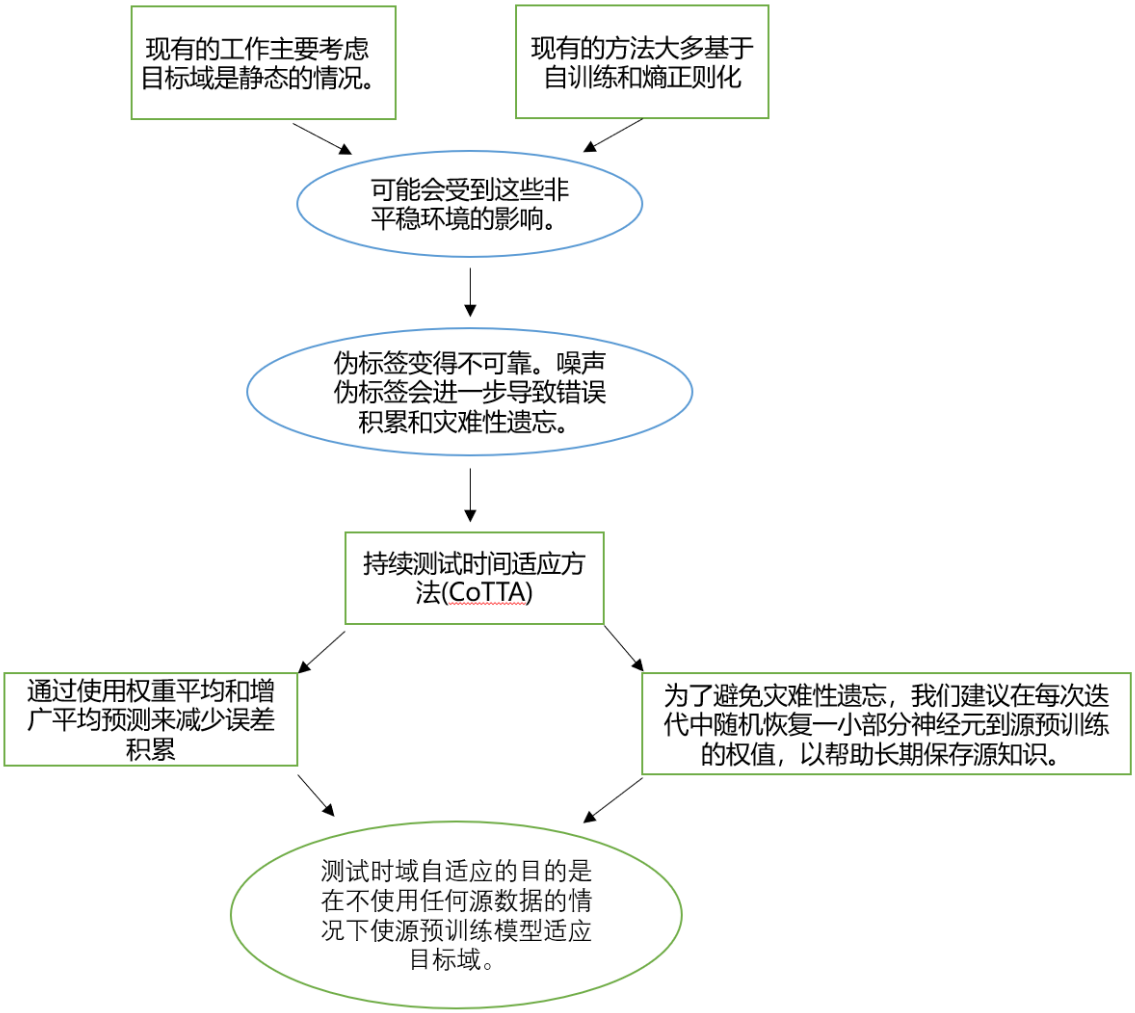
Continual Test-Time Domain Adaptation

https://qin.ee/cotta.

摘要

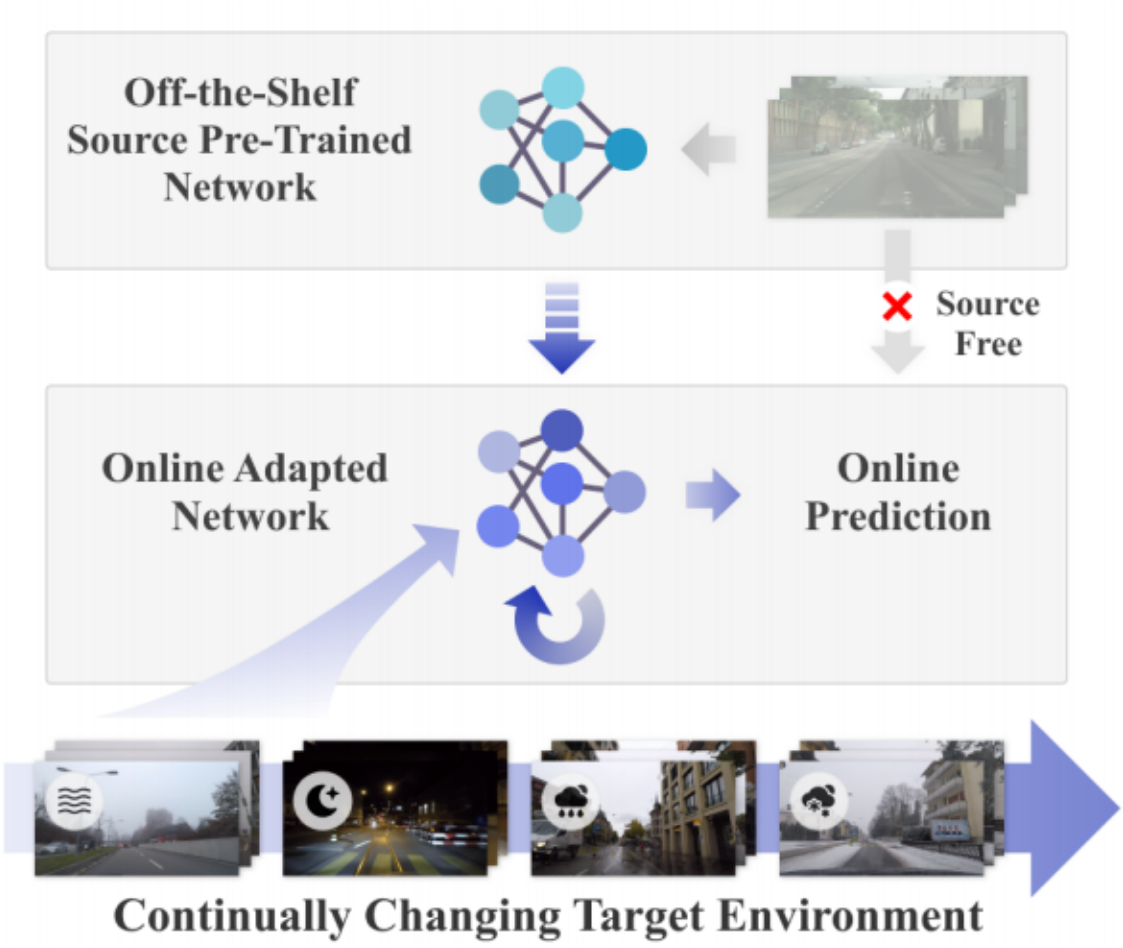


造成这一现象的**原因**有两个方面:首先，在不断变化的环境下，由于分布的移位，伪标签变得更杂讯和误校准[13]。因此，早期的预测错误更容易导致误差积累[4]。其次，由于模型在很长一段时间内不断适应新的分布，来自源领域的知识很难保存。

由于平均教师预测通常比标准模型具有更高的质量[55]，我们使用**加权平均教师模型来提供更准确的预测**。另一方面，**对于具有较大域间隙的测试数据，我们使用增强平均预测**来进一步提高伪标签的质量。

Introduction

测试-时域自适应的目的是通过在推理期间学习未标记的测试(目标)数据来适应源预训练模型。



目标数据是按顺序提供的，并且来自不断变化的环境。**使用现成的源预训练网络来初始化目标网络。**该模型基于当前目标数据在线更新，并以在线方式给出预测结果。目标网络的自适应不依赖于任何源数据。

预测和更新是在线执行的，这意味着模型只能访问当前的数据流，而不能访问完整的测试数据或任何源数据。所提出的设置与现实世界的机器感知系统非常相关。例如，对于自动驾驶系统来说，周围环境是不断变化的(例如，天气从晴天变为多云，然后变为雨天)。它们甚至可以突然改变(例如，当一辆汽车驶出隧道时，相机突然过度曝光)。感知模型需要在这些非平稳的域位移下进行自我调整和在线决策。

值得指出的是，我们的方法很容易实现。权重加增平均策略和随机恢复可以很容易地结合到任何现成的预训练模型中，而无需在源数据上重新训练它。

推理期间指的是模型进行预测或生成输出时的阶段。在无监督域适应中，源数据是可以在推理期间使用的，因为模型在学习阶段可以完全访问和利用源数据。然而，在隐私问题或法律限制下，源数据通常在推理期间不可用，意味着模型不能直接访问或使用源数据来进行预测或生成输出。

本方法

问题定义

我们在表1中列出了在线连续测试时间适应设置与现有适应设置之间的主要区别。

Table 1. The difference between our proposed continual test-time adaptation and related adaptation settings.

	Data		Learning	
Setting	Source	Target	Train stage	Test stage
standard domain adaptation	Yes	stationary	Yes	No
standard test-time training [54]	Yes	stationary	Yes (aux task)	Yes
fully test-time adaptation [61]	No	stationary	No (pre-trained)	Yes
continual test-time adaptation	No	continually changing	No (pre-trained)	Yes

在源数据(X^S, Y^S)**上训练参数为**θ的已有预训练模型$f_{\theta_0}(x)$，

未标记的目标域数据X^T按顺序提供，模型只能访问当前时间步长的数据。在时间步长t处，提供目标数据X^T_t作为输入，模型f_{θ_t}需要做出预测$f_{\theta_t}(X^T_t)$，根据未来输入$\theta_t \longrightarrow \theta_{t+1}$进行自我调整。$X^T_t$的数据分布是不断变化的。基于在线预测对模型进行评估。

方法论

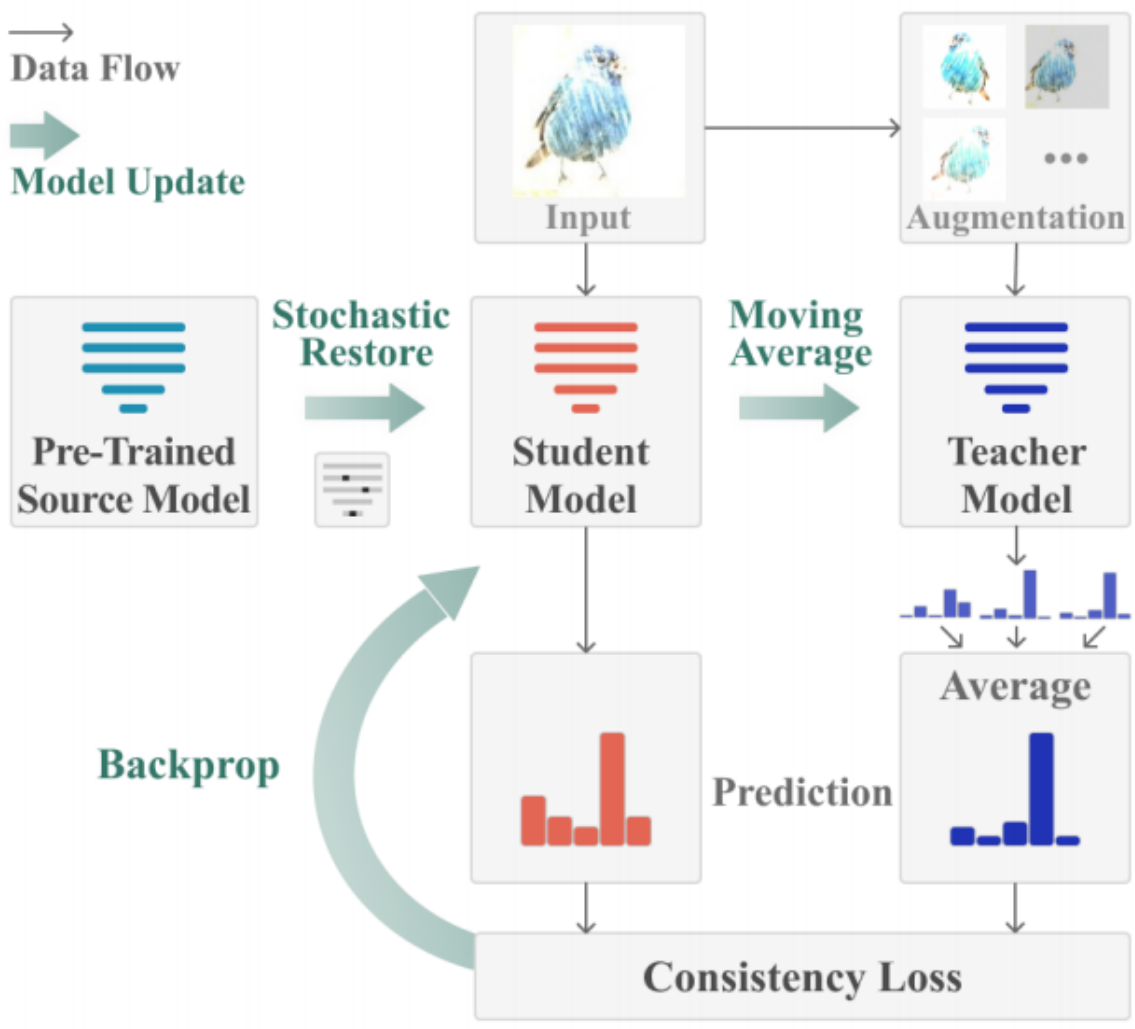
Algorithm 1 The proposed continual test-time adaptation

Initialization: A source pre-trained model $f_{\theta_0}(x)$, teacher model $f_{\theta'_0}(x)$ initialized from $f_{\theta_0}(x)$.

Input: For each time step t, current stream of data x_t.

1: Augment x_t and get weight and augmentation-averaged pseudo-labels from the teacher $f_{\theta'_t}$ by Equation 4.
2: Update student f_{θ_t} by consistency loss in Equation 5.
3: Update teacher $f_{\theta'_t}$ by moving average in Equation 2.
4: Stochastically restore student f_{θ_t} by Equation 8.

Output: Prediction $f_{\theta'_t}(x_t)$; Updated student model $f_{\theta_{t+1}}(x)$; Updated teacher model $f_{\theta'_{t+1}}(x)$.



源模型：这需要对源数据进行重新训练，并且不可能重用现有的预训练模型。在我们提出的测试时间适应方法中，我们解除了这个负担，并且不需要修改体系结构或额外的源培训过程。因此，任何现有的预训练模型都可以使用，而无需对源进行再训练。

权重平均伪标签：给定目标数据X^T_t和模型f_{θ_t}，自训练框架下常见的测试时间目标是最小化预测$\hat{y}^T_t = f_{\theta_t}(X^T_t)$与伪标签之间的交叉熵一致性。

我们使用**加权平均教师模型$f_{\theta'}$来生成伪标签**。在时间步长t = 0时，将教师网络初始化为与源预训练网络相同。在时间步长t处，伪标签首先由教师生成$\hat{y}'^T_t = f_{\theta'}(X^T_t)$

学生f_{θ_t}通过学生和教师预测之间的交叉熵损失来更新:

$$\mathcal{L}_{\theta_t}(x^T_t) = -\sum_c \hat{y}'^T_{tc} \log \hat{y}^T_{tc}, \tag{1}$$

其中，$\hat{y}'^T_{tc}$为教师模型软伪标签预测中c类出现的概率，$\hat{y}^T_{tc}$为主模型(学生)的预测。这种损失加强了教师和学生预测之间的一致性。

在使用公式1更新学生模型$\theta_t \rightarrow \theta_{t+1}$后，我们使用学生权重通过指数移动平均更新教师模型的权重:

$$\theta'_{t+1} = \alpha\theta'_t + (1-\alpha)\theta_{t+1}, \tag{2}$$

其中α是平滑因子。我们对输入数据X^T_t的最终预测是$\hat{y}'^T_t$中概率最高的类别。

增广平均伪标签：在不断变化的环境下，测试分布可能发生巨大变化，这可能使增强策略无效。在这里，我们考虑了测试时间的域移，并通过预测置信度近似域差。仅在域差较大时进行增广，以减少误差累积。

$$\tilde{y}'^T_t = \frac{1}{N}\sum_{i=0}^{N-1} f_{\theta'_t}(\text{aug}_i(x^T_t)), \tag{3}$$

$$y'^T_t = \begin{cases} \hat{y}'^T_t, & \text{if conf}(f_{\theta_0}(x^T_t)) \geq p_{th} \\ \tilde{y}'^T_t, & \text{otherwise,} \end{cases} \tag{4}$$

其中，$\tilde{y}'^T_t$是来自教师模型的增强平均预测，$\hat{y}'^T_t$是来自教师模型的直接预测，$\text{conf}(f_{\theta_0}(X^T_t))$是源预训练模型对当前输入$X^T_t$的预测置信度，$p_{th}$是置信度阈值。通过使用公式4中的预训练模型$f_{\theta_0}$计算当前输入$X^T_t$上的预测置信度，我们试图近似源和当前域之间的域差。

我们假设较低的置信度表示较大的领域差距，较高的置信度表示较小的领域差距。因此，当置信度较高且大于阈值时，我们直接使用$\hat{y}'^T_t$作为伪标签，不使用任何增值。当置信度较低时，我们额外应用N个随机增广来进一步提高伪标签质量。

在具有小域间隙的自信样本上随机扩增，有时会降低模型的性能。所以过滤至关重要。

通过改进的伪标签更新学生:

$$\mathcal{L}_{\theta_t}(x_t^T) = -\sum_c {y'}_{tc}^T \log \hat{y}_{tc}^T, \tag{5}$$

随机修复：长期自我训练的持续适应不可避免地会引入错误并导致遗忘。强烈的分布移位会导致校准错误甚至错误的预测。在这种情况下，自我训练可能只会强化错误的预测。即使新数据没有严重偏移，模型也可能因为不断的适应而无法恢复。

考虑在时间步长为t时，基于方程1的**梯度更新**后的学生模型f_θ内的卷积层:

$$x_{l+1} = W_{t+1} * x_l, \tag{6}$$

其中*表示卷积运算，x_l和x_{l+1}表示该层的输入和输出，W_{t+1}表示**可训练的卷积滤波器。**

本文提出的随机恢复方法通过以下方式对权值W进行更新:

$$M \sim \text{Bernoulli}(p), \tag{7}$$

$$W_{t+1} = M \odot W_0 + (1 - M) \odot W_{t+1}, \tag{8}$$

其中⊙表示逐元素的乘法。p是一个小的恢复概率，M是与W_{t+1}形状相同的掩模张量，Bernoulli表示伯努利分布。掩码张量决定W_{t+1}中的哪个元素要恢复到源权重W_0。如果M中对应位置的值是1，那么Wt+1中的对应元素将从源权重W0中恢复；如果M中对应位置的值是0，那么Wt+1中的对应元素将保持不变。

通过随机地将可训练权值中的少量张量元素恢复到初始权值，避免了网络偏离初始源模型太远，从而避免了灾难性遗忘。此外，通过保留源模型的信息，我们能够训练所有可训练的参数，而不会遭受模型崩溃的痛苦。

实验

我们在五个连续的测试时间适应基准任务上评估了我们提出的方法:cifar10到cifar10c(标准和渐进)，cifar100到cifar100c, imagenet到imagenet - c的图像分类，以及cityscapes到acdc的语义分割。

我的实验相关

cityscape -to- acdc是一个连续的语义分割任务，我们设计它来模拟现实世界中的连续分布变化。源模型是在cityscape数据集上训练的现成的预训练分割模型[7]。目标域包含来自不利条件数据集(ACDC)的各种场景的图像[50]。ACDC数据集与cityscape共享相同的语义类，并在四种不同的不利视觉条件下收集:雾、夜、雨和雪。我们按照相同的默认顺序评估连续的测试时间适应性。我们使用来自每个不利条件的400张未标记的图像进行适应。

为了模拟现实生活中可能会重新访问的类似环境的场景，并评估我们的方法的遗忘效果，**我们将同一序列组(四种条件)重复10次(即总共40次:雾-→夜-→雨-→雪-→雾...)。**这也为长期适应性能的评估提供了依据。

对于实现细节，我们在cityscape -to - acdc实验中采用了基于变压器的架构Segformer[64]。我们使用公开可用的预先训练过的Segformer-B5作为我们的现成资源模型。对于基线比较方法，TENT优化了归一化层中的参数。对于提出的CoTTA模型，所有可训练的层都被更新，而不需要选择特定的层。来自ACDC的图像分辨率为1920x1080。我们使用960x540的下采样分辨率作为网络的输入，并在原始分辨率下评估预测结果。Adam优化器的学习率比Segformer的默认学习率小8倍，因为我们在在线连续测试时间适应实验中使用批大小为1而不是8(源训练的默认值)。

我们使用带有翻转的多尺度输入作为所提出方法的增强方法来生成增强加权伪标签(如公式3所示)。遵循MMSeg中cityscape的默认做法[6]，我们使用尺度因子[0.5,0.75,1.0,1.25,1.5,1.75,2.0]。

数据集

最初创建CIFAR10C、CIFAR100C和ImageNet-C是为了对分类网络的鲁棒性进行基准测试[15]。每个数据集包含15种严重程度为5级的损坏类型。对于CIFAR10C和CIFAR100C数据集，每种损坏类型都有10,000张图像。

对于我们的在线连续测试时间自适应任务，使用在CIFAR10或CIFAR100数据集的干净训练集上预训练的网络。

对于**CIFAR10到cifar10c**，我们遵循TENT[61]对CIFAR10实验的官方公开实现。采用了相同的预训练模型，即来自**RobustBench基准测试[8]的WideResNet-28[71]模型**。我们在每次迭代中为一个步骤更新模型(即每个测试点一个梯度步骤)。我们使用与官方实现相同的**学习率为1e-3**的Adam优化器。在[5]之后，我们使用了相同的随机增强组合，包括颜色抖动、随机仿射、高斯模糊、随机水平翻转和高斯噪声。我们在实验中使用**32个增广**。我们在补充材料中讨论了增广阈值的选择。与TENT模型只更新BN尺度和转移权重不同，我们在实验中更新了所有可训练的参数。我们对所有实验使用**p = 0.01的恢复概率**。

对于**CIFAR100到cifar100c**的实验，我们采用了**来自[16]的预训练的ResNeXt-29[65]模型**，该模型在RobustBench基准测试中被用作CIFAR100的默认架构之一[8]。**使用与CIFAR10实验相同的超参数**。**ImageNet-toImageNet-C[15]实验**使用了RobustBench[8]中标准的预训练resnet50模型。在十种不同的损坏顺序下对ImageNet-C实验进行了评估。

1.配置环境

environment.yml里面的括号后面的编号删除。

`RobustBench`手动安装。

segment也得下载和配置环境（# Example logs are included in ./example_logs/base.log, tent.log, and cotta.log.）。

模型动物园：

由于 ImageNet 数据集的许可，应手动下载：

- ImageNet：在此处获取下载链接（只需从学术电子邮件注册，那里的审批系统是自动的，并且立即发生），然后按照 此处提取验证的说明 以与 PyTorch 兼容的格式设置为文件夹 。`val`
- ImageNet-C：请访问此处获取说明。
- ImageNet-3DCC：使用提供的工具从此处下载数据。数据将保存到名为 的文件夹中。`ImageNet-3DCC`

要使用模型库中的模型，您可以在下表中找到所有可用的模型 ID。

download.sh：下载并解压

2.内容

目的：

- CoTTA
- AdaBN / BN Adapt
- TENT

在

- CIFAR10/100 -> CIFAR10C/100C (standard/gradual)
- ImageNet -> ImageNetC
- Cityscapes -> ACDC (segmentation)

上面比较

代码

robustbench：标准化的对抗性稳健性基准，对标auto Attack，使用攻击来检测。

data

jpeg格式的图片和对应的标签

leader board

输出的结果是一个包含排行榜信息的 HTML 表格的字符串。

model zoo

调用模型，直接从命令行从模型库重现模型评估。

文件类型

运行.sh：配置环境、调用

.yaml:超参

.py:结构体和函数

我的实验

DA里，源域cityscapes，目标域Zurich，这个在.yaml文件中修改。

预训练模型（骨干网络）：refinenet