Protopia 图像转换的负责任的 AI

NetApp artificial intelligence solutions

NetApp
February 12, 2026

目录

Protopia 图像转换的负责任的 AI

TR-4928：负责任的 AI 和机密推理 - NetApp AI 与 Protopia 图像和数据转换

Sathish Thyagarajan、Michael Oglesby、 NetApp Byung Hoon Ahn、Jennifer Cwagenberg、Protopia

随着图像捕捉和图像处理技术的出现，视觉解读已经成为交流不可或缺的一部分。数字图像处理中的人工智能 (AI) 带来了新的商业机会，例如在医疗领域用于癌症和其他疾病的识别、在地理空间可视化分析中用于研究环境危害、在模式识别中、在视频处理中用于打击犯罪等等。然而，这一机遇也伴随着非凡的责任。

组织交给人工智能的决策越多，他们承担的与数据隐私和安全以及法律、道德和监管问题相关的风险就越大。负责任的人工智能使公司和政府组织能够建立信任和治理的实践，这对于大型企业大规模应用人工智能至关重要。本文档介绍了NetApp在三种不同场景下验证的 AI 推理解决方案，该解决方案使用NetApp数据管理技术与 Protopia 数据混淆软件来私有化敏感数据并降低风险和道德问题。

消费者和商业实体每天都会使用各种数字设备生成数百万张图像。随之而来的数据和计算工作量的激增使得企业转向云计算平台来实现规模和效率。同时，随着图像数据转移到公共云，人们对其所含敏感信息的隐私问题也产生了担忧。缺乏安全和隐私保障成为图像处理人工智能系统部署的主要障碍。

此外，还有 "删除权"根据 GDPR，个人有权要求组织删除其所有个人数据。还有 "隐私法"，该法案制定了公平信息实践准则。根据 GDPR，照片等数字图像可以构成个人数据，GDPR 规定了数据的收集、处理和删除方式。不这样做就是不遵守 GDPR，这可能会导致违反合规性的巨额罚款，这可能会对组织造成严重损害。隐私原则是实施负责任的人工智能的支柱之一，它确保机器学习 (ML) 和深度学习 (DL) 模型预测的公平性，并降低违反隐私或法规遵从性相关的风险。

本文档描述了在三种不同场景下经过验证的设计解决方案，包括带有和不带有图像混淆，这些场景与保护隐私和部署负责任的 AI 解决方案有关：

- 场景 1. Jupyter 笔记本中的按需推理。
- 场景 2.在 Kubernetes 上进行批量推理。
- 场景 3. NVIDIA Triton 推理服务器。

对于该解决方案，我们使用人脸检测数据集和基准（FDDB），这是一个为研究无约束人脸检测问题而设计的人脸区域数据集，结合 PyTorch 机器学习框架来实现 FaceBoxes。该数据集包含 2845 张不同分辨率图像中 5171 张人脸的注释。此外，本技术报告还介绍了从NetApp客户和现场工程师那里收集的一些适用于该解决方案的解决方案领域和相关用例。

目标受众

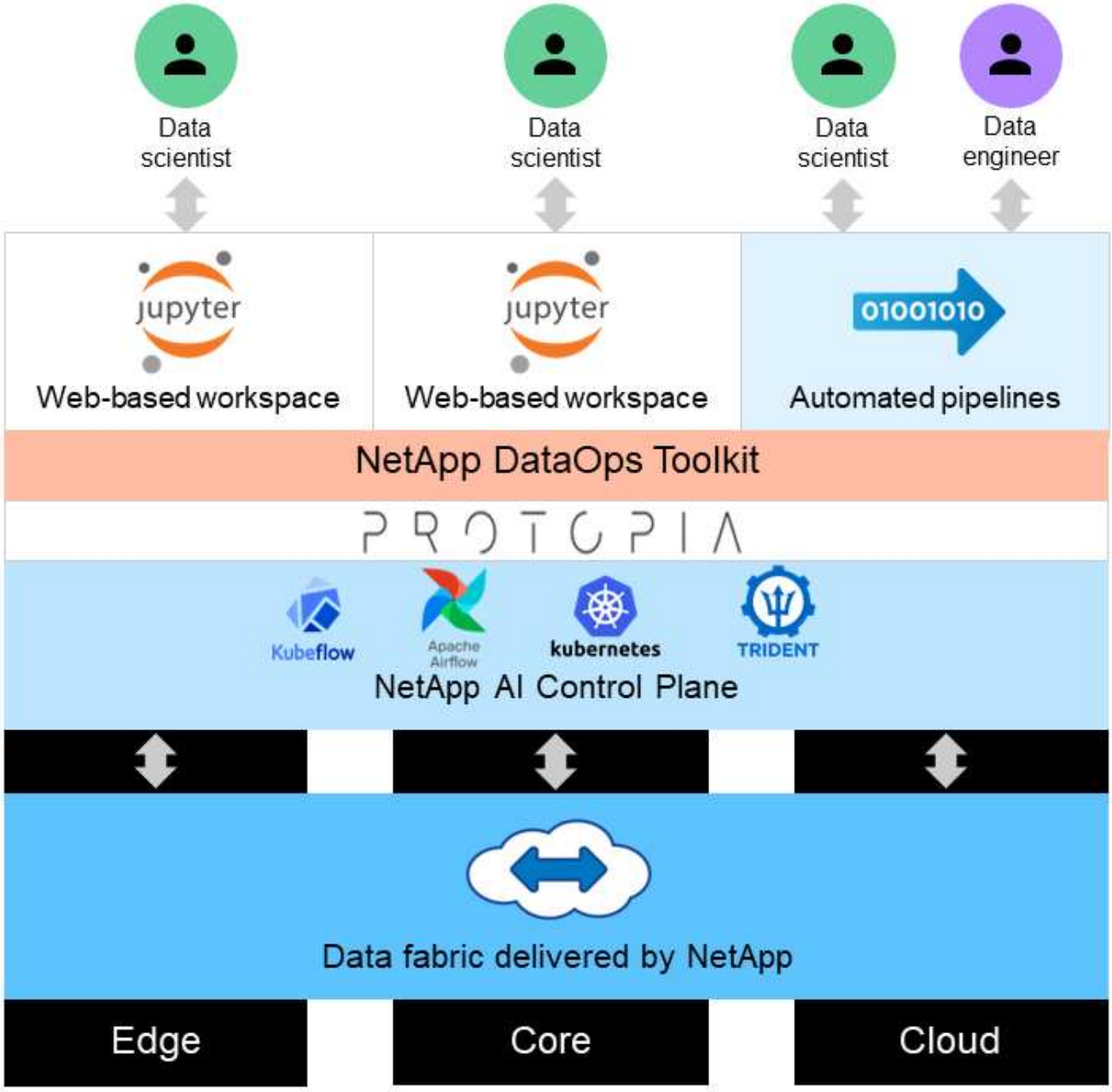
本技术报告面向以下受众：

- 希望设计和部署负责任的人工智能并解决公共场所面部图像处理的数据保护和隐私问题的商业领袖和企业架构师。
- 旨在保护和维护隐私的数据科学家、数据工程师、人工智能/机器学习 (ML) 研究人员以及人工智能/机器学习系统开发人员。

- 为符合 GDPR、CCPA 或国防部 (DoD) 和政府组织的隐私法等监管标准的 AI/ML 模型和应用程序设计数据混淆解决方案的企业架构师。
- 数据科学家和人工智能工程师正在寻找有效的方法来部署深度学习 (DL) 和 AI/ML/DL 推理模型来保护敏感信息。
- 负责边缘推理模型的部署和管理的边缘设备管理员和边缘服务器管理员。

解决方案架构

该解决方案旨在利用 GPU 和传统 CPU 的处理能力来处理大型数据集上的实时和批量推理 AI 工作负载。此验证证明了寻求负责任的 AI 部署的组织所需的 ML 隐私保护推理和最佳数据管理。该解决方案提供了一种适用于单节点或多节点 Kubernetes 平台的架构，用于边缘和云计算，并通过 Jupyter Lab 和 CLI 界面与核心本地的NetApp ONTAP AI、 NetApp DataOps Toolkit 和 Protopia 混淆软件互连。下图显示了由NetApp提供支持、采用 DataOps Toolkit 和 Protopia 的数据结构的逻辑架构概览。



Protopia 混淆软件在NetApp DataOps Toolkit 上无缝运行，并在离开存储服务器之前转换数据。

解决方案领域

数字图像处理具有许多优点，使许多组织能够充分利用与视觉表示相关的数据。 NetApp和 Protopia 解决方案提供了独特的 AI 推理设计，以在整个 ML/DL 生命周期中保护和私有化 AI/ML 数据。它使客户能够保留敏感数据的所有权，通过缓解与隐私相关的担忧，使用公共或混合云部署模型实现规模和效率，并在边缘部署人工智能推理。

环境情报

在环境危害领域，各行各业可以多种方式利用地理空间分析。政府和公共工程部门可以就公共卫生和天气状况获得可行的见解，以便在流行病或野火等自然灾害期间更好地为公众提供建议。例如，您可以在公共场所（例如机场或医院）识别 COVID 阳性患者，而不会损害受影响个人的隐私，并提醒相关部门和附近公众采取必要的安全措施。

边缘设备可穿戴设备

在军事和战场上，您可以使用边缘 AI 推理作为可穿戴设备来跟踪士兵的健康状况、监控驾驶员行为并向当局发出接近军用车辆的安全和相关风险警报，同时保护士兵的隐私。军队的未来将走向高科技，战场物联网 (IoBT) 和军事物联网 (IoMT) 将应用于可穿戴作战装备，帮助士兵通过使用快速边缘计算识别敌人并在战斗中表现得更好。保护和保存从无人机和可穿戴设备等边缘设备收集的视觉数据对于阻止黑客和敌人至关重要。

非战斗人员撤离行动

非战斗人员撤离行动 (NEO) 由国防部执行，旨在协助将生命受到威胁的美国公民和国民、国防部文职人员以及指定人员（东道国 (HN) 和第三国国民 (TCN)）撤离到适当的安全避难所。现有的行政控制措施主要采用手动的撤离人员筛选流程。然而，通过使用高度自动化的 AI/ML 工具结合 AI/ML 视频混淆技术，可以提高撤离人员识别、撤离人员跟踪和威胁筛查的准确性、安全性和速度。

医疗保健和生物医学研究

图像处理用于根据计算机断层扫描 (CT) 或磁共振成像 (MRI) 获得的 3D 图像诊断手术规划的病理。 HIPAA 隐私规则规定了组织如何收集、处理和删除所有个人信息和照片等数字图像。根据 HIPAA 安全港规定，要使数据符合可共享条件，必须删除正面照片和任何类似图像。用于从结构 CT/MR 图像中隐藏个人面部特征的去识别或颅骨剥离算法等自动化技术已成为生物医学研究机构数据共享过程的重要组成部分。

AI/ML分析的云迁移

企业客户传统上在本地训练和部署 AI/ML 模型。出于规模经济和效率的原因，这些客户正在扩展以将 AI/ML 功能转移到公共、混合或多云部署中。然而，它们受到可以向其他基础设施公开的数据的限制。 NetApp解决方案可应对各种网络安全威胁， "数据保护"和安全评估，并与 Protopia 数据转换相结合，最大限度地降低将图像处理 AI/ML 工作负载迁移到云端所带来的风险。

有关其他行业的边缘计算和 AI 推理的更多用例，请参阅"TR-4886 边缘人工智能推理"以及NetApp AI 博客， "情报与隐私" 。

技术概述

本节概述了完成此解决方案所需的各种技术组件。

普罗托邦

Protopia AI 为当今市场上的机密推理提供了一种不引人注目的纯软件解决方案。 Protopia 解决方案通过最大限度地减少敏感信息的暴露，为推理服务提供了无与伦比的保护。人工智能仅接收数据记录中对于执行手头任务真正必要的信息，仅此而已。大多数推理任务不会使用每个数据记录中存在的所有信息。无论您的 AI 使用的是图像、语音、视频还是结构化表格数据，Protopia 都只提供推理服务所需的内容。该专利核心技术使用数学策划的噪声来随机转换数据并混淆给定 ML 服务不需要的信息。该解决方案不会掩盖数据；相反，它通过使用精选的随机噪声来改变数据表示。

Protopia 解决方案将改变表示的问题表述为基于梯度的扰动最大化方法，该方法仍然保留与模型功能相关的输入特征空间中的信息。此发现过程在训练 ML 模型结束时作为微调过程运行。在传递过程自动生成一组概率分布之后，低开销数据转换会将这些分布中的噪声样本应用于数据，并在将其传递给模型进行推理之前对其进行混淆。

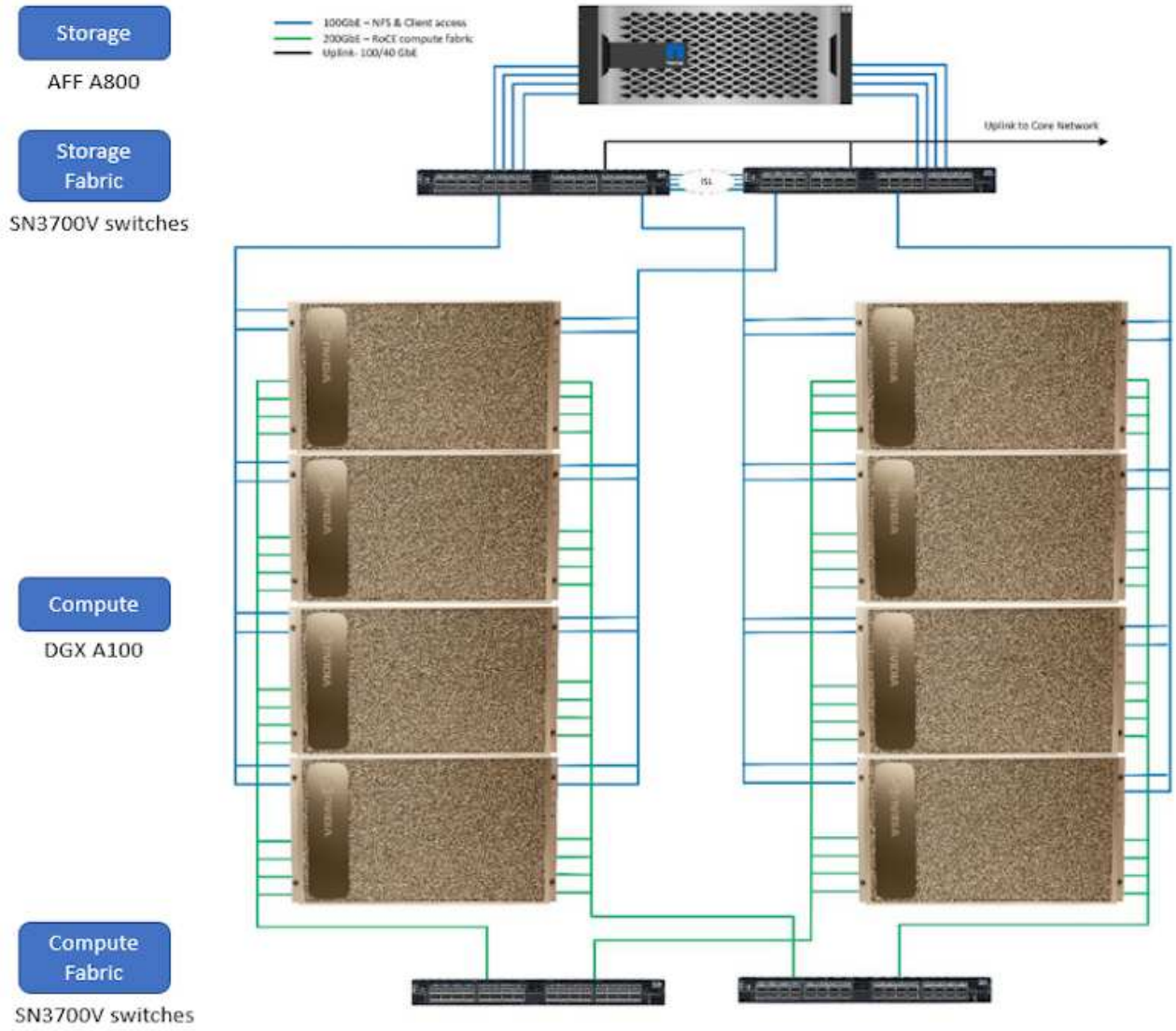
NetApp ONTAP AI

NetApp ONTAP AI 参考架构由 DGX A100 系统和NetApp云连接存储系统提供支持，由NetApp和NVIDIA开发和验证。它为 IT 组织提供了一种具有以下优势的架构：

- 消除设计复杂性
- 允许独立扩展计算和存储
- 支持客户从小规模起步，然后无缝扩展
- 提供广泛的存储选项，以满足各种性价比需求

ONTAP AI 将 DGX A100 系统和NetApp AFF A800存储系统与最先进的网络紧密集成。 ONTAP AI 通过消除设计复杂性和猜测来简化 AI 部署。客户可以从小规模开始，然后无中断地发展，同时智能地管理从边缘到核心到云端再返回的数据。

下图显示了采用 DGX A100 系统的ONTAP AI 系列解决方案的几种变体。 AFF A800系统性能已通过最多八个 DGX A100 系统验证。通过向ONTAP集群添加存储控制器对，该架构可以扩展到多个机架，以支持许多 DGX A100 系统和具有线性性能的 PB 级存储容量。这种方法可以灵活地根据所使用的 DL 模型的大小和所需的性能指标独立地改变计算与存储的比率。



有关ONTAP AI 的更多信息，请参阅 "NVA-1153：带有NVIDIA DGX A100 系统和 Mellanox Spectrum 以太网交换机的NetApp ONTAP AI。"

NetApp ONTAP

ONTAP 9.11 是NetApp最新一代存储管理软件，它支持企业实现基础架构现代化并过渡到云就绪数据中心。ONTAP利用业界领先的数据管理功能，只需一套工具即可管理和保护数据，无论数据位于何处。您还可以将数据自由移动到任何需要的地方：边缘、核心或云端。 ONTAP 9.11 包含许多功能，可简化数据管理、加速和保护关键数据，并支持跨混合云架构的下一代基础架构功能。

NetApp DataOps 工具包

NetApp DataOps Toolkit 是一个 Python 库，可帮助开发人员、数据科学家、DevOps 工程师和数据工程师轻松执行各种数据管理任务，例如近乎即时地配置新的数据卷或 JupyterLab 工作区、近乎即时地克隆数据卷或 JupyterLab 工作区，以及近乎即时地拍摄数据卷或 JupyterLab 工作区的快照以进行可追溯性或基准测试。这个 Python 库可以作为命令行实用程序或函数库，您可以将其导入到任何 Python 程序或 Jupyter 笔记本中。

NVIDIA Triton 推理服务器

NVIDIA Triton 推理服务器是一款开源推理服务软件，可帮助标准化模型部署和执行，以在生产中提供快速且可扩展的 AI。 Triton Inference Server 通过使团队能够在任何基于 GPU 或 CPU 的基础架构上从任何框架部署、运行和扩展经过训练的 AI 模型，简化了 AI 推理。 Triton Inference Server 支持所有主流框架，例如 TensorFlow、 NVIDIA TensorRT、PyTorch、MXNet、OpenVINO 等。 Triton 与 Kubernetes 集成，可进行编排和扩展，您可以在所有主要的公共云 AI 和 Kubernetes 平台中使用它。它还与许多 MLOps 软件解决方案集成。

PyTorch

"PyTorch"是一个开源的 ML 框架。它是一个针对使用 GPU 和 CPU 的深度学习而优化的张量库。 PyTorch 包包含多维张量的数据结构，它提供了许多实用程序，用于高效序列化张量以及其他有用的实用程序。它还有一个 CUDA 对应物，使您能够在具有计算能力的NVIDIA GPU 上运行张量计算。在本次验证中，我们使用 OpenCV-Python (cv2) 库来验证我们的模型，同时利用 Python 最直观的计算机视觉概念。

简化数据管理

数据管理对于企业 IT 运营和数据科学家至关重要，以便将适当的资源用于 AI 应用程序和训练 AI/ML 数据集。以下有关NetApp技术的附加信息超出了本次验证的范围，但可能与您的部署相关。

ONTAP数据管理软件包括以下功能，可简化操作并降低总运营成本：

- 内联数据压缩和扩展重复数据删除。数据压缩减少了存储块内部浪费的空间，重复数据删除显著增加了有效容量。这适用于本地存储的数据和分层到云的数据。
- 最小、最大和自适应服务质量 (AQoS)。细粒度的服务质量 (QoS) 控制有助于维持高度共享环境中关键应用程序的性能水平。
- NetApp FabricPool。提供冷数据自动分层到公共和私有云存储选项，包括 Amazon Web Services (AWS) 、Azure 和NetApp StorageGRID存储解决方案。有关FabricPool的更多信息，请参阅 "TR-4598：FabricPool最佳实践"。

加速并保护数据

ONTAP提供卓越级别的性能和数据保护，并通过以下方式扩展这些功能：

- 性能和更低的延迟。 ONTAP以尽可能低的延迟提供尽可能高的吞吐量。
- 数据保护。ONTAP提供内置数据保护功能，并在所有平台上提供通用管理。
- NetApp卷加密 (NVE)。 ONTAP提供原生卷级加密，同时支持板载和外部密钥管理。
- 多租户和多因素身份验证。 ONTAP支持以最高级别的安全性共享基础设施资源。

面向未来的基础设施

ONTAP具有以下功能，可帮助满足苛刻且不断变化的业务需求：

- 无缝扩展和无中断运行。 ONTAP支持无中断地向现有控制器和横向扩展集群添加容量。客户可以升级到最新技术，例如 NVMe 和 32Gb FC，而无需昂贵的数据迁移或中断。
- 云连接。 ONTAP是与云连接最紧密的存储管理软件，在所有公共云中均提供软件定义存储（ONTAP Select ）和云原生实例（Google Cloud NetApp Volumes）的选项。
- 与新兴应用程序的集成。 ONTAP使用支持现有企业应用的相同基础架构，为下一代平台和应用（如自动驾驶汽车、智能城市和工业 4.0）提供企业级数据服务。

NetApp Astra控制

NetApp Astra产品系列由NetApp存储和数据管理技术提供支持，为本地和公共云中的 Kubernetes 应用程序提供存储和应用程序感知数据管理服务。它使您能够轻松备份 Kubernetes 应用程序，将数据迁移到不同的集群，并立即创建可运行的应用程序克隆。如果您需要管理在公共云中运行的 Kubernetes 应用程序，请参阅 "Astra控制服务"。 Astra Control Service 是一项NetApp托管服务，可为 Google Kubernetes Engine (GKE) 和 Azure Kubernetes Service (AKS) 中的 Kubernetes 集群提供应用程序感知数据管理。

NetApp Trident

Astra "Trident"NetApp推出的一款适用于 Docker 和 Kubernetes 的开源动态存储编排器，可简化持久存储的创建、管理和使用。 Trident是一个 Kubernetes 原生应用程序，直接在 Kubernetes 集群中运行。 Trident使客户能够将 DL 容器映像无缝部署到NetApp存储上，并为 AI 容器部署提供企业级体验。 Kubernetes 用户（ML 开发人员、数据科学家等）可以创建、管理和自动化编排和克隆，以利用由NetApp技术提供支持的高级数据管理功能。

NetApp BlueXP复制和同步

"BlueXP复制和同步"是NetApp 的一项快速、安全的数据同步服务。无论您需要在本地 NFS 或 SMB 文件共享、NetApp StorageGRID、 NetApp ONTAP S3、 Google Cloud NetApp Volumes、 Azure NetApp Files、Amazon Simple Storage Service (Amazon S3)、Amazon Elastic File System (Amazon EFS)、Azure Blob、Google Cloud Storage 或 IBM Cloud Object Storage 之间传输文件， BlueXP Copy and Sync 都能快速安全地将文件移动到您需要的位置。数据传输完成后，可在源端和目标端完全使用。 BlueXP Copy 和 Syncc 根据您预先定义的计划持续同步数据，仅移动增量，从而最大限度地减少数据复制所花费的时间和金钱。 BlueXP Copy and Sync 是一种软件即服务 (SaaS) 工具，其设置和使用极其简单。 BlueXP Copy 和 Sync 触发的数据传输由数据代理执行。您可以在 AWS、Azure、Google Cloud Platform 或本地部署BlueXP Copy 和 Sync 数据代理。

NetApp BlueXP分类

在强大的AI算法驱动下， "NetApp BlueXP分类"为您的整个数据资产提供自动化控制和数据治理。您可以轻松找到节省成本的方法、识别合规性和隐私问题并找到优化机会。 BlueXP分类仪表板可让您洞察重复数据以消除冗余，映射个人、非个人和敏感数据，并针对敏感数据和异常情况发出警报。

测试和验证计划

对于此解决方案设计，验证了以下三种场景：

- 在 JupyterLab 工作区内，使用NetApp DataOps Toolkit for Kubernetes 进行编排的推理任务（有和没有 Protopia 混淆）。
- 在 Kubernetes 上执行批量推理作业（带和不带 Protopia 混淆），其数据卷是使用NetApp DataOps Toolkit for Kubernetes 进行编排的。
- 使用NVIDIA Triton 推理服务器实例的推理任务，该实例是通过使用NetApp DataOps Toolkit for Kubernetes 进行编排的。在调用 Triton 推理 API 之前，我们对图像应用了 Protopia 混淆，以模拟任何通过网络传输的数据都必须混淆的常见要求。此工作流程适用于在受信任区域内收集数据但必须传递到该受信任区域之外进行推理的用例。如果没有 Protopia 混淆，就不可能在敏感数据不离开受信任区域的情况下实现这种类型的工作流程。

测试配置

下表概述了解决方案设计验证环境。

组件	版本
Kubernetes	1.21.6
NetApp Trident CSI 驱动程序	22.01.0
适用于 Kubernetes 的NetApp DataOps 工具包	2.3.0
NVIDIA Triton 推理服务器	21.11-py3

测试程序

本节描述完成验证所需的任务。

前提条件

要执行本节中概述的任务，您必须能够访问安装并配置了以下工具的 Linux 或 macOS 主机：

- Kubectl（配置用于访问现有的 Kubernetes 集群）
  - 可以找到安装和配置说明 "此处"。
- 适用于 Kubernetes 的NetApp DataOps 工具包
  - 安装说明可以找到 "此处"。

场景 1 – JupyterLab 中的按需推理

1. 为 AI/ML 推理工作负载创建 Kubernetes 命名空间。

```
$ kubectl create namespace inference
namespace/inference created
```

2. 使用NetApp DataOps Toolkit 配置持久卷，用于存储您将执行推理的数据。

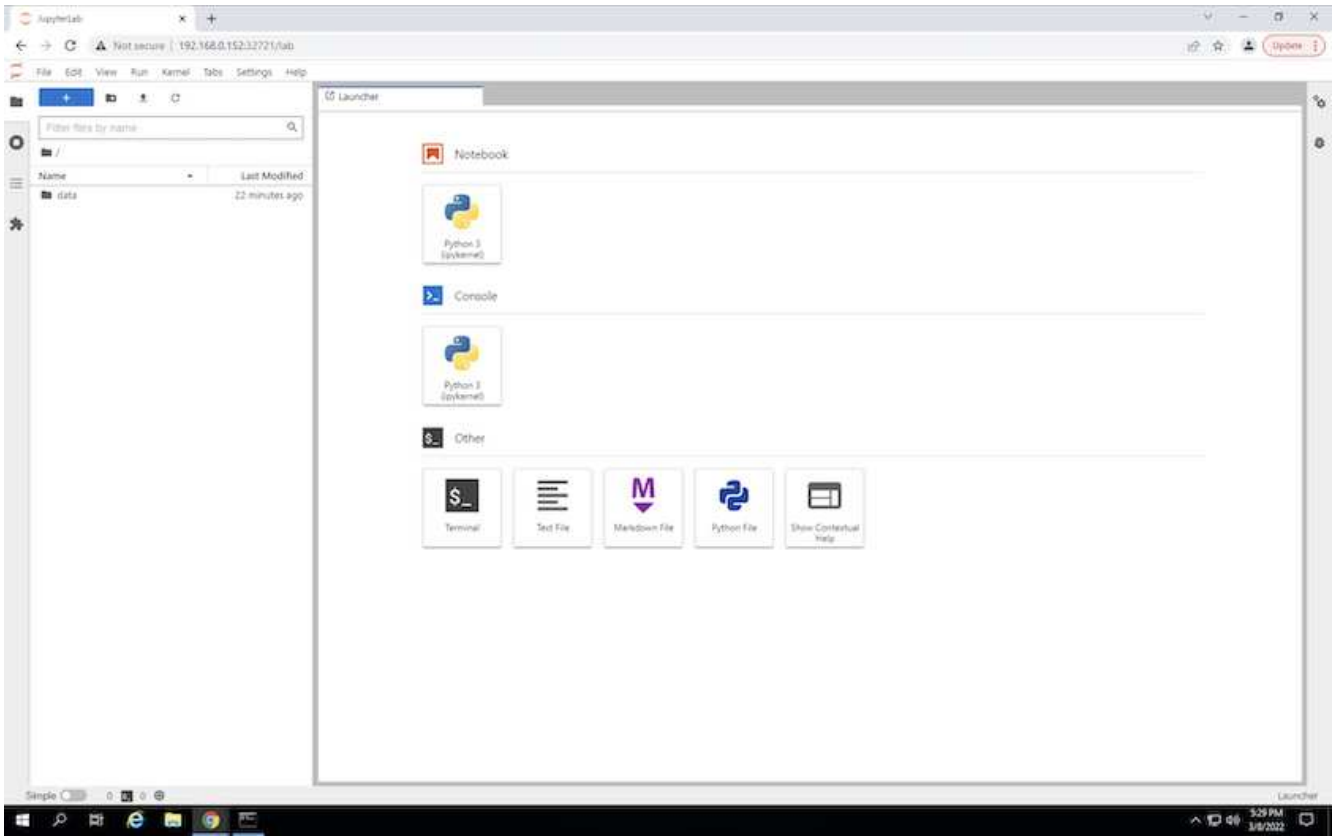
```
$ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc
-name=inference-data --size=50Gi
Creating PersistentVolumeClaim (PVC) 'inference-data' in namespace
'inference'.
PersistentVolumeClaim (PVC) 'inference-data' created. Waiting for
Kubernetes to bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC)
'inference-data' in namespace 'inference'.
```

3. 使用NetApp DataOps Toolkit 创建新的 JupyterLab 工作区。使用上一步创建的持久卷 `--mount- pvc`选项。根据需要NVIDIA `-- nvidia-gpu`选项。

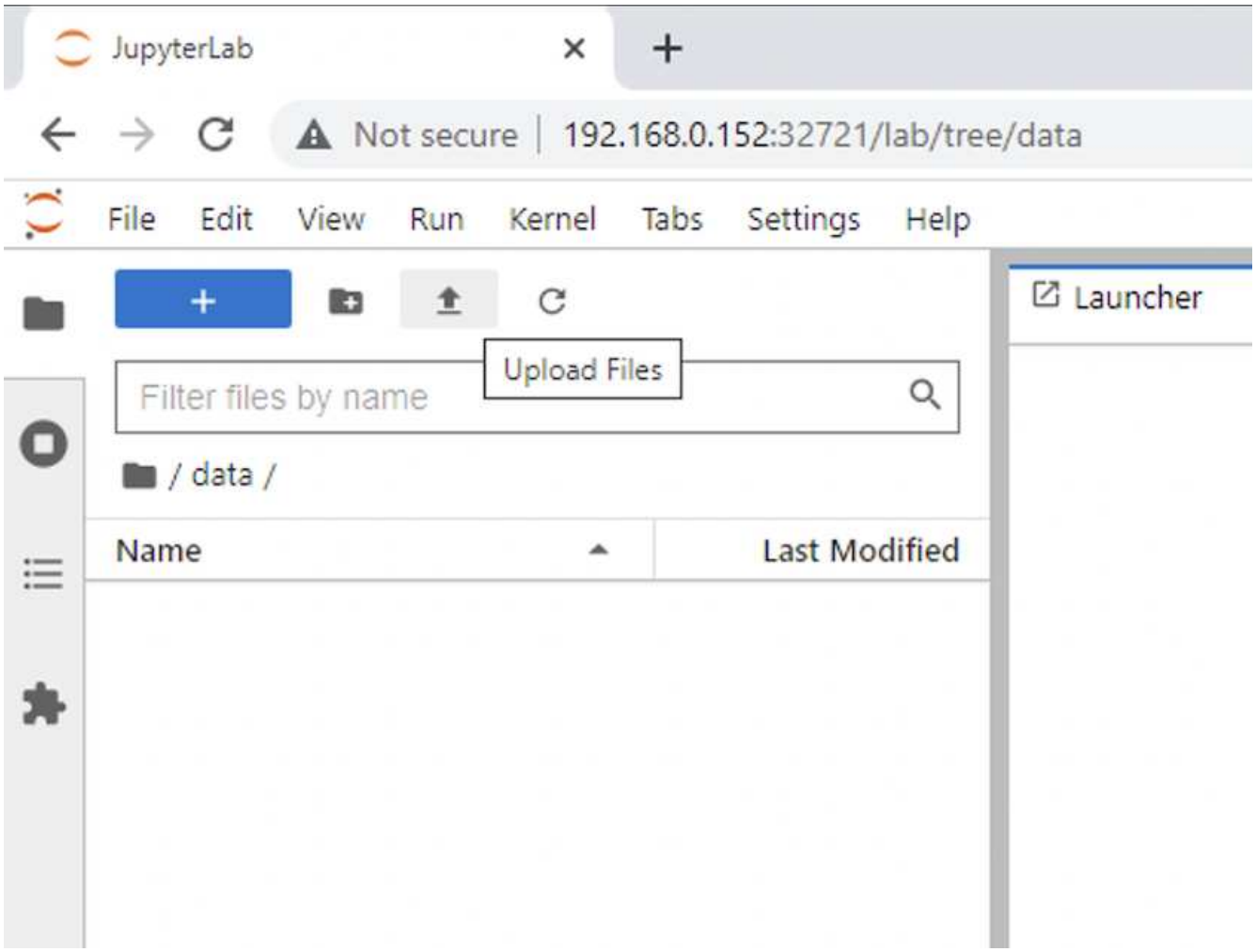
   在以下示例中，持久卷 `inference-data`被挂载到 `JupyterLab` 工作区容器中 `/home/jovyan/data`。使用官方 Project Jupyter 容器镜像时， `/home/jovyan`在 JupyterLab Web 界面中显示为顶级目录。

```
$ netapp_dataops_k8s_cli.py create jupyterlab --namespace=inference
--workspace-name=live-inference --size=50Gi --nvidia-gpu=2 --mount
-pvc=inference-data:/home/jovyan/data
Set workspace password (this password will be required in order to
access the workspace):
Re-enter password:
Creating persistent volume for workspace...
Creating PersistentVolumeClaim (PVC) 'ntap-dsutil-jupyterlab-live-
inference' in namespace 'inference'.
PersistentVolumeClaim (PVC) 'ntap-dsutil-jupyterlab-live-inference'
created. Waiting for Kubernetes to bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC)
'ntap-dsutil-jupyterlab-live-inference' in namespace 'inference'.
Creating Service 'ntap-dsutil-jupyterlab-live-inference' in namespace
'inference'.
Service successfully created.
Attaching Additional PVC: 'inference-data' at mount_path:
'/home/jovyan/data'.
Creating Deployment 'ntap-dsutil-jupyterlab-live-inference' in namespace
'inference'.
Deployment 'ntap-dsutil-jupyterlab-live-inference' created.
Waiting for Deployment 'ntap-dsutil-jupyterlab-live-inference' to reach
Ready state.
Deployment successfully created.
Workspace successfully created.
To access workspace, navigate to http://192.168.0.152:32721
```

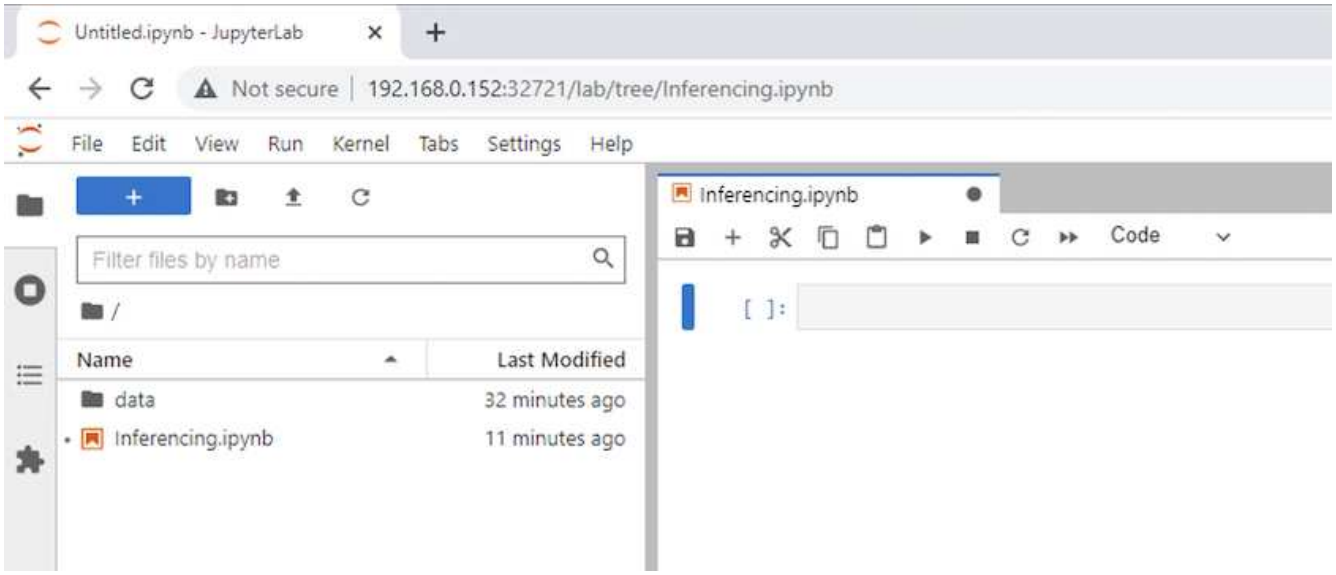
4. 使用输出中指定的 URL 访问 JupyterLab 工作区 `create jupyterlab`命令。数据目录代表挂载到工作区的持久卷。



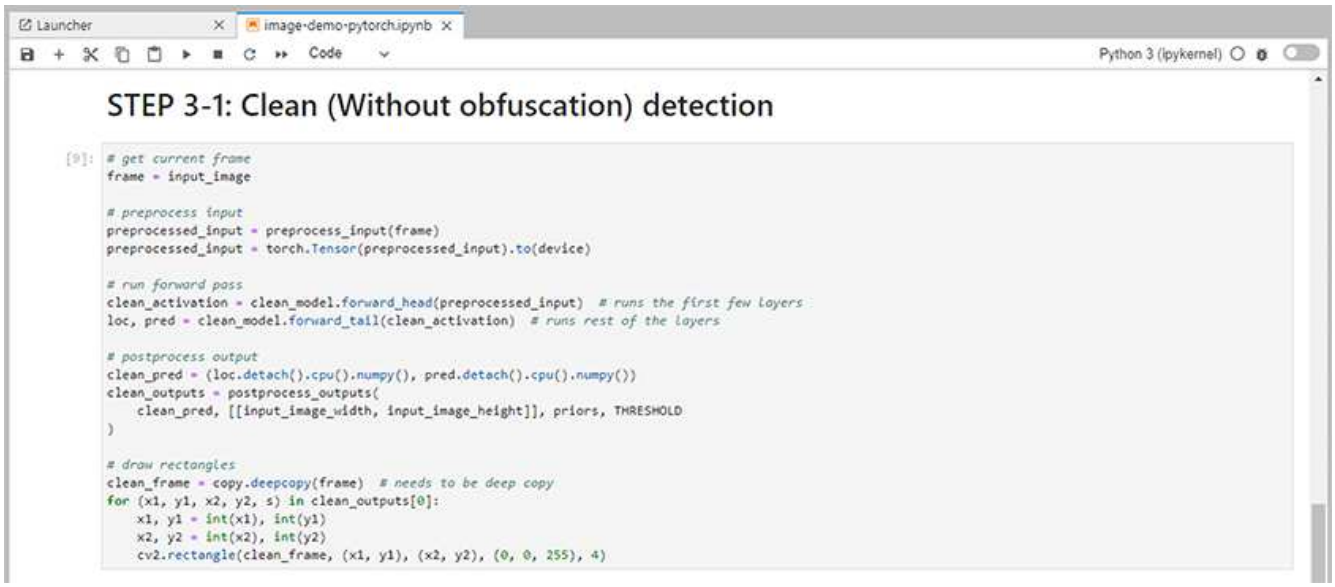
5. 打开 `data`目录并上传要执行推理的文件。当文件上传到数据目录时，它们会自动存储在挂载到工作区的持久卷上。要上传文件，请单击上传文件图标，如下图所示。

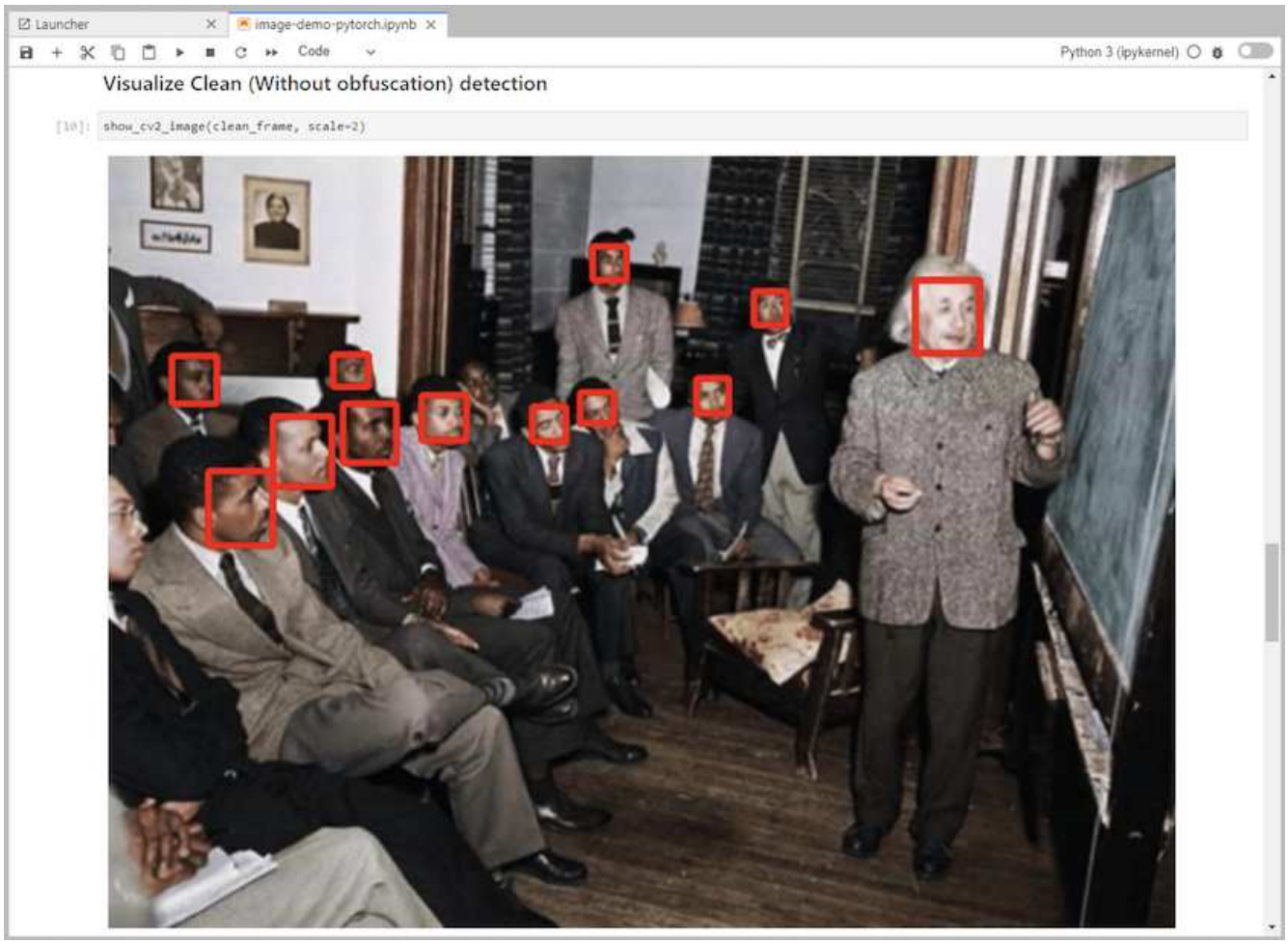


6. 返回顶级目录并创建一个新的笔记本。

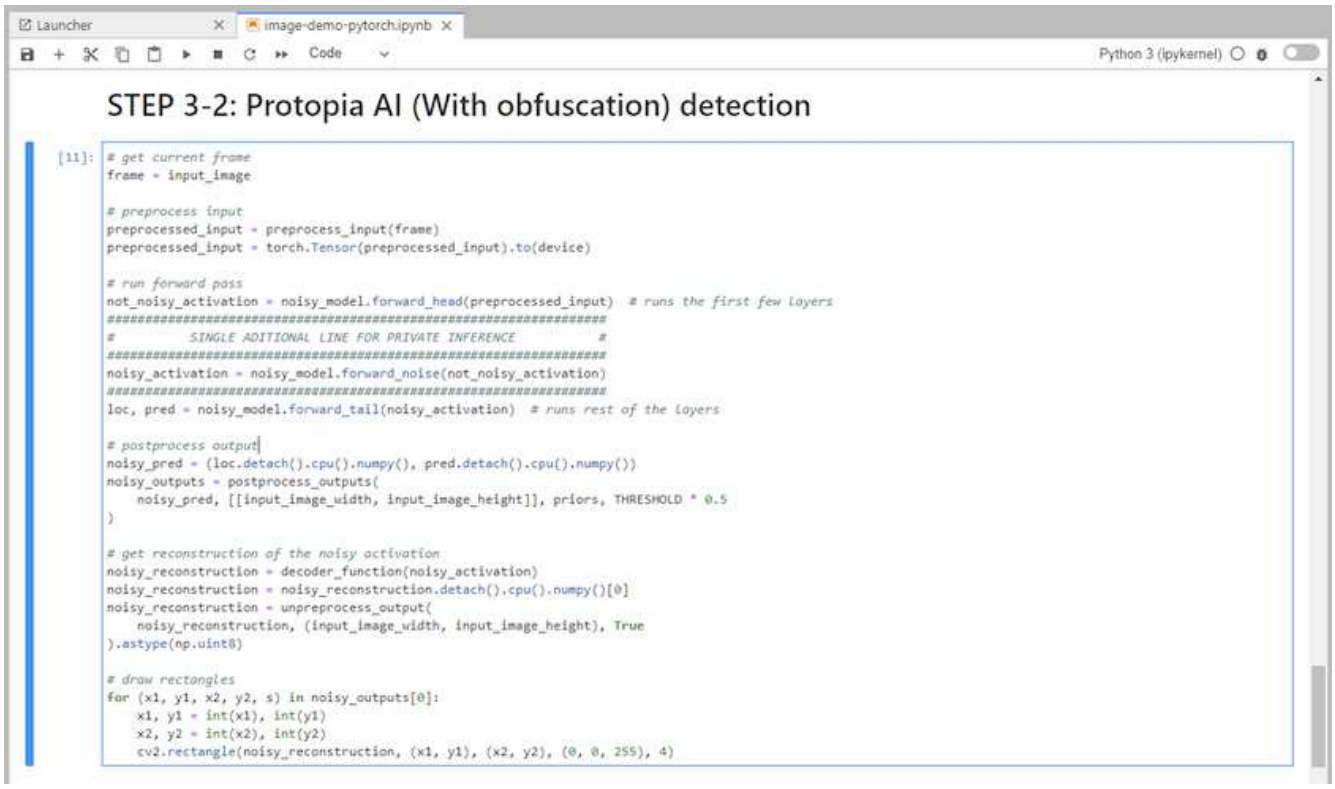


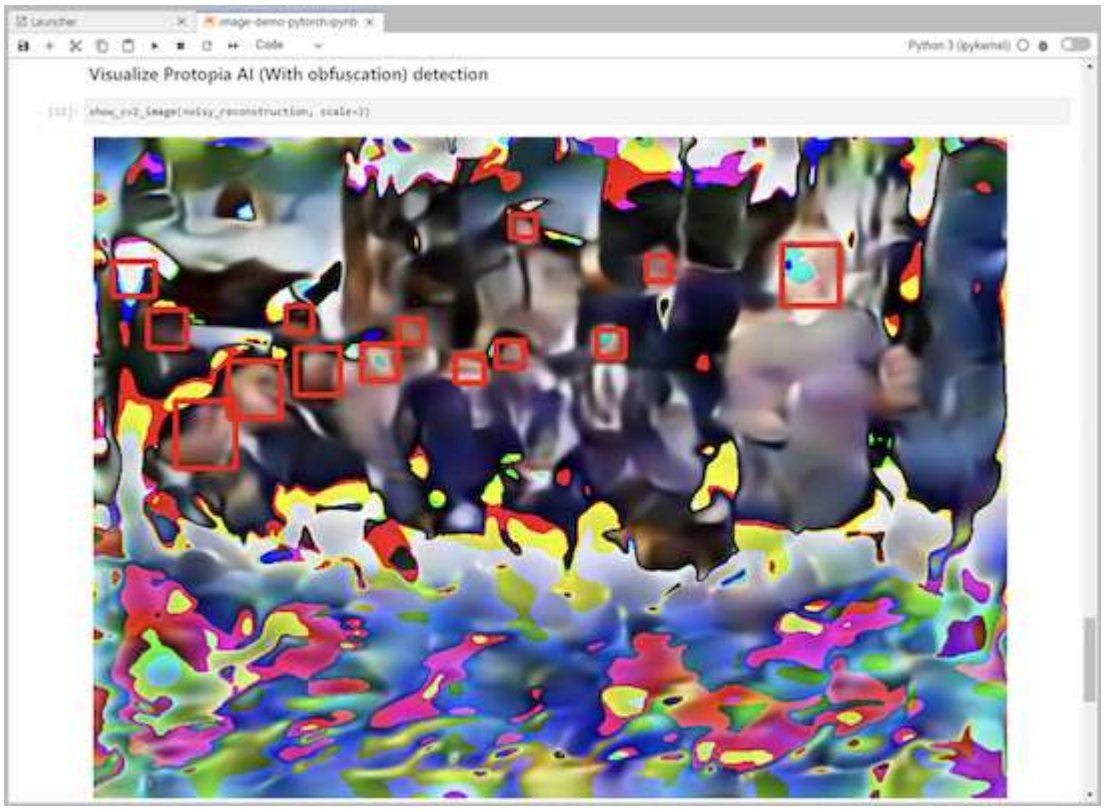
7. 将推理代码添加到笔记本中。以下示例显示了图像检测用例的推理代码。





8. 将 Protopia 混淆添加到您的推理代码中。 Protopia 直接与客户合作提供特定用例的文档，这超出了本技术报告的范围。以下示例展示了添加了 Protopia 混淆的图像检测用例的推理代码。





场景 **2 – Kubernetes** 上的批量推理

1. 为 AI/ML 推理工作负载创建 Kubernetes 命名空间。

```
$ kubectl create namespace inference
namespace/inference created
```

2. 使用NetApp DataOps Toolkit 配置持久卷，用于存储您将执行推理的数据。

```
$ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc
-name=inference-data --size=50Gi
Creating PersistentVolumeClaim (PVC) 'inference-data' in namespace
'inference'.
PersistentVolumeClaim (PVC) 'inference-data' created. Waiting for
Kubernetes to bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC)
'inference-data' in namespace 'inference'.
```

3. 使用您将执行推理的数据填充新的持久卷。

   有几种方法可以将数据加载到 PVC 上。如果您的数据当前存储在与 S3 兼容的对象存储平台（例如NetApp StorageGRID或 Amazon S3）中，那么您可以使用 "NetApp DataOps Toolkit S3 Data Mover 功能"。另一种简单的方法是创建一个 JupyterLab 工作区，然后通过 JupyterLab Web 界面上传文件，如“场景 1 – JupyterLab 中的按需推理 “

4. 为您的批量推理任务创建一个 Kubernetes 作业。以下示例展示了图像检测用例的批量推理作业。此作业对一组图像中的每个图像执行推理，并将推理准确度指标写入标准输出。

```
$ vi inference-job-raw.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-inference-raw
  namespace: inference
spec:
  backoffLimit: 5
  template:
    spec:
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: inference-data
      - name: dshm
        emptyDir:
          medium: Memory
      containers:
      - name: inference
        image: netapp-protopia-inference:latest
        imagePullPolicy: IfNotPresent
        command: ["python3", "run-accuracy-measurement.py", "--dataset",
"/data/netapp-face-detection/FDDB"]
        resources:
          limits:
            nvidia.com/gpu: 2
        volumeMounts:
        - mountPath: /data
          name: data
        - mountPath: /dev/shm
          name: dshm
      restartPolicy: Never
$ kubectl create -f inference-job-raw.yaml
job.batch/netapp-inference-raw created
```

5. 确认推理作业已成功完成。

```
$ kubectl -n inference logs netapp-inference-raw-255sp
100%|██████████| 89/89 [00:52<00:00,  1.68it/s]
Reading Predictions : 100%|██████████| 10/10 [00:01<00:00,  6.23it/s]
Predicting ... : 100%|██████████| 10/10 [00:16<00:00,  1.64s/it]
==================== Results ====================
FDDB-fold-1 Val AP: 0.9491256561145955
FDDB-fold-2 Val AP: 0.9205024466101926
FDDB-fold-3 Val AP: 0.9253013871078468
FDDB-fold-4 Val AP: 0.9399781485863011
FDDB-fold-5 Val AP: 0.9504280149478732
FDDB-fold-6 Val AP: 0.9416473519339292
FDDB-fold-7 Val AP: 0.9241631566241117
FDDB-fold-8 Val AP: 0.9072663297546659
FDDB-fold-9 Val AP: 0.9339648715035469
FDDB-fold-10 Val AP: 0.9447707905560152
FDDB Dataset Average AP: 0.9337148153739079
================================================
mAP: 0.9337148153739079
```

6. 将 Protopia 混淆添加到您的推理工作中。您可以直接从 Protopia 找到有关添加 Protopia 混淆的特定用例说明，这超出了本技术报告的范围。以下示例展示了针对人脸检测用例的批量推理作业，其中添加了 Protopia 混淆，并使用 ALPHA 值 0.8。此作业在对一组图像中的每个图像执行推理之前应用 Protopia 混淆，然后将推理准确度指标写入标准输出。

   我们对 ALPHA 值 0.05、0.1、0.2、0.4、0.6、0.8、0.9 和 0.95 重复了此步骤。您可以在"推理准确性比较。"

```
$ vi inference-job-protopia-0.8.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: netapp-inference-protopia-0.8
  namespace: inference
spec:
  backoffLimit: 5
  template:
    spec:
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: inference-data
      - name: dshm
        emptyDir:
          medium: Memory
      containers:
      - name: inference
        image: netapp-protopia-inference:latest
        imagePullPolicy: IfNotPresent
        env:
        - name: ALPHA
          value: "0.8"
        command: ["python3", "run-accuracy-measurement.py", "--dataset",
"/data/netapp-face-detection/FDDB", "--alpha", "$(ALPHA)", "--noisy"]
        resources:
          limits:
            nvidia.com/gpu: 2
        volumeMounts:
        - mountPath: /data
          name: data
        - mountPath: /dev/shm
          name: dshm
      restartPolicy: Never
$ kubectl create -f inference-job-protopia-0.8.yaml
job.batch/netapp-inference-protopia-0.8 created
```

7. 确认推理作业已成功完成。

```
$ kubectl -n inference logs netapp-inference-protopia-0.8-b4dkz
100%|██████████| 89/89 [01:05<00:00,  1.37it/s]
Reading Predictions : 100%|██████████| 10/10 [00:02<00:00,  3.67it/s]
Predicting ... : 100%|██████████| 10/10 [00:22<00:00,  2.24s/it]
=================== Results ====================
FDDB-fold-1 Val AP: 0.8953066115834589
FDDB-fold-2 Val AP: 0.8819580264029936
FDDB-fold-3 Val AP: 0.8781107458462862
FDDB-fold-4 Val AP: 0.9085731346308461
FDDB-fold-5 Val AP: 0.9166445508275378
FDDB-fold-6 Val AP: 0.9101178994188819
FDDB-fold-7 Val AP: 0.8383443678423771
FDDB-fold-8 Val AP: 0.8476311547659464
FDDB-fold-9 Val AP: 0.8739624502111121
FDDB-fold-10 Val AP: 0.8905468076424851
FDDB Dataset Average AP: 0.8841195749171925
================================================
mAP: 0.8841195749171925
```

场景 3 – NVIDIA Triton 推理服务器

1. 为 AI/ML 推理工作负载创建 Kubernetes 命名空间。

```
$ kubectl create namespace inference
namespace/inference created
```

2. 使用NetApp DataOps Toolkit 配置持久卷，用作NVIDIA Triton 推理服务器的模型存储库。

```
$ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc
-name=triton-model-repo --size=100Gi
Creating PersistentVolumeClaim (PVC) 'triton-model-repo' in namespace
'inference'.
PersistentVolumeClaim (PVC) 'triton-model-repo' created. Waiting for
Kubernetes to bind volume to PVC.
Volume successfully created and bound to PersistentVolumeClaim (PVC)
'triton-model-repo' in namespace 'inference'.
```

3. 将您的模型存储在新的持久卷中 "格式"NVIDIA Triton 推理服务器可以识别它。

   有几种方法可以将数据加载到 PVC 上。一种简单的方法是创建一个 JupyterLab 工作区，然后通过 JupyterLab Web 界面上传文件，如“场景 1 – JupyterLab 中的按需推理 。"

4. 使用NetApp DataOps Toolkit 部署新的NVIDIA Triton Inference Server 实例。

```
$ netapp_dataops_k8s_cli.py create triton-server --namespace=inference
--server-name=netapp-inference --model-repo-pvc-name=triton-model-repo
Creating Service 'ntap-dsutil-triton-netapp-inference' in namespace
'inference'.
Service successfully created.
Creating Deployment 'ntap-dsutil-triton-netapp-inference' in namespace
'inference'.
Deployment 'ntap-dsutil-triton-netapp-inference' created.
Waiting for Deployment 'ntap-dsutil-triton-netapp-inference' to reach
Ready state.
Deployment successfully created.
Server successfully created.
Server endpoints:
http: 192.168.0.152: 31208
grpc: 192.168.0.152: 32736
metrics: 192.168.0.152: 30009/metrics
```

5. 使用 Triton 客户端 SDK 执行推理任务。以下 Python 代码摘录使用 Triton Python 客户端 SDK 执行人脸检测用例的推理任务。此示例调用 Triton API 并传入图像进行推理。然后，Triton 推理服务器接收请求，调用模型，并将推理输出作为 API 结果的一部分返回。

```
# get current frame
frame = input_image
# preprocess input
preprocessed_input = preprocess_input(frame)
preprocessed_input = torch.Tensor(preprocessed_input).to(device)
# run forward pass
clean_activation = clean_model_head(preprocessed_input)  # runs the
first few layers
##########################################################################
##############
#          pass clean image to Triton Inference Server API for
inferencing           #
##########################################################################
##############
triton_client =
httpclient.InferenceServerClient(url="192.168.0.152:31208",
verbose=False)
model_name = "face_detection_base"
inputs = []
outputs = []
inputs.append(httpclient.InferInput("INPUT__0", [1, 128, 32, 32],
"FP32"))
inputs[0].set_data_from_numpy(clean_activation.detach().cpu().numpy(),
binary_data=False)
```

```
outputs.append(httpclient.InferRequestedOutput("OUTPUT__0",
binary_data=False))
outputs.append(httpclient.InferRequestedOutput("OUTPUT__1",
binary_data=False))
results = triton_client.infer(
    model_name,
    inputs,
    outputs=outputs,
    #query_params=query_params,
    headers=None,
    request_compression_algorithm=None,
    response_compression_algorithm=None)
#print(results.get_response())
statistics =
triton_client.get_inference_statistics(model_name=model_name,
headers=None)
print(statistics)
if len(statistics["model_stats"]) != 1:
    print("FAILED: Inference Statistics")
    sys.exit(1)

loc_numpy = results.as_numpy("OUTPUT__0")
pred_numpy = results.as_numpy("OUTPUT__1")
##################################################################
##############
# postprocess output
clean_pred = (loc_numpy, pred_numpy)
clean_outputs = postprocess_outputs(
    clean_pred, [[input_image_width, input_image_height]], priors,
THRESHOLD
)
# draw rectangles
clean_frame = copy.deepcopy(frame)  # needs to be deep copy
for (x1, y1, x2, y2, s) in clean_outputs[0]:
    x1, y1 = int(x1), int(y1)
    x2, y2 = int(x2), int(y2)
    cv2.rectangle(clean_frame, (x1, y1), (x2, y2), (0, 0, 255), 4)
```

6. 将 Protopia 混淆添加到您的推理代码中。您可以直接从 Protopia 找到有关添加 Protopia 混淆的特定用例说明；但是，此过程超出了本技术报告的范围。以下示例显示了与前面步骤 5 中所示的相同的 Python 代码，但添加了 Protopia 混淆。

   请注意，在将图像传递给 Triton API 之前，会对其进行 Protopia 混淆处理。因此，未混淆的图像永远不会离开本地机器。只有经过混淆的图像才会在网络上传递。此工作流程适用于在受信任区域内收集数据但随后需要传递到该受信任区域之外进行推理的用例。如果没有 Protopia 混淆技术，就不可能实现这种类型的工作流程，因为敏感数据永远不会离开受信任区域。

```
# get current frame
frame = input_image
# preprocess input
preprocessed_input = preprocess_input(frame)
preprocessed_input = torch.Tensor(preprocessed_input).to(device)
# run forward pass
not_noisy_activation = noisy_model_head(preprocessed_input)  # runs the
first few layers
#####################################################################
#          obfuscate image locally prior to inferencing             #
#          SINGLE ADITIONAL LINE FOR PRIVATE INFERENCE              #
#####################################################################
noisy_activation = noisy_model_noise(not_noisy_activation)
#####################################################################
#####################################################################
###################
#          pass obfuscated image to Triton Inference Server API for
inferencing           #
#####################################################################
#####################################################################
###################
triton_client =
httpclient.InferenceServerClient(url="192.168.0.152:31208",
verbose=False)
model_name = "face_detection_noisy"
inputs = []
outputs = []
inputs.append(httpclient.InferInput("INPUT__0", [1, 128, 32, 32],
"FP32"))
inputs[0].set_data_from_numpy(noisy_activation.detach().cpu().numpy(),
binary_data=False)
outputs.append(httpclient.InferRequestedOutput("OUTPUT__0",
binary_data=False))
outputs.append(httpclient.InferRequestedOutput("OUTPUT__1",
binary_data=False))
results = triton_client.infer(
    model_name,
    inputs,
    outputs=outputs,
    #query_params=query_params,
    headers=None,
    request_compression_algorithm=None,
    response_compression_algorithm=None)
#print(results.get_response())
statistics =
triton_client.get_inference_statistics(model_name=model_name,
headers=None)
```

```
print(statistics)
if len(statistics["model_stats"]) != 1:
    print("FAILED: Inference Statistics")
    sys.exit(1)

loc_numpy = results.as_numpy("OUTPUT__0")
pred_numpy = results.as_numpy("OUTPUT__1")
############################################################################
###################

# postprocess output
noisy_pred = (loc_numpy, pred_numpy)
noisy_outputs = postprocess_outputs(
    noisy_pred, [[input_image_width, input_image_height]], priors,
THRESHOLD * 0.5
)
# get reconstruction of the noisy activation
noisy_reconstruction = decoder_function(noisy_activation)
noisy_reconstruction = noisy_reconstruction.detach().cpu().numpy()[0]
noisy_reconstruction = unpreprocess_output(
    noisy_reconstruction, (input_image_width, input_image_height), True
).astype(np.uint8)
# draw rectangles
for (x1, y1, x2, y2, s) in noisy_outputs[0]:
    x1, y1 = int(x1), int(y1)
    x2, y2 = int(x2), int(y2)
    cv2.rectangle(noisy_reconstruction, (x1, y1), (x2, y2), (0, 0, 255),
4)
```

推理精度比较

为了进行此验证，我们使用一组原始图像对图像检测用例进行了推理。然后，我们对同一组图像执行相同的推理任务，并在推理之前添加了 Protopia 混淆。我们使用 Protopia 混淆组件的不同 ALPHA 值重复了该任务。在 Protopia 混淆的背景下，ALPHA 值表示所应用的混淆量，ALPHA 值越高表示混淆级别越高。然后，我们比较了这些不同运行的推理准确性。

以下两个表提供了有关我们的用例的详细信息并概述了结果。

Protopia 直接与客户合作，确定特定用例的适当 ALPHA 值。

组件	详细信息
型号	FaceBoxes（PyTorch）-
数据集	FDDB数据集

原始托邦的混淆	阿尔法	准确性
否	不适用	0.9337148153739079
是	0.05	0.9028766627325002
是	0.1	0.9024301009661478
是	0.2	0.9081836283186224
是	0.4	0.9073066107482036
是	0.6	0.8847816568680239
是	0.8	0.8841195749171925
是	0.9	0.8455427675252052
是	0.95	0.8455427675252052

混淆速度

为了进行此验证，我们将 Protopia 混淆应用于 1920 x 1080 像素图像五次，并测量每次完成混淆步骤所需的时间。

我们使用在单个NVIDIA V100 GPU 上运行的 PyTorch 来应用混淆，并在运行之间清除了 GPU 缓存。在五次运行中，混淆步骤分别花费 5.47ms、5.27ms、4.54ms、5.24ms 和 4.84ms 完成。平均速度为5.072毫秒。

结束语

数据有三种状态：静止、传输和计算。任何人工智能推理服务的一个重要部分应该是在整个过程中保护数据免受威胁。在推理过程中保护数据至关重要，因为该过程可能会暴露有关外部客户和提供推理服务的企业的私人信息。 Protopia AI 是当今市场上用于机密 AI 推理的非侵入式纯软件解决方案。借助 Protopia，AI 仅接收数据记录中对于执行手头的 AI/ML 任务至关重要的转换信息，仅此而已。这种随机变换不是一种掩蔽形式，而是基于通过使用精心策划的噪声以数学方式改变数据的表示。

具有ONTAP功能的NetApp存储系统可提供与本地 SSD 存储相同或更佳的性能，并且与NetApp DataOps Toolkit 结合使用，可为数据科学家、数据工程师、AI/ML 开发人员以及业务或企业 IT 决策者带来以下优势：

- 在人工智能系统、分析系统和其他关键业务系统之间轻松共享数据。这种数据共享减少了基础设施开销，提高了性能，并简化了整个企业的数据管理。
- 独立可扩展的计算和存储，以最大限度地降低成本并提高资源利用率。
- 使用集成的 Snapshot 副本和克隆来简化开发和部署工作流程，以实现即时且节省空间的用户工作区、集成版本控制和自动部署。
- 针对灾难恢复、业务连续性和监管要求的企业级数据保护和数据治理。
- 简化数据管理操作的调用；从 Jupyter 笔记本中的NetApp DataOps Toolkit 快速获取数据科学家工作区的 Snapshot 副本以进行备份和追溯。

NetApp和 Protopia 解决方案提供了灵活的横向扩展架构，非常适合企业级 AI 推理部署。它可以实现数据保护并为敏感信息提供隐私，其中机密的 AI 推理要求可以通过内部部署和混合云部署中的负责任的 AI 实践来满足。

在哪里可以找到更多信息和致谢

要了解有关本文档中描述的信息的更多信息，请参阅以下文档和/或网站：

- NetApp ONTAP数据管理软件 — ONTAP信息库

  http://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62286

- NetApp容器持久存储 — NetApp Trident

  "https://netapp.io/persistent-storage-provisioner-for-kubernetes/"

- NetApp DataOps 工具包

  "https://github.com/NetApp/netapp-dataops-toolkit"

- NetApp容器持久存储 — NetApp Trident

  "https://netapp.io/persistent-storage-provisioner-for-kubernetes/"

- Protopia AI—机密推理

  "https://protopia.ai/blog/protopia-ai-takes-on-the-missing-link-in-ai-privacy-confidential-inference/"

- NetApp BlueXP复制和同步

  "https://docs.netapp.com/us-en/occm/concept_cloud_sync.html#how-cloud-sync-works"

- NVIDIA Triton 推理服务器

  "https://developer.nvidia.com/nvidia-triton-inference-server"

- NVIDIA Triton 推理服务器文档

  "https://docs.nvidia.com/deeplearning/triton-inference-server/index.html"

- PyTorch 中的 FaceBoxes

  "https://github.com/zisianw/FaceBoxes.PyTorch"

声明

- NetApp首席产品经理 Mark Cates
- Sufian Ahmad， NetApp技术营销工程师
- Protopia AI 首席技术官兼教授 Hadi Esmaeilzadeh

版权信息

商标信息