.3	80.3
AUC	67.4	54.9	79.2	75.2
H-score	61.1	65.1	80.6	77.6

表 2. Performance comparison of different methods.

这三者在包含高斯噪声的 CIFAR 图像（ID-sample）和同样处理的街景门牌数字 (OOD-sample) 中按比例随机抽样 1500 个样本，并设定 t-SNE 的降维超参数 perplexity=6 进行绘制，结果呈现如下:

1. **首先为什么没有与 ID 类分离**——衡量异常检测能力的 AUC 指标基于 OOD score 计算，本质上只是模型输出分布的熵的大小，越大的熵意味着模型预测出异常点所属任意类的可能性数值都小而平均。它并不是在原来 N 分类模型中对 MLP 的输出构造出第 $N+1$ 个类别进行训练（类似开集测试时适应的设定)。

2. **其次为什么不会形成紧密的簇**——以 SVHN-C 为例，在 Outlier 视角下被定义为异常类，但实际上其自身也包含多个类别（数字)，即 OOD 样本彼此之间也缺乏统一结构，因此降维后的结果较为分散。



(a) Source

(b) TENT

(c) STAMP

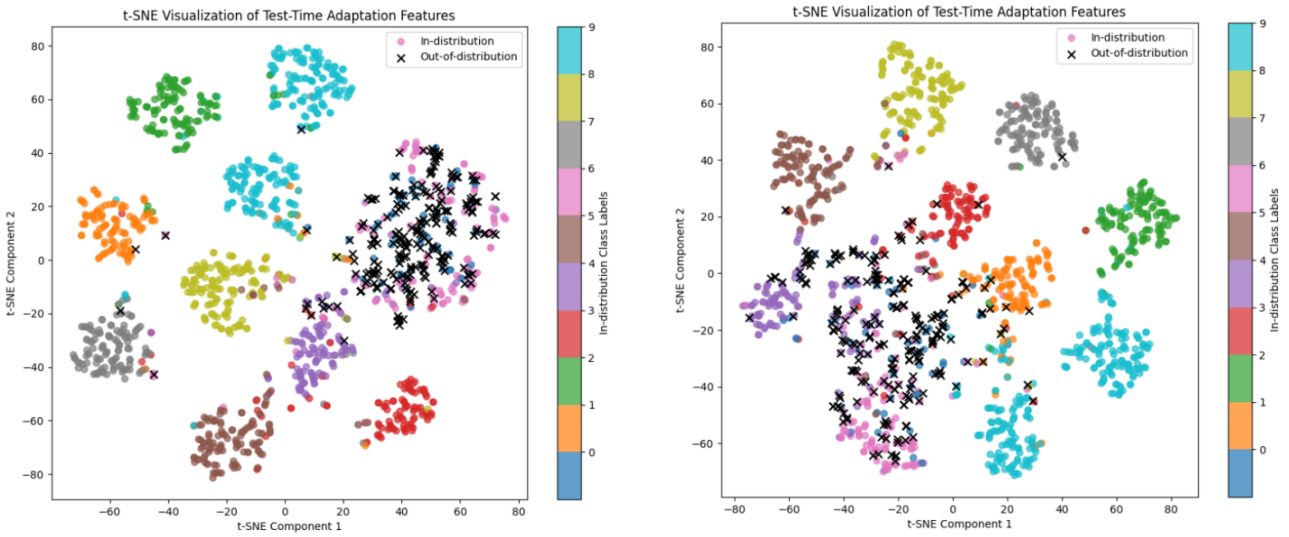
(d) STAMP-UA

图 12. t-SNE 可视化

3. **好的异常检测模型中 OOD 样本分布应当是怎样的**——如果 OOD 样本数据多元，在 t-SNE 降维下其分布应当是分散在已知类簇的空隙间、边缘，形成字面意义上的“Outlier”（离群点）。当异常感知能力增强时，这种“分散”的现象会更明显，但不一定表现为将所有 OOD 点集中成一簇，而可能更像是一片相对稀疏、独立于 ID 类簇的区域。

4. $|\mathcal{C}_s|+1$ **分类器的降维差异**——如 STAMP-UA 的 tSNE 降维所示，由于我们进行未知类激活，Outlier 新语义类明显不再混杂在已知类的支撑当中，而是形成中心簇，但由于新语义类其实并不是一个未知类，如 SVHN-C 中是十个数字类，所以该类簇并不紧簇。

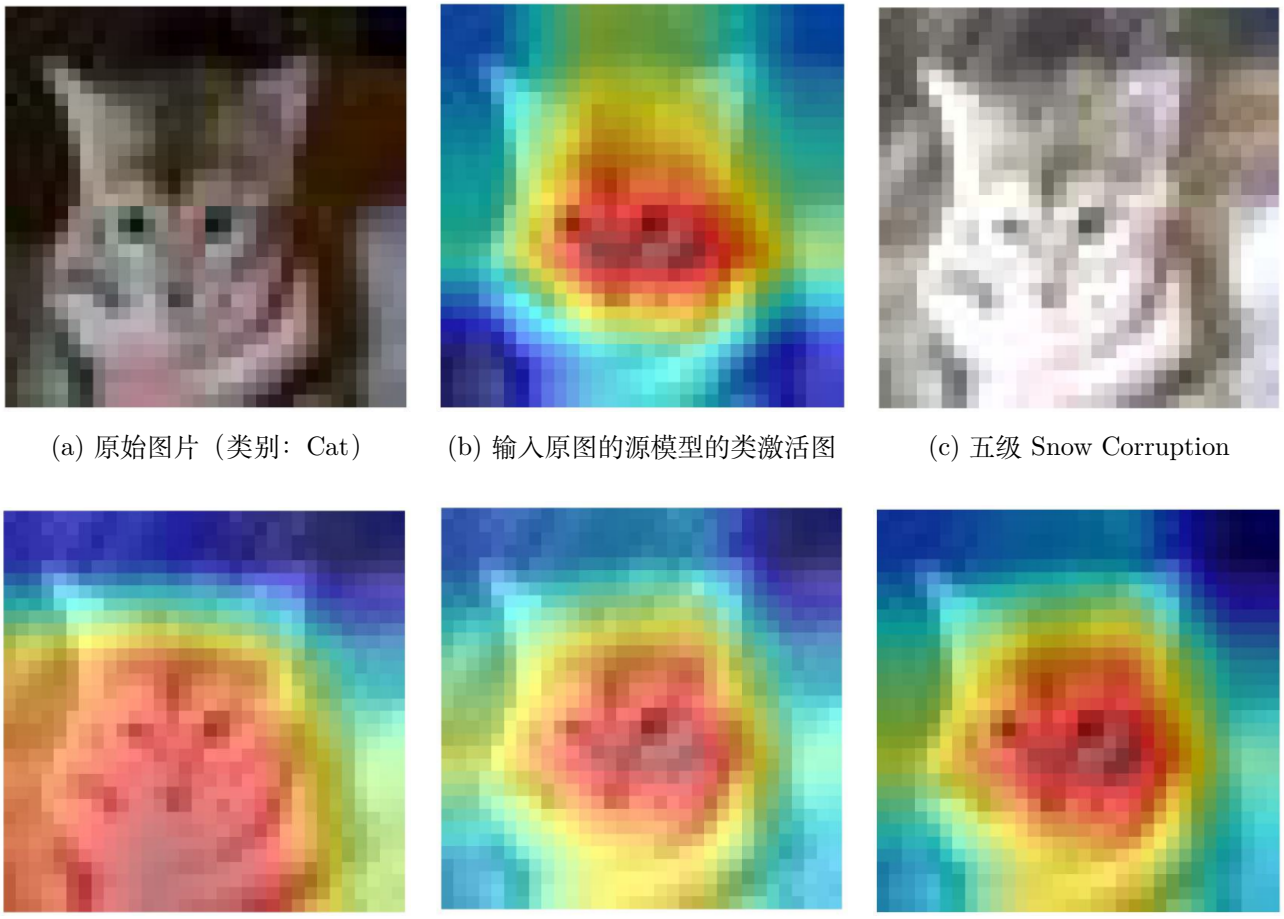
我们进一步分析另一种 Brightness Corruption 下基于熵阈值的 OOD 检测场景下 Tent 和 STAMP 的降维结果差异，如下图所示。可以发现 OOD 样本在 STAMP 比在 Tent 中有更加“离群”的分布状态。OOD 样本有可能更多地聚集在 ID 类簇之间的空隙或边界地带，不再那么深入地混入各 ID 类簇中。这可能体现出 STAMP 对特征空间的重构，使得 OOD 样本相对更容易从 ID 分布中被“挤出”到相对独特的区域



(a) t-SNE for Tent in Brightness Corruption

(b) t-SNE for STAMP in Brightness Corruption

图 13. STAMP 与 Tent 的特征空间重构对比



(a) 原始图片（类别：Cat）

(b) 输入原图的源模型的类激活图

(c) 五级 Snow Corruption

(d) 五级 Snow Corruption 的误分类 CAM

(e) snow 传入 STAMP 模型的类激活图

(f) 传入 STAMP 模型的类激活图

图 14. Grad-CAM 类激活映射图

我们还使用 Grad-CAM 类激活映射对测试时适应算法进行可解释性分析。我们首先比较对于将最严重等级的 Snow Corruption 前后的图片输入源域模型的类激活区域差异来阐述测

试时适应的必要性。进一步阐述将此 Corruption 前后的图片输入 STAMP 模型适应后得到的模型的类激活区域差异来比对模型的适应性能。

源模型对原图正确分类，并且激活区域主要集中在图像中猫的脸部。这表明模型在分类时关注到了关键特征区域，如猫的眼睛和嘴巴等。但当对原图进行合成扰动操作后，源模型无法正确分类，错误地将图像分类为 “Ship”，并且激活区域分散，缺乏明确的焦点。模型可能因为图像的高扰动级别，无法捕捉到原始的关键特征区域，导致分类错误。使用 STAMP 适应后的模型对域偏移图像进行预测又能成功分类并使其激活区域重新聚焦在猫的脸部特征上，展现出 STAMP 对关键特征区域的稳健性识别。

6 总结与展望

本文首先概述了分布偏移并介绍了解决分布偏移的测试时适应方法以及相关工作。然后介绍了 ECCV2024 文章 STAMP 的算法流程，并阐述复现流程。并实现了构建具有未知类平滑激活的分类器得到 OTDG 场景下的源模型、实现基于原型的未知类激活改进算法 PUA、实现基于 STAMP 的改进算法 STAMP-UA 三个创新点，最后增加了 t-SNE 和 Grad-CAM 的可视化以及可解释性分析。下一步可以将具有新语义类的测试时适应场景拓展到语义分割任务上。

参考文献

[1] Chaoqi Chen, Luyao Tang, Yue Huang, Xiaoguang Han, and Yizhou Yu. Coda: Generalizing to open and unseen domains with compaction and disambiguation. In *Neural Information Processing Systems*, 2023.

[2] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *ArXiv*, abs/2010.01412, 2020.

[3] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *ArXiv*, abs/2302.12400, 2023.

[4] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128:336 – 359, 2016.

[5] Yu Sun, X. Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, 2019.

[6] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[7] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

[8] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7191–7201, 2022.

[9] Yongcan Yu, Lijun Sheng, Ran He, and Jian Liang. Stamp: Outlier-aware test-time adaptation with stable memory replay. In *European Conference on Computer Vision*, 2024.