負責的AI和機密推斷資料-採用Protopia Image Transformation的NetApp AI

NetApp Solutions

NetApp
April 12, 2024

目錄

負責的AI和機密推斷資料-採用Protopia Image Transformation的NetApp AI

TR-4928：負責的AI與機密資料推斷-採用Protopia Image與資料轉型的NetApp AI

Sathish Thyagarajan、Michael Oglesby、NetApp秉恩、Jennifer Cwagenberg、Protopia

視覺判讀已成為溝通不可或缺的一部分、因為影像擷取和影像處理的出現。數位影像處理中的人工智慧（AI）帶來嶄新商機、例如癌症醫學領域和其他疾病辨識、研究環境危害的地理空間視覺分析、模式辨識、打擊犯罪的影片處理等。然而、這次機會也帶來非凡的責任。

企業組織將越多決策交給AI、就越能接受與資料隱私和安全性、以及法律、道德和法規問題有關的風險。負責的AI能讓公司和政府組織建立信任與治理、這對於大型企業的AI規模而言至關重要。本文件說明NetApp在三種不同案例中驗證的AI推斷解決方案、其方法是將NetApp資料管理技術與Protopia資料混淆軟體搭配使用、以將敏感資料私有化、並降低風險和道德疑慮。

消費者和企業實體每天都會透過各種數位裝置產生數百萬個影像。隨著資料與運算工作負載的大規模爆炸、企業紛紛改用雲端運算平台來實現擴充性與效率。同時、將影像資料中所含的敏感資訊傳輸至公有雲時、也會產生隱私顧慮。缺乏安全性與隱私權保證、成為部署映像處理AI系統的主要障礙。

此外、還有 "銷毀權利" 根據GDPR、個人有權要求組織清除所有個人資料。也有 "隱私權法案"，建立公平資訊實務守則。照片等數位影像可能構成GDPR規定的個人資料、而GDPR規定必須如何收集、處理及清除資料。若未遵守GDPR、可能會因違反法規而導致高額罰款、嚴重傷害組織。隱私權原則是實作負責AI的骨幹、可確保機器學習（ML）和深度學習（DL）模型預測的公平性、並降低違反隱私權或法規遵循的相關風險。

本文件說明已通過驗證的設計解決方案、可在三種不同的情境下、使用或不使用與維護隱私權及部署負責AI解決方案相關的影像模糊化：

- 情境**1.** Jupyter筆記型電腦內的隨需推斷。
- *案例2.*在Kubernetes上進行批次推斷。
- 情境**3.** NVIDIA Triton推斷伺服器。

針對此解決方案、我們使用專為研究不受限制的面偵測問題而設計的面向區域資料集（FDDB）、以及用於實作FaceBoxes的PyTorch機器學習架構。此資料集包含一組2845個不同解析度影像中5171個面的註釋。此外、本技術報告也提供一些解決方案領域、以及在適用本解決方案的情況下、從NetApp客戶和現場工程師收集到的相關使用案例。

目標對象

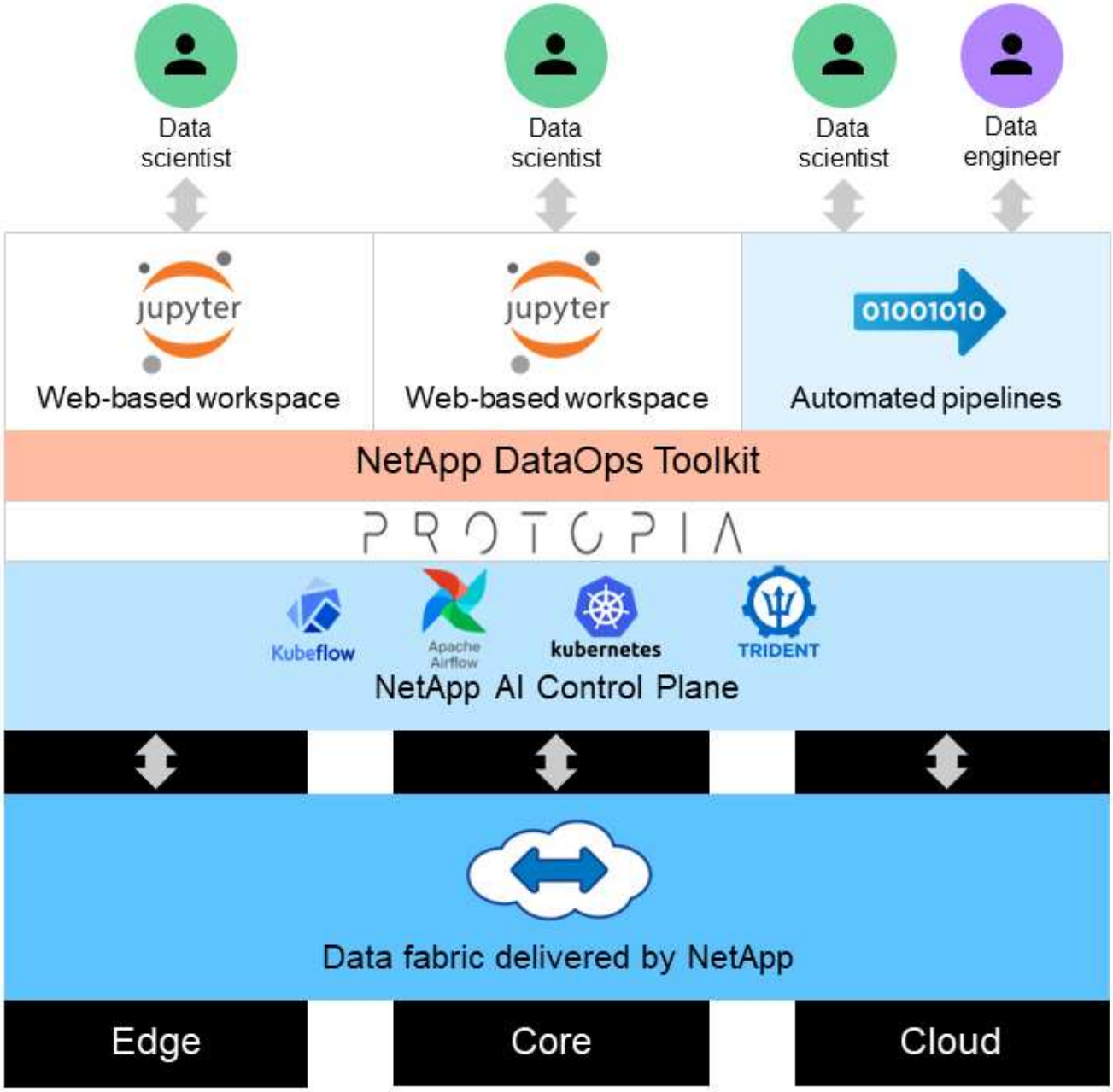
本技術報告適用於下列對象：

- 想要設計及部署負責AI、並解決公共空間中的面向影像處理相關資料保護與隱私權問題的企業領導者與企業架構設計師。
- 資料科學家、資料工程師、AI /機器學習（ML）研究人員、以及致力保護及維護隱私權的AI/ML系統開發人員。

- 企業架構設計師、為符合GDPR、CCPA或國防部隱私權法案（DoD）和政府組織等法規標準的AI/ML模型和應用程式設計資料混淆解決方案。
- 資料科學家和AI工程師正在尋求有效方法來部署深度學習（DL）和AI / ML / DL推斷模型、以保護敏感資訊。
- 負責部署及管理Edge推斷模型的Edge裝置管理員和Edge Server管理員。

解決方案架構

此解決方案的設計、是為了處理大型資料集上的即時和批次推斷AI工作負載、其處理能力是利用GPU與傳統CPU的處理能力。這項驗證可證明對於ML的隱私保護推論、以及尋求負責AI部署的組織所需的最佳資料管理。此解決方案提供適用於單一或多節點Kubernetes平台的架構、可用於邊緣與雲端運算、透過ONTAP 內部部署核心的NetApp AI、NetApp DataOps Toolkit、以及使用Jupyter Lab和CLI介面的Protopia混淆軟體進行互連。下圖顯示採用DataOps Toolkit和Protopia技術之NetApp技術的資料架構邏輯架構總覽。



Protopia混淆軟體可在NetApp DataOps Toolkit上順暢執行、並在離開儲存伺服器之前轉換資料。

解決方案領域

數位影像處理有許多優點、讓許多組織能夠充分利用視覺呈現的相關資料。此NetApp與Protopia解決方案提供獨特的AI推斷設計、可在ML/DL生命週期內保護及私有化AI/ML資料。它可讓客戶保留敏感資料的所有權、運用公有雲或混合雲部署模式來擴充規模和提高效率、減輕隱私權相關疑慮、並在邊緣部署AI推斷。

環境情報

產業在環境危害領域中、有許多方法可以利用地理空間分析。政府和公共工程部門可針對公共衛生和天氣狀況、提出可行的見解、以便在流行病或天然災難（例如野火）期間、更好地向大眾提供建議。例如、您可以在機場或醫院等公共空間中識別COVID-正向病患、而不會影響受影響個人的隱私、並提醒鄰近的相關機關和大眾採取必要的安全措施。

邊緣裝置穿戴式裝置

在軍事領域和戰場上、您可以使用邊緣的AI推斷裝置作為穿戴式裝置、來追蹤戰士的健康狀況、監控駕駛行為、並向主管機關警示接近軍事車輛的安全風險及相關風險、同時保留及保護戰士的隱私。軍事的未來將透過戰場物聯網（IoBT）和軍事物聯網（IoMT）來發展高科技、以提供穿戴式戰鬥裝備、協助戰士利用快速邊緣運算來識別敵人、並在戰鬥中表現更好。保護和保留從無人機和穿戴式設備等邊緣設備收集的視覺資料、對於讓駭客和敵人留在海灣是至關重要的。

非戰鬥人員清空作業

非戰鬥人員清空作業（NEO）由國防部執行、以協助疏散美國公民和國民、DOD文職人員、以及生命面臨適當安全避風港危險的指定人員（接待國（HN）和第三國國民（TCN）。已實施的管理控管措施主要是手動進行疏散者篩選程序。不過、使用高度自動化的AI/ML工具搭配AI / ML影片混淆技術、可能會改善疏散人員識別、疏散追蹤及威脅篩選的準確度、安全性及速度。

醫療與生物醫學研究

影像處理是用來診斷從電腦斷層掃描（CT）或磁共振造影（MRI）取得的3D影像進行手術規劃的不正常情況。HIPAA隱私權規則規範組織如何收集、處理及清除所有個人資訊和數位影像（例如照片）的資料。為了讓資料符合HIPAA安全港規定的可共享資格、必須移除全面相片影像和任何類似影像。用於從結構性CT/磁 共振影像中隱藏個人的面向功能的自動辨識或頭骨刪除演算法等技術、已成為生物醫學研究機構資料共享程序的重要一環。

AI / ML分析的雲端移轉

企業客戶過去曾接受過內部部署的AI/ML模式訓練與部署。基於規模經濟與效率的考量、這些客戶正不斷擴充、將AI/ML功能移轉至公有雲、混合雲或多雲端雲端部署。但是、它們受到其他基礎架構所能接觸到的資料所約束。NetApp解決方案可解決所需的各種網路安全威脅 "資料保護" 以及安全性評估、並與Protopia資料轉型結合使用、將影像處理AI/ML工作負載移轉至雲端的相關風險降至最低。

如需其他跨其他產業的邊緣運算和AI推斷使用案例、請參閱 "TR-4886 AI在邊緣推斷" 以及NetApp AI部落格、 "情報與隱私"。

技術總覽

本節概述完成本解決方案所需的各種技術元件。

Protopia

Protopia AI提供不引人注目的純軟體解決方案、可在當今市場上進行機密推論。Protopia解決方案可將敏感資訊的曝險降至最低、為推論服務提供無與倫比的保護。AI只會提供資料記錄中的資訊、而這些資訊對於執行手邊的工作而言、是真正不可或缺的。大多數推斷工作並不會使用每個資料記錄中的所有資訊。無論您的AI是使用影像、語音、視訊、甚至是結構化的表格式資料、Protopia都能提供您所需的推斷服務。專利的核心技術使用數學上精心策劃的雜訊、以隨機方式轉換資料、並找出特定ML服務不需要的資訊。此解決方案並不會遮罩資料、而是使用精選的隨機雜訊來變更資料呈現方式。

Protopia解決方案會產生將呈現變更為漸層式的靜止最大化方法的問題、此方法仍會保留輸入功能空間中與模型功能相關的資訊。此探索程序會在訓練結束時、以微調通過的方式執行ML模型。通過後會自動產生一組機率分配、低成本的資料轉換會將這些分配的雜訊樣本套用至資料、在將其傳遞至模型進行推斷之前、先將其模糊化。

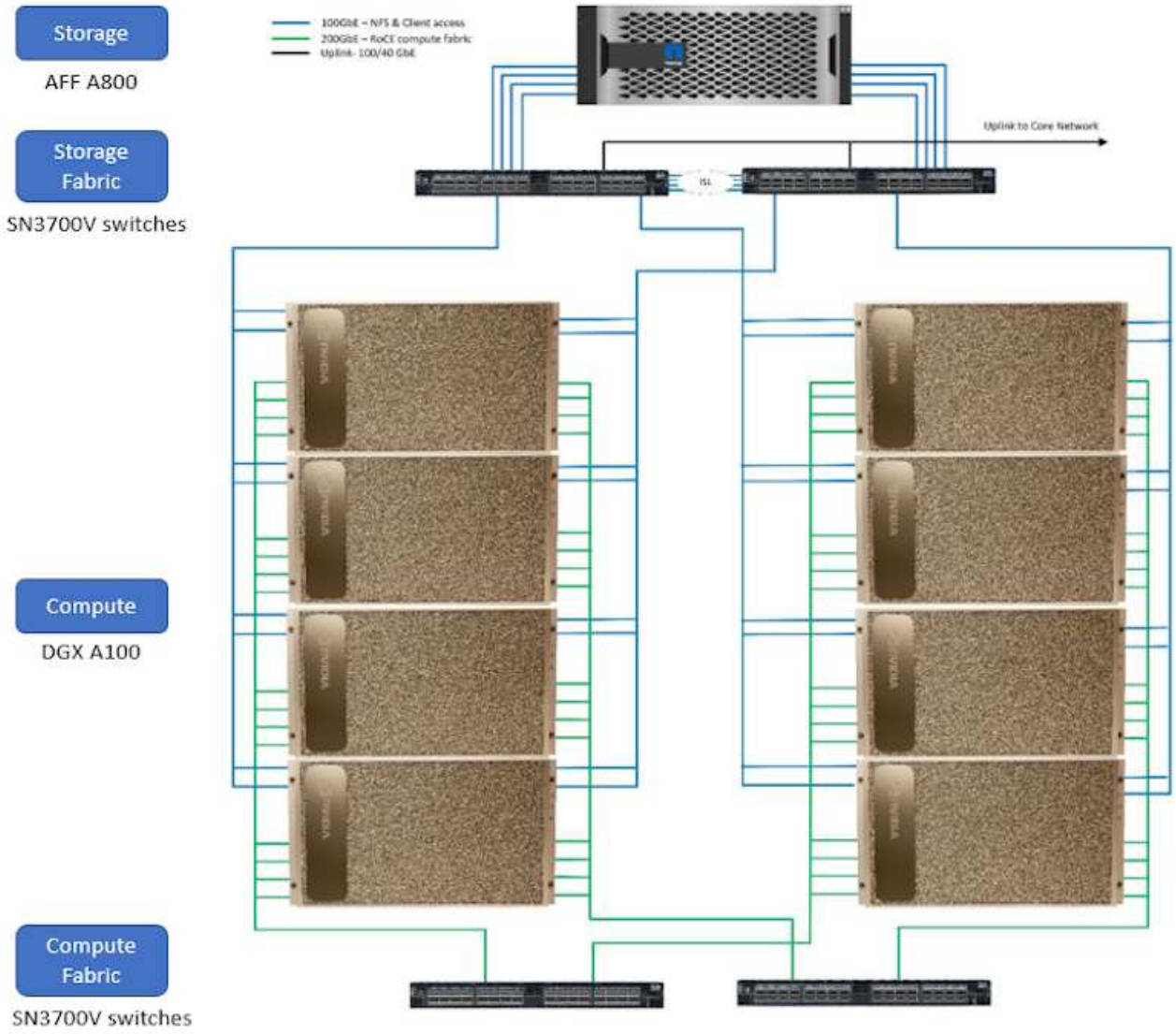
NetApp ONTAP AI

NetApp ONTAP 採用DGX A100系統和NetApp雲端連線儲存系統的NetApp AI參考架構、是由NetApp和NVIDIA開發與驗證的。它為IT組織提供以下優點的架構：

- 消除設計複雜性
- 可獨立擴充運算與儲存設備
- 讓客戶從小規模開始、並無縫擴充
- 提供多種儲存選項、可滿足各種效能與成本效益

AI將DGX A100系統與NetApp的Arde-A800儲存系統與最先進的網路技術緊密整合。ONTAP AFF藉由消除設計複雜度和猜測、AI簡化了AI部署。ONTAP客戶可以從小規模開始、不中斷營運地成長、同時以智慧方式管理從邊緣到核心到雲端及後端的資料。

下圖顯示ONTAP 採用DGX A100系統的整個解決方案系列有多種差異。多達八個DGX A100系統可驗證系統效能。AFF透過將儲存控制器配對新增至ONTAP VMware叢集、架構可擴充至多個機架、以線性效能支援許多DGX A100系統及PB儲存容量。此方法可根據所使用的DL機型大小和所需的效能指標、靈活地改變運算與儲存設備的比率。



如需ONTAP 更多關於AI的資訊、請參閱 "NVA-1153：採用ONTAP NVIDIA DGX A100系統和Mellanox Spectrum乙太網路交換器的NetApp W人工 智慧。"

NetApp ONTAP

NetApp最新一代的儲存管理軟體《支援產業》（NetApp）、可讓企業將基礎架構現代化、並移轉至雲端就緒的資料中心。ONTAP利用領先業界的資料管理功能ONTAP 、無論資料位於何處、只要使用一組工具、即可管理及保護資料。您也可以自由地將資料移至任何需要的位置：邊緣、核心或雲端。包含許多功能、可簡化資料管理、加速及保護關鍵資料、並在混合雲架構中提供新一代基礎架構功能。ONTAP

NetApp DataOps工具套件

NetApp DataOps Toolkit是Python程式庫、讓開發人員、資料科學家、DevOps工程師及資料工程師能夠輕鬆執行各種資料管理工作、例如近乎即時地配置新的資料Volume或JupyterLab工作區、近乎即時地複製資料Volume或JupyterLab工作區、 並近乎即時地擷取資料磁碟區或JupyterLab工作區的快照、以供追蹤或建立基準。此Python程式庫可做為命令列公用程式、或是可匯入任何Python程式或Jupyter筆記型電腦的函數庫。

NVIDIA Triton Inference伺服器

NVIDIA Triton Inference Server是開放原始碼的推斷服務軟體、可協助標準化模型部署與執行、在正式作業中提供快速且可擴充的AI。Triton Inference Server可讓團隊從任何GPU或CPU基礎架構上的任何架構、部署、執行及擴充訓練精良的AI模型、進而簡化AI推斷。Triton Inference Server支援所有主要架構、例如TensorFlow、NVIDIA TensorRT、PyTorch、MXNet、 OpenVINO等。Triton與Kubernetes整合、提供協調與擴充功能、可用於所有主要公有雲AI和Kubernetes平台。它也與許多MLOps軟體解決方案整合。

PyTorch

"PyTorch" 是開放原始碼ML架構。這是最佳化的張量程式庫、可用於深度學習、使用GPU和CPU。PyTorch套件包含多維度游標的資料結構、可提供許多公用程式、以便在其他實用的公用程式之間有效序列化游標。它也有CUDA對應產品、可讓您在具有運算能力的NVIDIA GPU上執行張量運算。在這項驗證中、我們使用OpenCV-Python（CV2）程式庫來驗證我們的模型、同時充分運用Python最直覺式的電腦視覺概念。

簡化資料管理

資料管理對於企業IT營運和資料科學家而言至關重要、因此可將適當的資源用於AI應用程式和訓練AI/ML資料集。下列關於NetApp技術的其他資訊超出此驗證範圍、但可能會因您的部署而有所差異。

包含下列功能的資料管理軟體、可簡化及簡化作業、並降低您的總營運成本：ONTAP

- 即時資料精簡與擴充重複資料刪除技術。資料壓縮可減少儲存區塊內的空間浪費、重複資料刪除技術可大幅提升有效容量。這適用於本機儲存的資料、以及分層至雲端的資料。
- 最低、最大及可調適的服務品質（AQO）。精細的服務品質（QoS）控制有助於維持高共享環境中關鍵應用程式的效能等級。
- NetApp FabricPool自動將冷資料分層至公有和私有雲端儲存選項、包括Amazon Web Services（AWS）、Azure和NetApp StorageGRID 等儲存解決方案。如需FabricPool 更多有關資訊、請參閱 "TR-4598：FabricPool 最佳實務做法"。

加速並保護資料

提供優異的效能與資料保護、並以下列方式擴充這些功能：ONTAP

- 效能與較低的延遲。以最低的延遲提供最高的處理量。ONTAP
- 資料保護：支援所有平台的通用管理功能、可提供內建的資料保護功能。ONTAP
- NetApp Volume Encryption（NVE）。支援內建和外部金鑰管理、提供原生Volume層級的加密功能。ONTAP
- 多租戶和多因素驗證。支援以最高安全等級共享基礎架構資源。ONTAP

符合未來需求的基礎架構

下列功能可協助滿足嚴苛且不斷變化的業務需求：ONTAP

- 無縫擴充與不中斷營運。支援在不中斷營運的情況下、將容量新增至現有控制器和橫向擴充叢集。ONTAP客戶可以升級至最新技術、例如NVMe和32GB FC、而不需進行昂貴的資料移轉或中斷運作。
- 雲端連線：NetApp是最具雲端連線能力的儲存管理軟體、可在所有公有雲中選擇軟體定義儲存（AI）和雲端原生執行個體（NetApp）ONTAP ONTAP Select Cloud Volumes Service 。
- 與新興應用程式整合。利用支援現有企業應用程式的相同基礎架構、為新一代平台和應用程式提供企業級資

料服務、例如自動駕駛車輛、智慧城市和產業4.0。ONTAP

NetApp Astra Control

NetApp Astra產品系列提供儲存與應用程式感知資料管理服務、適用於Kubernetes應用程式的內部部署與公有雲、採用NetApp儲存與資料管理技術。它可讓您輕鬆備份Kubernetes應用程式、將資料移轉至不同的叢集、並即時建立運作中的應用程式複本。如果您需要管理在公有雲上執行的Kubernetes應用程式、請參閱的文件 "Astra控制服務"。Astra Control Service是NetApp管理的服務、可在Google Kubernetes Engine（GKE）和Azure Kubernetes Service（KS）中、提供Kubernetes叢集的應用程式感知資料管理功能。

NetApp Astra Trident

Astra "Trident" NetApp是適用於Docker和Kubernetes的開放原始碼動態儲存協調工具、可簡化持續儲存的建立、管理和使用。Kubernetes原生應用程式Trident直接在Kubernetes叢集內執行。Trident可讓客戶將DL Container映像無縫部署到NetApp儲存設備、並為AI Container部署提供企業級體驗。Kubernetes使用者（ML開發人員、資料科學家等）可以建立、管理及自動化協調與複製、以充分利用NetApp技術所提供的進階資料管理功能。

NetApp BlueXP 複製與同步

"BlueXP 複製與同步" 是一項NetApp服務、可快速且安全地同步資料。無論您需要在內部部署 NFS 或 SMB 檔案共用之間傳輸檔案、 NetApp StorageGRID 、 NetApp ONTAP S3 、 NetApp Cloud Volumes Service 、 Azure NetApp Files 、 Amazon Simple Storage Service （ Amazon S3 ）、 Amazon Elastic File System （ Amazon EFS ）、 Azure Blob 、 Google Cloud Storage 、 或者 IBM Cloud Object Storage 、 BlueXP 複製與同步功能可快速安全地將檔案移至所需的位置。資料傳輸完成後、即可在來源和目標上完全使用。BlueXP 複製和 Syncc 會根據預先定義的排程持續同步資料、只移動資料量、將資料複寫所花的時間和金錢降到最低。BlueXP 複製與同步是一種軟體即服務（ SaaS ）工具、設定與使用極為簡單。BlueXP 複製與同步所觸發的資料傳輸是由資料代理人執行。您可以在 AWS 、 Azure 、 Google Cloud Platform 或內部部署中部署 BlueXP 複製和同步資料代理人。

NetApp BlueXP 分類

採用強大的AI演算法、 "NetApp BlueXP 分類" 在整個資料產業中提供自動化控管與資料治理功能。您可以輕鬆找出成本節約效益、找出法規遵循與隱私權的考量、並找出最佳化商機。BlueXP 分類儀表板可讓您深入瞭解如何識別重複的資料、以消除備援、對應個人、非個人及敏感資料、並針對敏感資料和異常狀況開啟警示。

測試與驗證計畫

針對此解決方案設計、已驗證下列三種情境：

- JupyterLab工作區中的一項推斷工作、無論是否使用Protopia混淆、都是使用適用於Kubernetes的NetApp DataOps Toolkit進行協調。
- Kubernetes上的批次推斷工作（無論是否有Protopia混淆）、是使用NetApp DataOps Toolkit for Kubernetes所協調的資料Volume。
- 使用NVIDIA Triton Inference Server執行個體的推斷工作、使用適用於Kubernetes的NetApp DataOps Toolkit進行協調。在啟動Triton推斷API之前、我們先將Protopia混淆套用至映像、以模擬透過網路傳輸的任何資料都必須模糊不清的常見需求。此工作流程適用於在信任區域內收集資料、但必須在信任區域外傳遞資料以供推斷的使用案例。如果不使用Protopia混淆、就無法在未將敏感資料離開信任區域的情況下實作此類工作流程。

測試組態

下表概述解決方案設計驗證環境。

元件	版本
Kubernetes	1.21.6
NetApp Astra Trident SCSI驅動程式	22.01.0
適用於Kubernetes的NetApp DataOps工具套件	2.3.0
NVIDIA Triton Inference伺服器	21.11-py3.

測試程序

本節說明完成驗證所需的工作。

先決條件

若要執行本節所述的工作、您必須安裝並設定下列工具、才能存取Linux或MacOS主機：

- Kubectl（設定為存取現有的Kubernetes叢集）
  - 您可以找到安裝與組態指示 "請按這裡"。
- 適用於Kubernetes的NetApp DataOps工具套件
  - 您可以找到安裝指示 "請按這裡"。

案例**1–JupyterLab**的隨需推斷

1. 為AI / ML推斷工作負載建立Kubernetes命名空間。

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

2. 使用NetApp DataOps Toolkit來配置持續磁碟區、以儲存您要在其中執行推斷的資料。

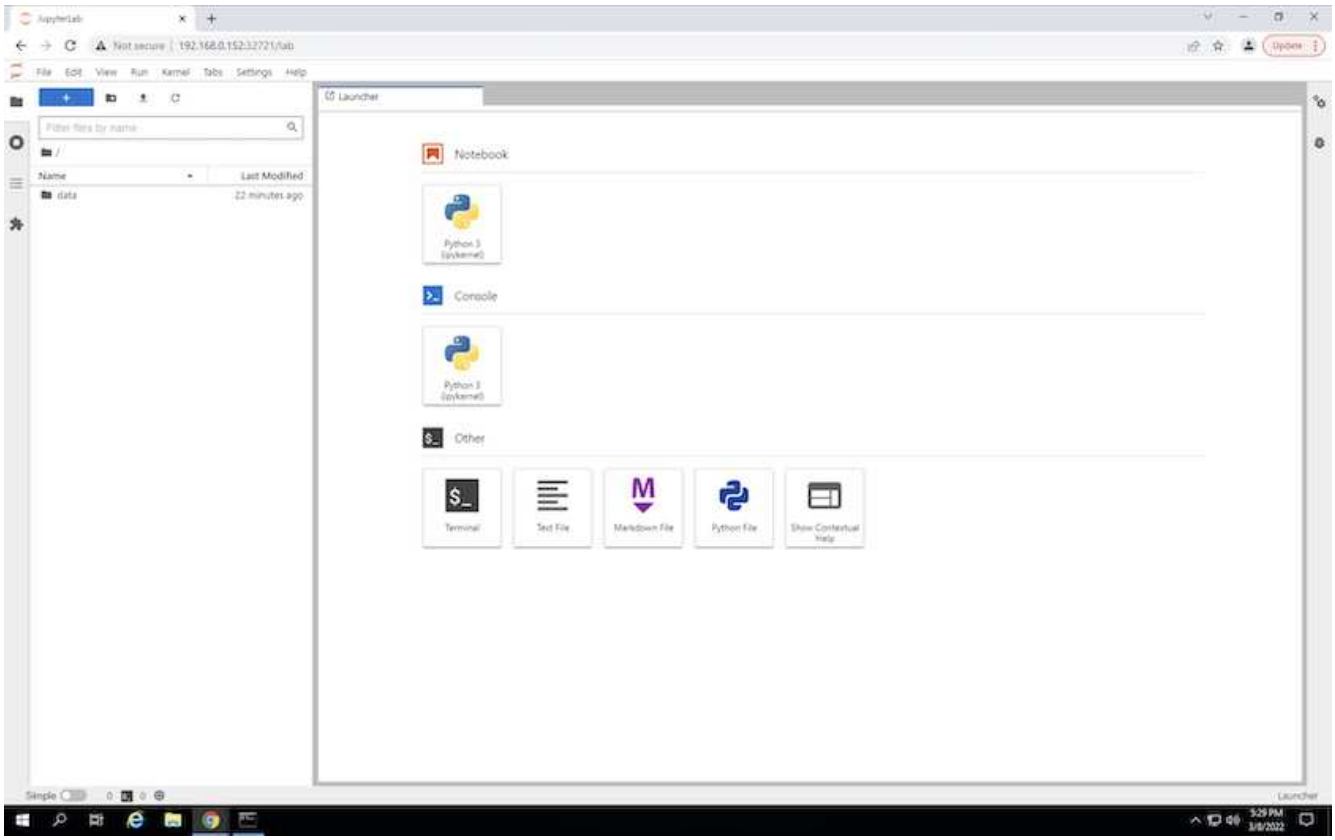
```
$ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc
-name=inference-data --size=50Gi
Creating PersistentVolumeClaim (PVC) 'inference-data' in namespace
'inference'.
PersistentVolumeClaim (PVC) 'inference-data' created. Waiting for
Kubernetes to bind volume to PVC.
Volume successfully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

3. 使用NetApp DataOps Toolkit建立新的JupyterLab工作區。使用「-mount-PVC'」選項來掛載上一步建立的持續磁碟區。使用「-nvidia-GPU」選項、視需要將NVIDIA GPU分配給工作區。

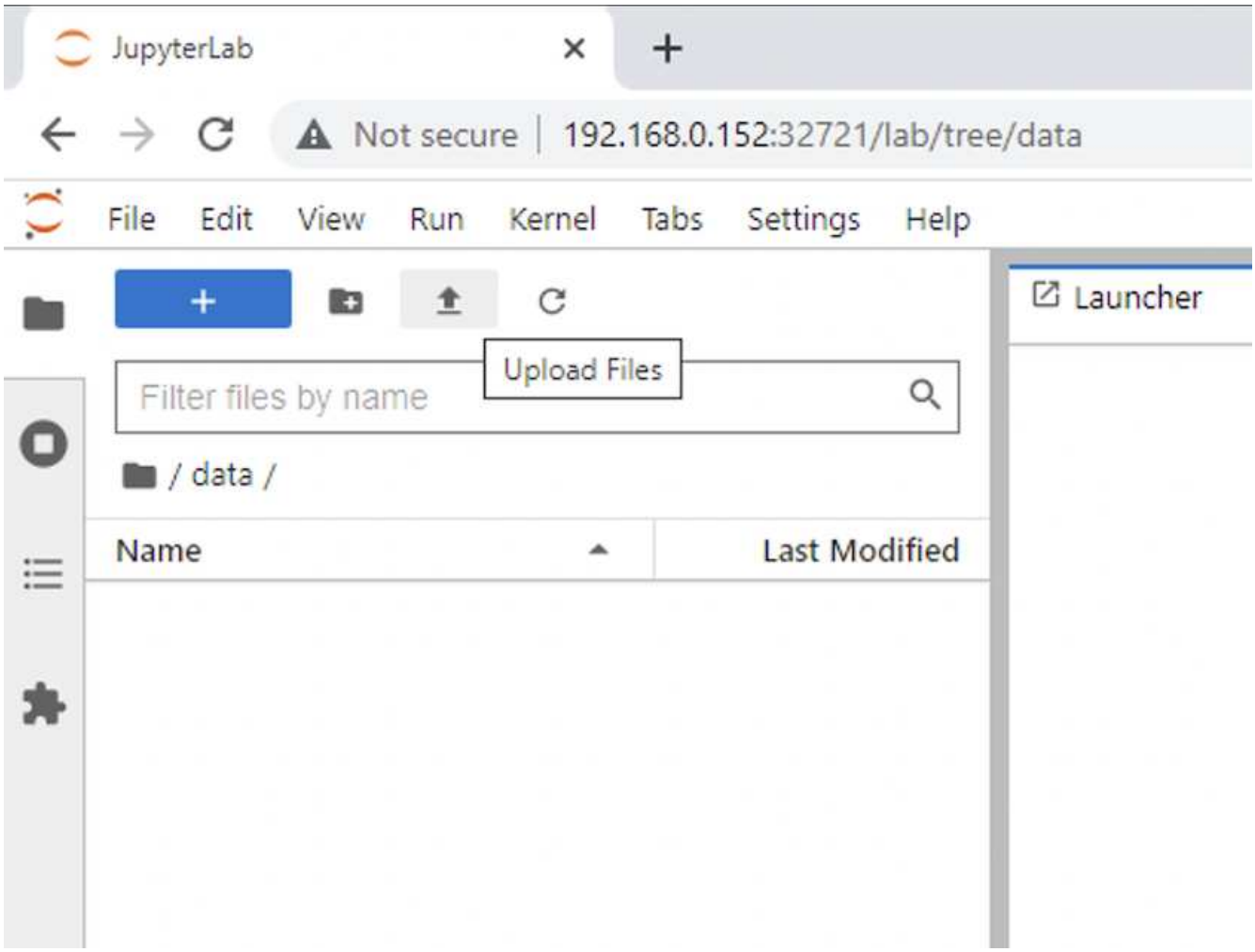
   在以下範例中、持續性磁碟區「推斷資料」會掛載到JupyterLab工作區容器、位於「/home/jovyan/data」。使用正式的Project Jupyter Container映像時、「/home/jovyan」會顯示為JupyterLab網路介面中的頂層目錄。

```
$ 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
```

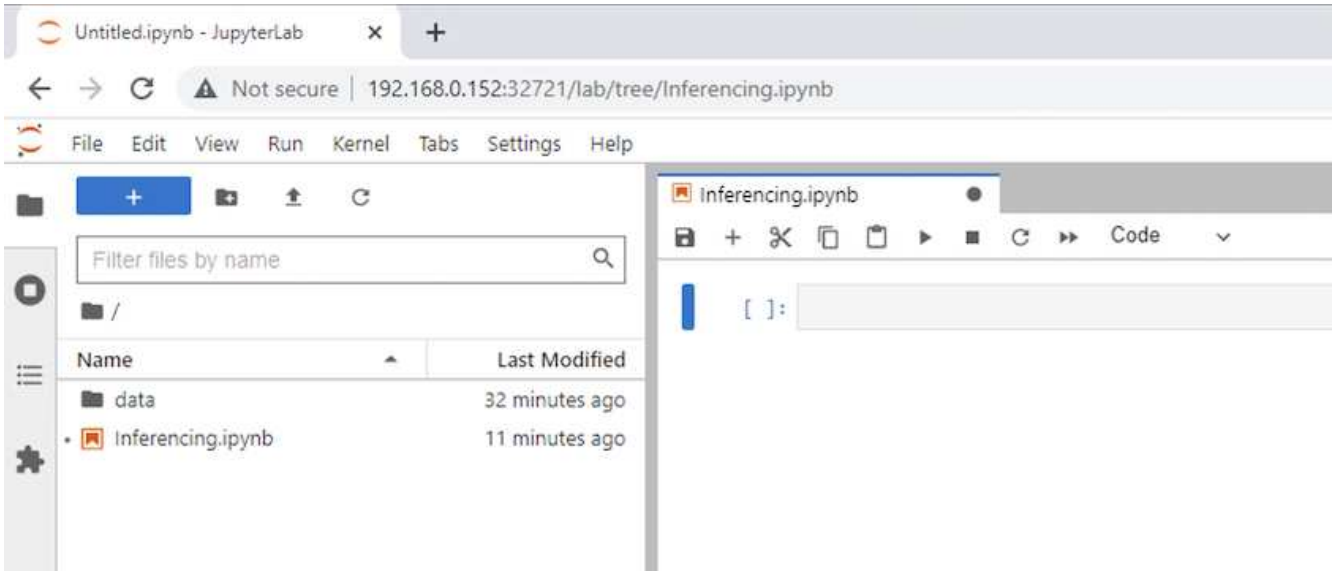
4. 使用「create jupyterlab」命令輸出中指定的URL存取JupyterLab工作區。資料目錄代表掛載到工作區的持續磁碟區。



5. 開啟「DATA」目錄、然後上傳要執行提示的檔案。檔案上傳至資料目錄時、會自動儲存在掛載至工作區的持續磁碟區上。若要上傳檔案、請按一下「上傳檔案」圖示、如下圖所示。



6. 返回最上層目錄並建立新的筆記本。



7. 在筆記本中加入推斷程式碼。下列範例顯示影像偵測使用案例的推斷代碼。

Launcher | image-demo-pytorch.ipynb

Code | Python 3 (ipykernel)

STEP 3-1: Clean (Without obfuscation) detection

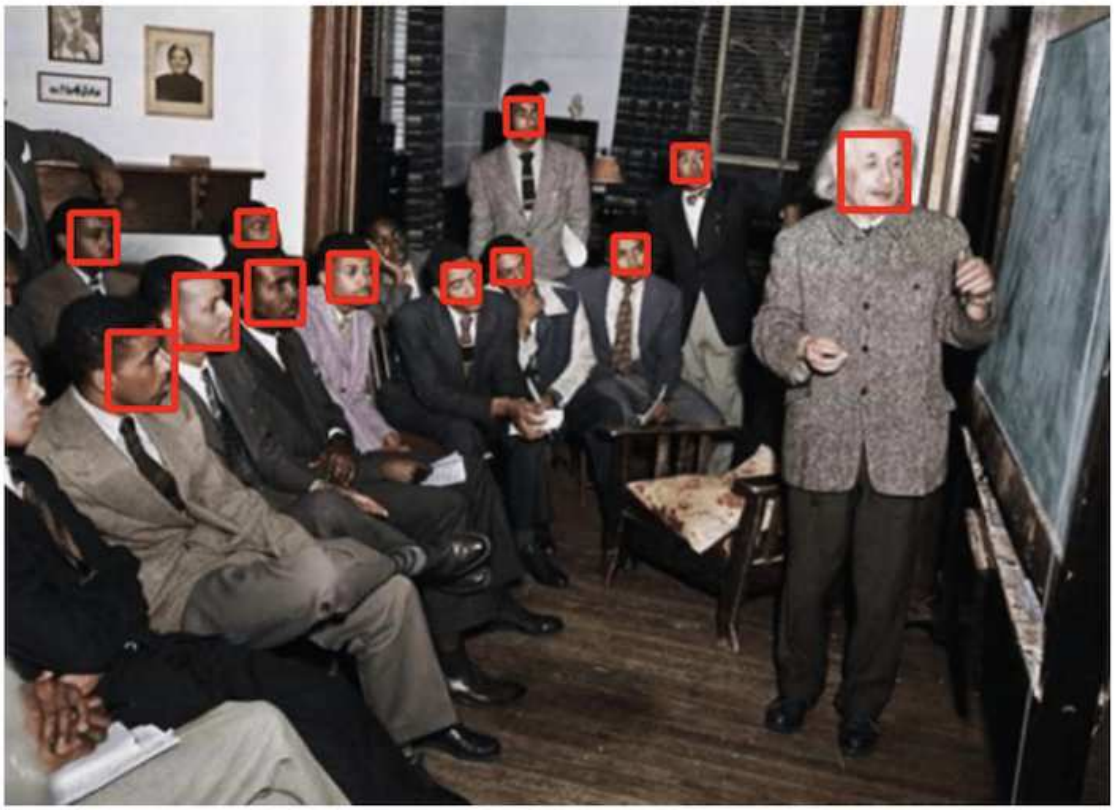
[9]:
```
# get current frame
frame = input_image

# preprocess input
preprocessed_input = preprocess_input(frame)
preprocessed_input = torch.Tensor(preprocessed_input).to(device)

# run forward pass
clean_activation = clean_model.forward_head(preprocessed_input)  # runs the first few layers
loc, pred = clean_model.forward_tail(clean_activation)  # runs rest of the layers

# postprocess output
clean_pred = (loc.detach().cpu().numpy(), pred.detach().cpu().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)
```

Launcher | image-demo-pytorch.ipynb

Code | Python 3 (ipykernel)

Visualize Clean (Without obfuscation) detection

[10]:
```
show_cv2_image(clean_frame, scale=2)
```



8. 將Protopia混淆新增至您的推斷程式碼。Protopia直接與客戶合作、提供特定使用案例的文件、並不在本技術報告的範圍之內。以下範例顯示新增Protopia混淆功能時、影像偵測使用案例的推斷程式碼。

Launcher | image-demo-pytorch.ipynb

Code | Python 3 (ipykernel)

STEP 3-2: Protopia AI (With obfuscation) detection

[11]:
```python
# 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.forward_head(preprocessed_input)  # runs the first few layers
##################################################################
#          SINGLE ADITIONAL LINE FOR PRIVATE INFERENCE           #
##################################################################
noisy_activation = noisy_model.forward_noise(not_noisy_activation)
##################################################################
loc, pred = noisy_model.forward_tail(noisy_activation)  # runs rest of the layers

# postprocess output
noisy_pred = (loc.detach().cpu().numpy(), pred.detach().cpu().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)
```

Launcher | image-demo-pytorch.ipynb

Code | Python 3 (ipykernel)

Visualize Protopia AI (With obfuscation) detection

[12]:
```python
show_cv2_image(noisy_reconstruction, scale=2)
```

案例2：Kubernetes上的批次推斷

1. 為AI / ML推斷工作負載建立Kubernetes命名空間。

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

2. 使用NetApp DataOps Toolkit來配置持續磁碟區、以儲存您要在其中執行推斷的資料。

```
$ 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. 在新的持續磁碟區中填入您要執行推斷的資料。

   有多種方法可將資料載入至PVc。如果您的資料目前儲存在S3相容的物件儲存平台、例如NetApp StorageGRID 功能區或Amazon S3、您就可以使用 "NetApp DataOps Toolkit S3 Data Mover功能"。另一種簡單的方法是建立JupyterLab工作區、然後透過JupyterLab網頁介面上傳檔案、如「」一節中步驟3至5所述案例1–JupyterLab的隨需推斷。」

4. 為批次推斷工作建立Kubernetes工作。下列範例顯示影像偵測使用案例的批次推斷工作。此工作會在一組映像中的每個映像上執行推斷、並將推斷準確度指標寫入stdout。

```
$ 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. 確認推斷工作已成功完成。

```
$ 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. 在推斷工作中加入Protopia混淆。您可以在本技術報告範圍之外的Protopia中、找到直接新增Protopia混淆的使用案例特定指示。下列範例顯示使用0.8的Alpha值新增Protopia模糊處理時、面偵測使用案例的批次推斷工作。此工作會先套用Protopia混淆、再對一組影像中的每個影像進行推斷、然後將推斷準確度指標寫入stdout。

   我們重複此步驟以取得Alpha值、包括0.05、0.1、0.2、0.4、0.6、 0.8、0.9及0.95。您可以在中看到結果 "「推斷準確度比較」。"

```
$ 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. 確認推斷工作已成功完成。

```
$ 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
```

案例3–NVIDIA Triton Inference Server

1. 為AI / ML推斷工作負載建立Kubernetes命名空間。

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

2. 使用NetApp DataOps Toolkit來配置持續磁碟區、以作為NVIDIA Triton Inference Server的模型儲存庫。

```
$ 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. 將您的模型儲存在中的新持續磁碟區上 "格式" NVIDIA Triton Inference伺服器也能辨識這點。

   有多種方法可將資料載入至PVc。簡單的方法是建立JupyterLab工作區、然後透過JupyterLab網路介面上傳檔案、如「」中的步驟3至5所述案例1–JupyterLab的隨需推斷。」

4. 使用NetApp DataOps Toolkit部署新的NVIDIA Triton Inference Server執行個體。

```
$ 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用戶端SDK執行推斷工作。下列Python程式碼摘錄使用Triton Python用戶端SDK、針對面偵測使用案例執行推斷工作。此範例會呼叫Triton API、並傳入影像以供參考。然後Triton Inference伺服器會收到要求、啟動模型、並傳回推斷輸出、做為API結果的一部分。

```
# 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. 將Protopia混淆新增至您的推斷程式碼。您可以找到直接從Protopia新增Protopia混淆的使用案例特定指示、不過此程序不在本技術報告的範圍之內。以下範例顯示與前述步驟5相同的Python程式碼、但新增了Protopia混淆功能。

   請注意、Protopia混淆會套用至映像、然後再傳遞至Triton API。因此、不模糊的影像永遠不會離開本機機器。只有模糊的映像會透過網路傳送。此工作流程適用於在信任區域內收集資料、但需要在信任區域外傳遞資料以進行推斷的使用案例。如果沒有Protopia混淆、就無法在不敏感資料離開信任區域的情況下實作這類工作流程。

```
# 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)
```

推斷準確度比較

為了進行此驗證、我們使用一組原始影像來推斷影像偵測使用案例。然後、我們在相同的影像集上執行相同的推斷工作、並在推斷之前新增Protopia模糊功能。我們使用不同的Alpha值來重複執行Protopia混淆元件的工作。在Protopia混淆的情況下、Alpha值代表套用的模糊處理量、而較高的Alpha值代表較高層級的模糊處理。然後我們比較這些不同路跑的推斷準確度。

下表提供使用案例的詳細資料、並概述結果。

Protopia直接與客戶合作、針對特定使用案例來判斷適當的Alpha值。

元件	詳細資料
模型	FaceBoxes（PyTorch）-
資料集	FDDB資料集

普洛皮亞混淆	**Alpha**	準確度
否	不適用	0.9337148153739079
是的	0.05	0.9028766627325002
是的	0.1	0.9024301009661478
是的	0.2	0.9081836283186280
是的	0.4	0.90766107482036
是的	0.6	0.8847816568680239
是的	0.8	0.8841195749171925
是的	0.9.	0.84554276775252052
是的	0.95	0.84554276775252052

模糊化速度

在這項驗證中、我們將Protopia混淆套用至1920 x 1080像素映像五倍、並測量每次執行模糊化步驟所需的時間量。

我們使用在單一NVIDIA V100 GPU上執行的PyTorch來套用混淆、並在執行期間清除GPU快取。在這五次路跑中、混淆步驟分別花費5.47毫秒、5.27毫秒、4.54毫秒、5.24毫秒和4.84毫秒來完成。平均速度為5.072ms。

結論

資料存在三種狀態：閒置、傳輸和運算。任何AI推斷服務的重要一環、都應該是在整個流程中保護資料免受威脅。在推斷過程中保護資料是非常重要的、因為此程序可能會揭露外部客戶和提供推斷服務的企業的私人資訊。Protopia AI是專為機密AI提供的不突兀軟體專屬解決方案、可在當今市場中提供相關資訊。有了Protopia、AI只會在資料記錄中輸入經過轉換的資訊、而這些資訊對於執行手邊的AI/ML工作而言非常重要、而且不再需要其他資訊。這種隨機轉型並不是一種遮罩形式、而是以數學方式使用「精選雜訊」來變更資料的呈現方式為基礎。

NetApp儲存系統ONTAP 具備各種功能、可提供與本機SSD儲存設備相同或更高的效能、並搭配NetApp DataOps Toolkit、為資料科學家、資料工程師、AI/ML開發人員、企業或企業IT決策者提供下列效益：

- 輕鬆在AI系統、分析和其他關鍵業務系統之間共享資料。這種資料共享可降低基礎架構的負荷、改善效能、並簡化整個企業的資料管理。
- 獨立擴充的運算與儲存設備、可將成本降至最低、並改善資源使用率。
- 利用整合式Snapshot複本與複製、簡化開發與部署工作流程、提供即時且節省空間的使用者工作區、整合式版本控制、以及自動化部署。
- 企業級的資料保護與資料治理功能、可滿足災難恢復、營運不中斷及法規要求。
- 簡化資料管理作業的叫用作業；快速取得資料科學家工作區的Snapshot複本、以便從Jupyter筆記型電腦的NetApp DataOps Toolkit進行備份與追蹤。

NetApp與Protopia解決方案提供靈活的橫向擴充架構、是企業級AI推斷部署的理想選擇。它能保護資料、並為敏

感資訊提供隱私保護、在內部部署和混合雲部署中、均能以負責的AI實務做法來滿足機密AI推斷要求。

何處可以找到其他資訊和認可

若要深入瞭解本文件所述資訊、請參閱下列文件和/或網站：

- NetApp ONTAP 數據管理軟體ONTAP —資訊庫

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

- NetApp容器持續儲存設備：NetApp Trident

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

- NetApp DataOps工具套件

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

- NetApp容器持續儲存設備：NetApp Astra Trident

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

- Protopia AI—機密推論

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

- NetApp BlueXP 複製與同步

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

- NVIDIA Triton Inference伺服器

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

- NVIDIA Triton Inference Server文件

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

- PyTorch中的FaceBoxes

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

感謝

- NetApp首席產品經理Mark Cates
- NetApp技術行銷工程師Sufian Ahmad
- Protopia AI技術長暨教授HADI Esmaeilzadeh