プロトピア画像変換による責任あるAI

NetApp artificial intelligence solutions

NetApp
August 18, 2025

This PDF was generated from https://docs.netapp.com/ja-jp/netapp-solutions-ai/software/ai-conf-protopia-overview.html on August 18, 2025. Always check docs.netapp.com for the latest.

目次

プロトピア画像変換による責任あるAI . . . 1
TR-4928: 責任あるAIと機密推論 - NetApp AIとProtopiaイメージおよびデータ変換 . . . 1
対象 . . . 2
ソリューションアーキテクチャ . . . 2
ソリューション領域 . . . 3
環境情報 . . . 4
エッジデバイスウェアラブル . . . 4
非戦闘員避難作戦 . . . 4
ヘルスケアとバイオメディカル研究 . . . 4
AI/ML分析のクラウド移行 . . . 4
技術概要 . . . 4
プロトピア . . . 5
NetApp ONTAP AI . . . 5
NetApp ONTAP . . . 6
NetApp DataOps ツールキット . . . 6
NVIDIA Triton 推論サーバー . . . 7
パイトーチ . . . 7
NetApp Astraコントロール . . . 8
NetAppTrident . . . 8
NetApp BlueXPコピーと同期 . . . 8
NetApp BlueXP分類 . . . 9
テストおよび検証計画 . . . 9
テスト構成 . . . 9
テスト手順 . . . 9
前提条件 . . . 9
シナリオ1 – JupyterLabでのオンデマンド推論 . . . 10
シナリオ2 – Kubernetesでのバッチ推論 . . . 15
シナリオ3 – NVIDIA Triton推論サーバー . . . 20
推論精度の比較 . . . 24
難読化速度 . . . 25
まとめ . . . 25
追加情報と謝辞の参照先 . . . 26
謝辞 . . . 27

プロトピア画像変換による責任あるAI

TR-4928: 責任あるAIと機密推論 - NetApp AIとProtopiaイメージおよびデータ変換

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

画像キャプチャと画像処理の出現により、視覚的な解釈はコミュニケーションの不可欠な部分になりました。デジタル画像処理における人工知能 (AI) は、がんやその他の病気の特定といった医療分野、環境危険の研究のための地理空間視覚分析、パターン認識、犯罪対策のためのビデオ処理など、新たなビジネスチャンスをもたらします。しかし、この機会には特別な責任も伴います。

組織が AI に意思決定を委ねるほど、データのプライバシーとセキュリティ、および法律、倫理、規制の問題に関連するリスクを受け入れることになります。責任ある AI により、企業や政府機関は、大規模な企業における AI の導入に不可欠な信頼とガバナンスを構築できるようになります。このドキュメントでは、 NetApp のデータ管理テクノロジと Protopia データ難読化ソフトウェアを使用して機密データを非公開化し、リスクと倫理的な懸念を軽減する、3 つの異なるシナリオでNetAppによって検証された AI 推論ソリューションについて説明します。

消費者と企業の両方によって、さまざまなデジタルデバイスを使用して毎日何百万もの画像が生成されています。その結果、データと計算ワークロードが爆発的に増加し、企業は規模と効率を求めてクラウド コンピューティング プラットフォームを導入するようになりました。一方、パブリッククラウドへの転送では、画像データに含まれる機密情報に対するプライバシーの懸念が生じます。セキュリティとプライバシーの保証の欠如は、画像処理 AI システムの導入における主な障壁となります。

さらに、 "消去権" GDPR により、個人は組織に対して自分の個人データをすべて消去するよう要求する権利を有します。また、 "プライバシー法"公正な情報慣行の規範を確立する。写真などのデジタル画像は、データの収集、処理、消去方法を規定する GDPR では個人データを構成する場合があります。これを怠ると GDPR に準拠していないことになり、コンプライアンス違反に対する高額の罰金が科せられ、組織に深刻な損害を与える可能性があります。プライバシー原則は、機械学習 (ML) およびディープラーニング (DL) モデルの予測における公平性を確保し、プライバシー違反や規制遵守に関連するリスクを低減する、責任ある AI を実装するための基盤の 1 つです。

このドキュメントでは、プライバシーの保護と責任ある AI ソリューションの導入に関連する、画像の難読化の有無にかかわらず、3 つの異なるシナリオにおける検証済みの設計ソリューションについて説明します。

- シナリオ**1** Jupyter ノートブック内でのオンデマンド推論。
- シナリオ**2** Kubernetes でのバッチ推論。
- シナリオ**3** NVIDIA Triton 推論サーバー。

このソリューションでは、制約のない顔検出の問題を研究するために設計された顔領域のデータセットである顔検出データセットとベンチマーク (FDDB) を、FaceBoxes の実装用の PyTorch 機械学習フレームワークと組み合わせて使用します。このデータセットには、さまざまな解像度の 2845 枚の画像セットに含まれる 5171 個の顔の注釈が含まれています。さらに、この技術レポートでは、このソリューションを適用できる状況において、 NetApp の顧客とフィールド エンジニアから収集されたソリューション領域と関連するユースケースの一部を紹介します。

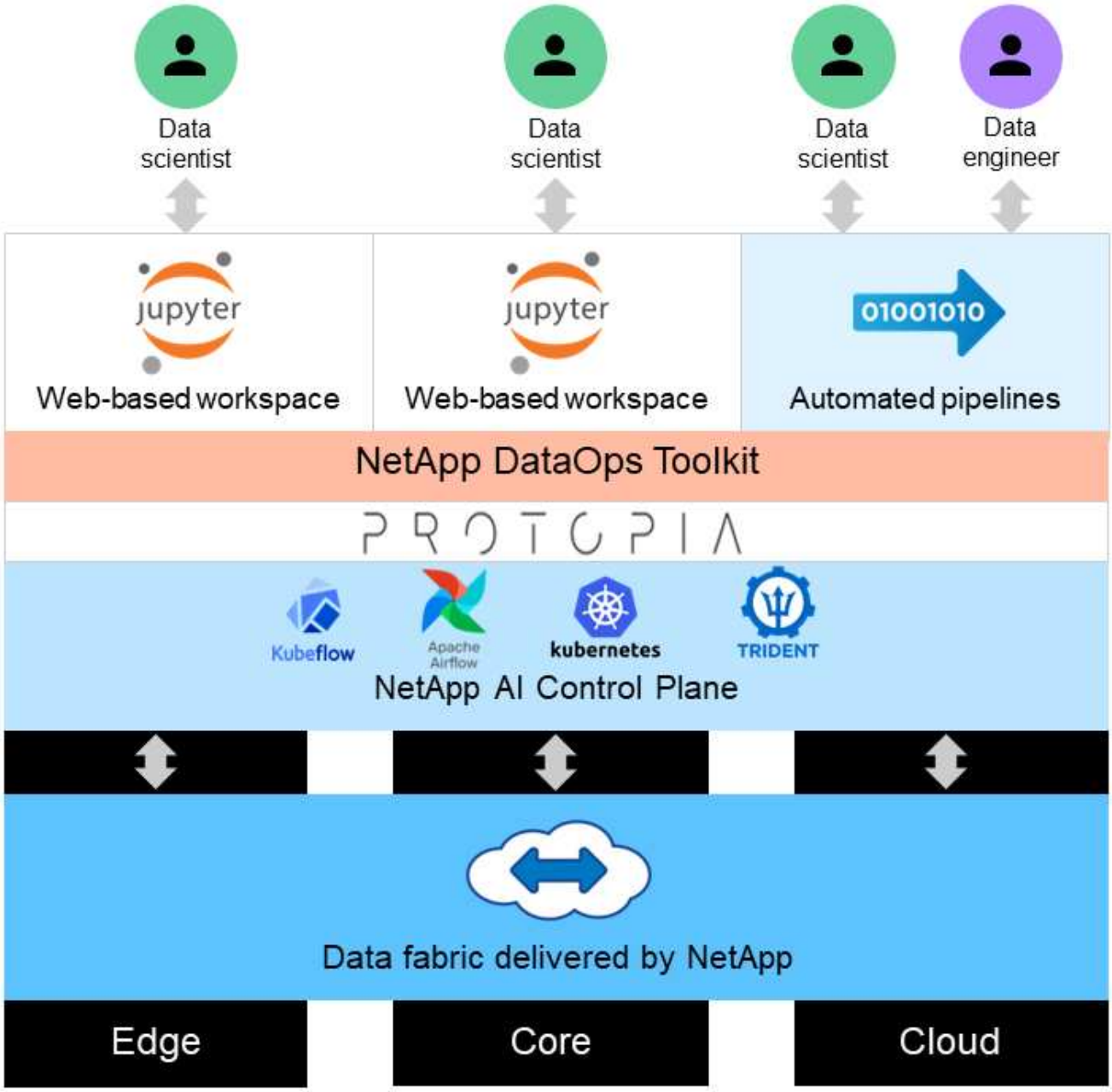
対象

この技術レポートは、次の読者を対象としています。

- 責任ある AI を設計および導入し、公共空間での顔画像処理に関するデータ保護とプライバシーの問題に対処したいビジネス リーダーおよびエンタープライズ アーキテクト。
- プライバシーの保護と保全を目的とするデータ サイエンティスト、データ エンジニア、AI/機械学習 (ML) 研究者、AI/ML システムの開発者。
- GDPR、CCPA、国防総省 (DoD) および政府機関のプライバシー法などの規制基準に準拠した AI/ML モデルおよびアプリケーションのデータ難読化ソリューションを設計するエンタープライズ アーキテクト。
- 機密情報を保護するディープラーニング (DL) および AI/ML/DL 推論モデルを効率的に導入する方法を探しているデータ サイエンティストと AI エンジニア。
- エッジ推論モデルの展開と管理を担当するエッジ デバイス マネージャーとエッジ サーバー管理者。

ソリューションアーキテクチャ

このソリューションは、従来の CPU と並行して GPU の処理能力を活用し、大規模なデータセットに対するリアルタイムおよびバッチ推論 AI ワークロードを処理するように設計されています。この検証は、責任ある AI 導入を目指す組織に必要な、ML のプライバシー保護推論と最適なデータ管理を実証しています。このソリューションは、オンプレミスの中核にあるNetApp ONTAP AI、 NetApp DataOps Toolkit、および Jupyter Lab と CLI インターフェイスを使用した Protopia 難読化ソフトウェアと相互接続された、エッジおよびクラウドコンピューティング用の単一またはマルチノードの Kubernetes プラットフォームに適したアーキテクチャを提供します。次の図は、DataOps Toolkit と Protopia を搭載したNetAppによるデータ ファブリックの論理アーキテクチャの概要を示しています。



Protopia 難読化ソフトウェアは、 NetApp DataOps Toolkit 上でシームレスに実行され、ストレージ サーバーから送信される前にデータを変換します。

ソリューション領域

デジタル画像処理には多くの利点があり、多くの組織が視覚表現に関連するデータを最大限に活用できるようになります。このNetAppと Protopia のソリューションは、ML/DL ライフサイクル全体にわたって AI/ML データを保護し、プライベート化するための独自の AI 推論設計を提供します。これにより、顧客は機密データの所有権を保持し、プライバシーに関する懸念を軽減することでパブリック クラウドまたはハイブリッド クラウドの導入モデルを使用して拡張性と効率性を高め、エッジで AI 推論を導入できるようになります。

環境情報

環境危険の分野で、業界が地理空間分析を活用できる方法は数多くあります。政府や公共事業局は、公衆衛生や気象状況に関する実用的な洞察を得て、パンデミックや山火事などの自然災害の際に国民に適切なアドバイスを提供できるようになります。たとえば、空港や病院などの公共の場で、感染者のプライバシーを侵害することなく COVID 陽性患者を特定し、関係当局や付近の一般の人々に必要な安全対策を警告することができます。

エッジデバイスウェアラブル

軍隊や戦場では、エッジでの AI 推論をウェアラブル デバイスとして使用し、兵士の健康状態を追跡し、運転者の行動を監視し、兵士のプライバシーを維持し保護しながら、軍用車両に近づく際の安全性と関連するリスクについて当局に警告することができます。軍隊の未来は、戦場のモノのインターネット (IoBT) と軍事用のモノのインターネット (IoMT) によってハイテク化しており、兵士が敵を識別し、迅速なエッジ コンピューティングを使用して戦闘でより優れたパフォーマンスを発揮できるように支援するウェアラブル戦闘装備が登場しています。ドローンやウェアラブル機器などのエッジデバイスから収集された視覚データを保護し、保存することは、ハッカーや敵を寄せ付けないために非常に重要です。

非戦闘員避難作戦

非戦闘員避難作戦（NEO）は、生命の危険にさらされている米国市民および米国国民、国防総省の民間人、および指定された人物（ホスト国（HN）および第三国国民（TCN））を適切な安全な避難場所へ避難させるのを支援するために国防総省によって実施されます。実施されている行政管理では、主に手作業による避難者の審査プロセスが採用されています。ただし、高度に自動化された AI/ML ツールと AI/ML ビデオ難読化テクノロジーを組み合わせることで、避難者の識別、避難者の追跡、脅威のスクリーニングの精度、セキュリティ、速度が向上する可能性があります。

ヘルスケアとバイオメディカル研究

画像処理は、コンピューター断層撮影 (CT) や磁気共鳴画像 (MRI) から取得した 3D 画像から病理を診断し、手術計画を立てるために使用されます。 HIPAA プライバシー ルールは、すべての個人情報や写真などのデジタル画像について、組織がデータを収集、処理、消去する方法を規定しています。 HIPAA セーフ ハーバー規制に基づいてデータを共有可能にするには、顔全体の写真画像およびそれに類似する画像を削除する必要があります。構造CT/MR画像から個人の顔の特徴を隠すために使用される匿名化や頭蓋骨除去アルゴリズムなどの自動化技術は、生物医学研究機関のデータ共有プロセスの不可欠な部分になっています。

AI/ML分析のクラウド移行

従来、エンタープライズ顧客は AI/ML モデルをオンプレミスでトレーニングおよび展開してきました。規模の経済と効率性の理由から、これらの顧客は AI/ML 機能をパブリック、ハイブリッド、またはマルチクラウドのクラウド展開に移行するように拡大しています。ただし、他のインフラストラクチャに公開できるデータには制限があります。 NetAppソリューションは、次のようなあらゆるサイバーセキュリティの脅威に対応します。 "データ保護"セキュリティ評価も実行し、Protopia のデータ変換と組み合わせることで、画像処理 AI/ML ワークロードをクラウドに移行する際のリスクを最小限に抑えます。

他の業界におけるエッジコンピューティングとAI推論のさらなるユースケースについては、以下を参照してください。"TR-4886 エッジにおけるAI推論" NetApp AIブログ "知性 vs プライバシー"。

技術概要

このセクションでは、このソリューションを完了するために必要なさまざまな技術コン

ポーネントの概要を説明します。

プロトピア

Protopia AI は、今日の市場において、目立たない、機密推論のためのソフトウェアのみのソリューションを提供しています。 Protopia ソリューションは、機密情報の漏洩を最小限に抑えることで、推論サービスに比類のない保護を提供します。 AI には、手元のタスクを実行するために本当に必要なデータ レコードの情報のみが与えられ、それ以上の情報は与えられません。ほとんどの推論タスクでは、すべてのデータ レコードに存在するすべての情報が使用されるわけではありません。 AI が画像、音声、ビデオ、あるいは構造化された表形式のデータを使用するかどうかに関係なく、Protopia は推論サービスに必要なものだけを提供します。特許取得済みのコアテクノロジーは、数学的にキュレーションされたノイズを使用してデータを確率的に変換し、特定の ML サービスに必要のない情報を改ざんします。このソリューションはデータをマスクするのではなく、厳選されたランダムノイズを使用してデータ表現を変更します。

Protopia ソリューションは、モデルの機能に関する入力特徴空間内の関連情報を保持しながら、表現を変更する問題を勾配ベースの摂動最大化法として定式化します。この検出プロセスは、ML モデルのトレーニングの最後に微調整パスとして実行されます。パスによって確率分布のセットが自動的に生成された後、低オーバーヘッドのデータ変換によってこれらの分布からのノイズ サンプルがデータに適用され、推論のためにモデルに渡される前に難読化されます。

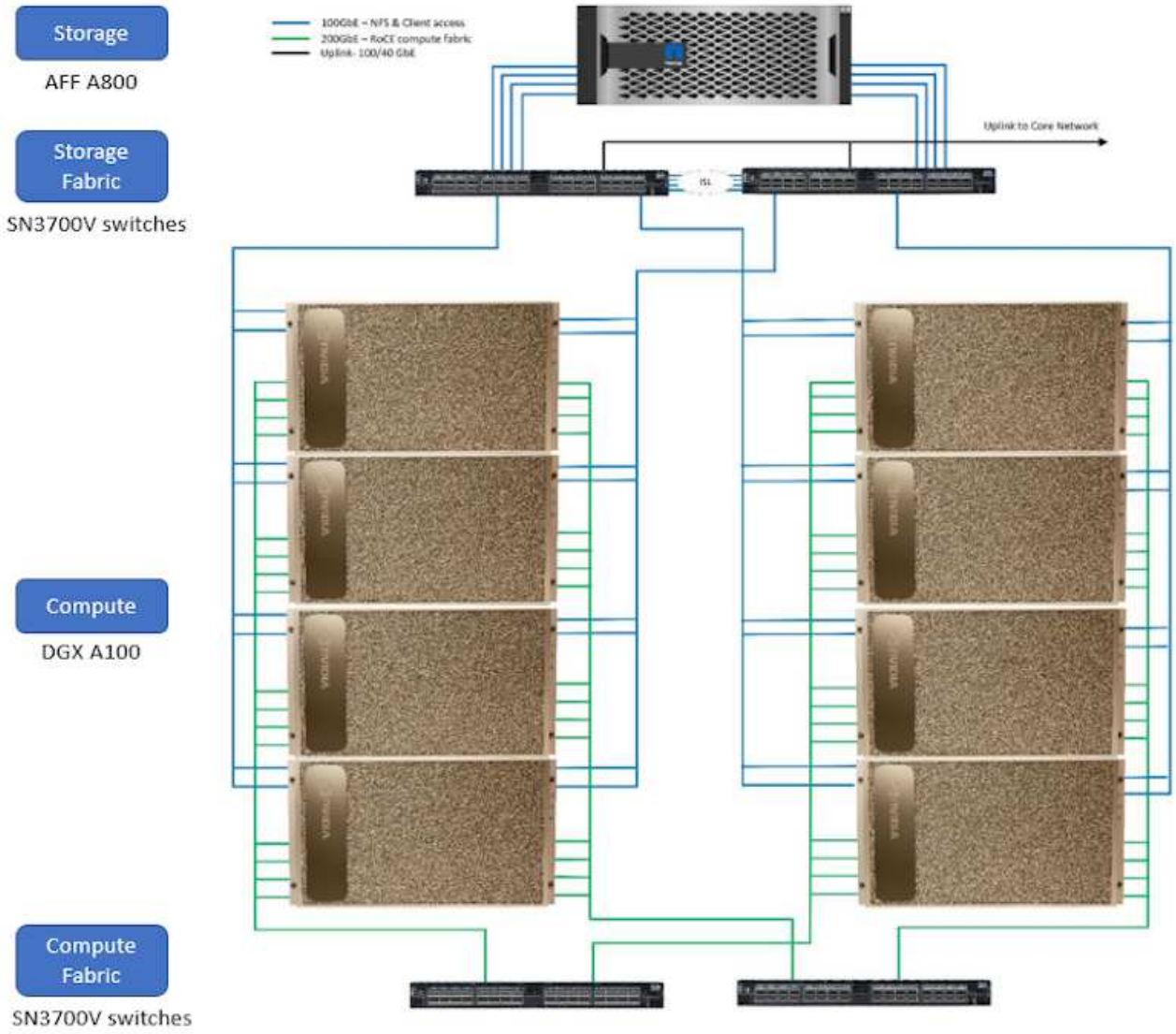
NetApp ONTAP AI

DGX A100 システムとNetAppクラウド接続ストレージ システムを搭載したNetApp ONTAP AI リファレンス アーキテクチャは、 NetAppとNVIDIAによって開発および検証されました。これにより、IT 組織に次の利点をもたらすアーキテクチャが提供されます。

- 設計の複雑さを排除
- コンピューティングとストレージの独立したスケーリングが可能
- 顧客が小規模から始めてシームレスに拡張できるようにする
- さまざまなパフォーマンスとコストポイントに応じた幅広いストレージオプションを提供

ONTAP AI は、DGX A100 システムとNetApp AFF A800ストレージ システムを最先端のネットワークと緊密に統合します。 ONTAP AI は、設計の複雑さと推測を排除することで AI の導入を簡素化します。お客様は、エッジからコア、クラウドに至るまでデータをインテリジェントに管理しながら、小規模から始めて中断なく拡張することができます。

次の図は、DGX A100 システムを使用したONTAP AI ファミリ ソリューションのいくつかのバリエーションを示しています。 AFF A800システムのパフォーマンスは、最大 8 台の DGX A100 システムで検証されます。 ONTAPクラスタにストレージ コントローラ ペアを追加することで、アーキテクチャを複数のラックに拡張し、多数の DGX A100 システムとペタバイト単位のストレージ容量を線形パフォーマンスでサポートできるようになります。このアプローチは、使用される DL モデルのサイズと必要なパフォーマンス メトリックに基づいて、コンピューティングとストレージの比率を個別に変更できる柔軟性を提供します。



ONTAP AIの詳細については、以下を参照してください。 "NVA-1153: NVIDIA DGX A100 システムと Mellanox Spectrum Ethernet スイッチを搭載したNetApp ONTAP AI。"

NetApp ONTAP

NetAppの最新世代のストレージ管理ソフトウェアであるONTAP 9.11 により、企業はインフラストラクチャを最新化し、クラウド対応のデータセンターに移行できるようになります。 ONTAP は業界をリードするデータ管理機能を活用し、データの保存場所に関係なく、単一のツール セットでデータの管理と保護を可能にします。また、エッジ、コア、クラウドなど、必要な場所にデータを自由に移動することもできます。 ONTAP 9.11 には、データ管理を簡素化し、重要なデータを高速化および保護し、ハイブリッド クラウド アーキテクチャ全体で次世代のインフラストラクチャ機能を有効にする多数の機能が含まれています。

NetApp DataOps ツールキット

NetApp DataOps Toolkit は、開発者、データ サイエンティスト、DevOps エンジニア、データ エンジニアが、新しいデータ ボリュームまたは JupyterLab ワークスペースのほぼ瞬時のプロビジョニング、データ ボリュームまたは JupyterLab ワークスペースのほぼ瞬時のクローン作成、追跡可能性やベースライン設定のためのデータ ボリュームまたは JupyterLab ワークスペースのほぼ瞬時のスナップショット作成など、さまざまなデータ管理タスクを簡単に実行できるようにする Python ライブラリです。この Python ライブラリは、コマ

ンドライン ユーティリティとして、または任意の Python プログラムや Jupyter ノートブックにインポートできる関数のライブラリとして機能します。

NVIDIA Triton 推論サーバー

NVIDIA Triton Inference Server は、モデルの展開と実行を標準化し、実稼働環境で高速かつスケーラブルな AI を実現するのに役立つオープンソースの推論サービス ソフトウェアです。 Triton Inference Server は、チームが GPU または CPU ベースのインフラストラクチャ上の任意のフレームワークからトレーニング済みの AI モデルを展開、実行、拡張できるようにすることで、AI 推論を効率化します。 Triton Inference Server は、TensorFlow、 NVIDIA TensorRT、PyTorch、MXNet、OpenVINO など、すべての主要なフレームワークをサポートしています。 Triton は Kubernetes と統合されており、主要なパブリック クラウド AI および Kubernetes プラットフォームで使用できるオーケストレーションとスケーリングを実現します。また、多くの MLOps ソフトウェア ソリューションとも統合されています。

パイトーチ

"パイトーチ"オープンソースの ML フレームワークです。 GPU と CPU を使用するディープラーニング向けに最適化されたテンソル ライブラリです。 PyTorch パッケージには、他の便利なユーティリティの中でも、テンソルを効率的にシリアル化するための多くのユーティリティを提供する多次元テンソルのデータ構造が含まれています。また、コンピューティング機能を備えたNVIDIA GPU でテンソル計算を実行できるようにする CUDA 対応機能も備えています。この検証では、OpenCV-Python (cv2) ライブラリを使用して、Python の最も直感的なコンピューター ビジョンの概念を活用しながらモデルを検証します。

データ管理を簡素化

データ管理は、AI アプリケーションと AI/ML データセットのトレーニングに適切なリソースが使用されるように、企業の IT 運用とデータ サイエンティストにとって非常に重要です。 NetAppテクノロジーに関する次の追加情報は、この検証の範囲外ですが、導入によっては関連する可能性があります。

ONTAPデータ管理ソフトウェアには、運用を合理化および簡素化し、総運用コストを削減するための次の機能が含まれています。

- インライン データ圧縮と拡張重複排除。データ圧縮によりストレージ ブロック内の無駄なスペースが削減され、重複排除により実効容量が大幅に増加します。これは、ローカルに保存されたデータとクラウドに階層化されたデータに適用されます。
- 最小、最大、および適応型サービス品質 (AQoS)。きめ細かなサービス品質 (QoS) 制御により、高度に共有された環境における重要なアプリケーションのパフォーマンス レベルを維持できます。
- NetAppFabricPool。 Amazon Web Services (AWS)、Azure、 NetApp StorageGRIDストレージ ソリューションなどのパブリックおよびプライベート クラウド ストレージ オプションへのコールド データの自動階層化を提供します。 FabricPoolの詳細については、以下を参照してください。 "TR-4598: FabricPool のベストプラクティス" 。

データの高速化と保護

ONTAP は優れたレベルのパフォーマンスとデータ保護を提供し、これらの機能を次のように拡張します。

- パフォーマンスと低レイテンシ。 ONTAP は、可能な限り低いレイテンシで最高のスループットを提供します。
- データ保護：ONTAP は、すべてのプラットフォームにわたる共通管理を備えた組み込みのデータ保護機能を提供します。

- NetAppボリューム暗号化 (NVE)。 ONTAP は、オンボードと外部キー管理の両方をサポートするネイティブのボリューム レベルの暗号化を提供します。
- マルチテナントと多要素認証。 ONTAP は、最高レベルのセキュリティでインフラストラクチャ リソースを共有できるようにします。

将来を見据えたインフラ

ONTAP は、次の機能により、要求が厳しく常に変化するビジネス ニーズへの対応に役立ちます。

- シームレスなスケーリングと中断のない運用。 ONTAP は、既存のコントローラおよびスケールアウト クラスタへの無停止の容量追加をサポートします。お客様は、コストのかかるデータ移行や停止なしで、NVMe や 32Gb FC などの最新テクノロジーにアップグレードできます。
- クラウド接続。 ONTAP は、すべてのパブリック クラウドでソフトウェア定義ストレージ (ONTAP Select) とクラウド ネイティブ インスタンス (Google Cloud NetApp Volumes) のオプションを備えた、最もクラウドに接続されたストレージ管理ソフトウェアです。
- 新しいアプリケーションとの統合。 ONTAP は、既存のエンタープライズ アプリケーションをサポートするのと同じインフラストラクチャを使用して、自律走行車、スマート シティ、インダストリー 4.0 などの次世代プラットフォームとアプリケーション向けにエンタープライズ グレードのデータ サービスを提供します。

NetApp Astraコントロール

NetApp Astra製品ファミリーは、 NetAppのストレージおよびデータ管理テクノロジーを活用し、オンプレミスおよびパブリック クラウドの Kubernetes アプリケーション向けのストレージおよびアプリケーション対応のデータ管理サービスを提供します。これにより、Kubernetes アプリケーションを簡単にバックアップし、データを別のクラスターに移行し、実用的なアプリケーションのクローンを即座に作成できるようになります。パブリッククラウドで実行されるKubernetesアプリケーションを管理する必要がある場合は、 "Astraコントロールサービス" 。 Astra Control Service は、Google Kubernetes Engine (GKE) および Azure Kubernetes Service (AKS) 内の Kubernetes クラスターのアプリケーション対応データ管理を提供する、 NetAppが管理するサービスです。

NetAppTrident

Astra "Trident"NetAppの は、永続ストレージの作成、管理、および使用を簡素化する、Docker および Kubernetes 向けのオープンソースの動的ストレージ オーケストレーターです。 Kubernetes ネイティブ アプリケーションであるTridentは、Kubernetes クラスター内で直接実行されます。 Trident を使用すると、顧客は DL コンテナ イメージをNetAppストレージにシームレスに導入でき、AI コンテナの導入にエンタープライズ グレードのエクスペリエンスを提供できます。 Kubernetes ユーザー (ML 開発者、データ サイエンティストなど) は、オーケストレーションとクローンを作成、管理、自動化して、 NetAppテクノロジーを活用した高度なデータ管理機能を活用できます。

NetApp BlueXPコピーと同期

"BlueXPコピーと同期"高速かつ安全なデータ同期を実現するNetAppサービスです。オンプレミスの NFS または SMB ファイル共有、 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、または IBM Cloud Object Storage の間でファイルを転送する必要がある場合でも、 BlueXP Copy and Sync を使用すると、必要な場所にファイルを迅速かつ安全に移動します。データが転送されると、ソースとターゲットの両方で完全に使用できるようになります。 BlueXP Copy and Syncc は、事前に定義されたスケジュールに基づいてデータを継続的に同期し、差分のみを移動するため、データ複製にかかる時間とコストが最小限に抑えられます。 BlueXP Copy and Sync は、セットアップと

使用が非常に簡単なソフトウェア アズ ア サービス (SaaS) ツールです。 BlueXP Copy and Sync によってトリガーされるデータ転送は、データ ブローカーによって実行されます。 BlueXPコピーおよび同期データブローカーは、AWS、Azure、Google Cloud Platform、またはオンプレミスにデプロイできます。

NetApp BlueXP分類

強力なAIアルゴリズムによって駆動され、 "NetApp BlueXP分類"データ資産全体にわたって自動制御とデータガバナンスを提供します。コスト削減箇所を簡単に特定し、コンプライアンスとプライバシーに関する懸念を特定し、最適化の機会を見つけることができます。 BlueXP分類ダッシュボードを使用すると、重複データを識別して冗長性を排除し、個人データ、非個人データ、機密データをマッピングし、機密データと異常に関するアラートをオンにするための洞察が得られます。

テストおよび検証計画

このソリューション設計では、次の 3 つのシナリオが検証されました。

- NetApp DataOps Toolkit for Kubernetes を使用してオーケストレーションされた JupyterLab ワークスペース内の推論タスク（Protopia 難読化ありとなし）。
- NetApp DataOps Toolkit for Kubernetes を使用してオーケストレーションされたデータ ボリュームを持つ Kubernetes 上のバッチ推論ジョブ (Protopia 難読化ありとなし)。
- NetApp DataOps Toolkit for Kubernetes を使用してオーケストレーションされたNVIDIA Triton Inference Server インスタンスを使用した推論タスク。 Triton 推論 API を呼び出す前に、画像に Protopia 難読化を適用し、ネットワーク経由で送信されるすべてのデータは難読化する必要があるという一般的な要件をシミュレートしました。このワークフローは、データが信頼できるゾーン内で収集されるが、推論のためにその信頼できるゾーンの外部に渡す必要があるユースケースに適用できます。 Protopia の難読化がなければ、機密データが信頼できるゾーンから出ることなく、このタイプのワークフローを実装することはできません。

テスト構成

次の表は、ソリューション設計の検証環境の概要を示しています。

コンポーネント	**version**
Kubernetes	1.21.6
NetApp Trident CSI ドライバー	22.01.0
Kubernetes 用NetApp DataOps ツールキット	2.3.0
NVIDIA Triton 推論サーバー	21.11-py3

テスト手順

このセクションでは、検証を完了するために必要なタスクについて説明します。

前提条件

このセクションで説明するタスクを実行するには、次のツールがインストールおよび構成された Linux または macOS ホストにアクセスできる必要があります。

- Kubectl (既存の Kubernetes クラスターにアクセスできるように構成)
  - インストールと設定の手順については、 "ここをクリックしてください。" 。
- Kubernetes 用NetApp DataOps ツールキット
  - インストール手順については、 "ここをクリックしてください。" 。

シナリオ1 – JupyterLabでのオンデマンド推論

1. AI/ML 推論ワークロード用の Kubernetes 名前空間を作成します。

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

2. NetApp DataOps Toolkit を使用して、推論を実行するデータを保存するための永続ボリュームをプロビジョニングします。

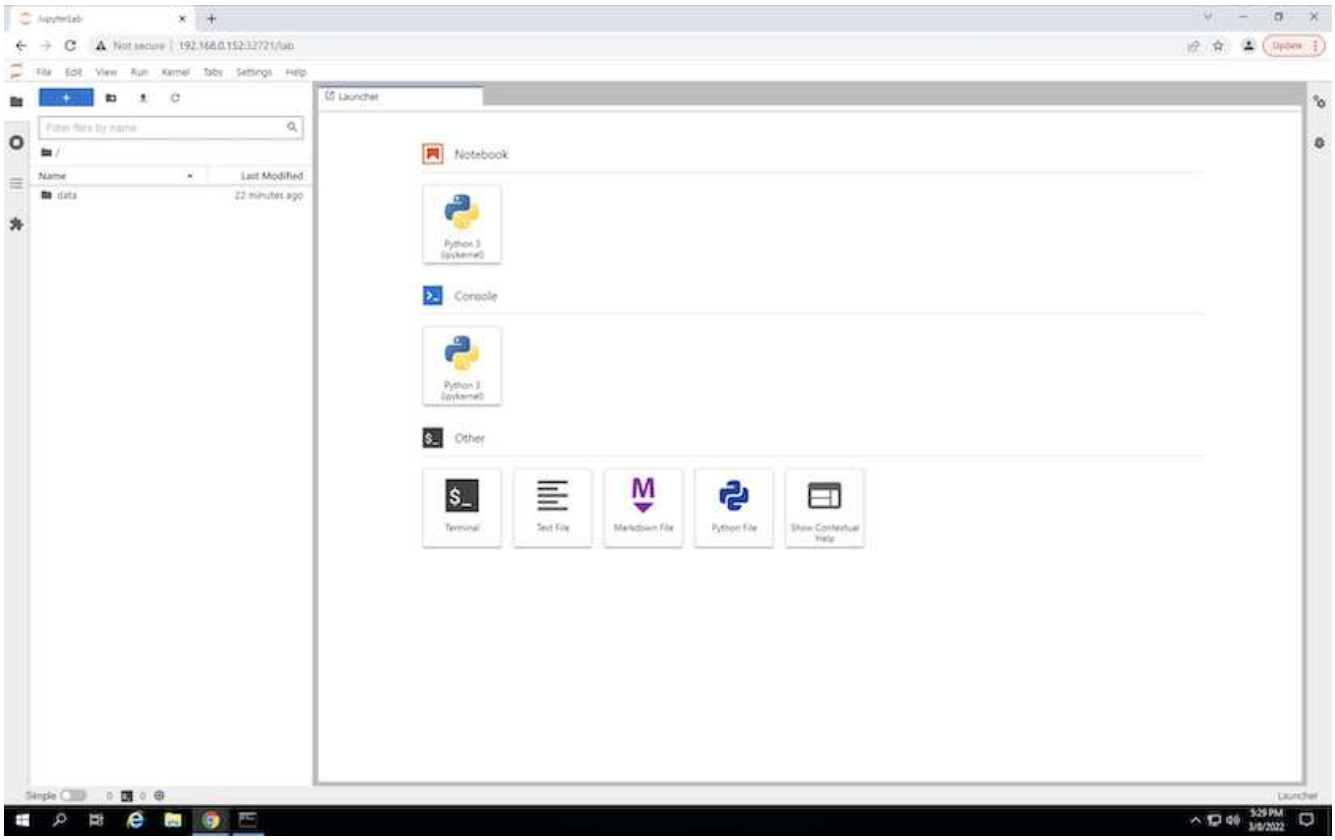
```
$ netapp_dataops_k8s_cli.py create volume --namespace=inference --pvc
-name=inference-data --size=50Gi
Creating PersistentVolumeClaim (PVC) 'inference-data' in namespace
'inference'.
PersistentVolumeClaim (PVC) 'inference-data' created. Waiting for
Kubernetes to bind volume to PVC.
Volume successfully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

3. NetApp DataOps Toolkit を使用して、新しい JupyterLab ワークスペースを作成します。前の手順で作成した永続ボリュームをマウントするには、 `--mount- pvc`オプション。必要に応じて、ワークスペースにNVIDIA GPUを割り当てます。 `-- nvidia-gpu`オプション。

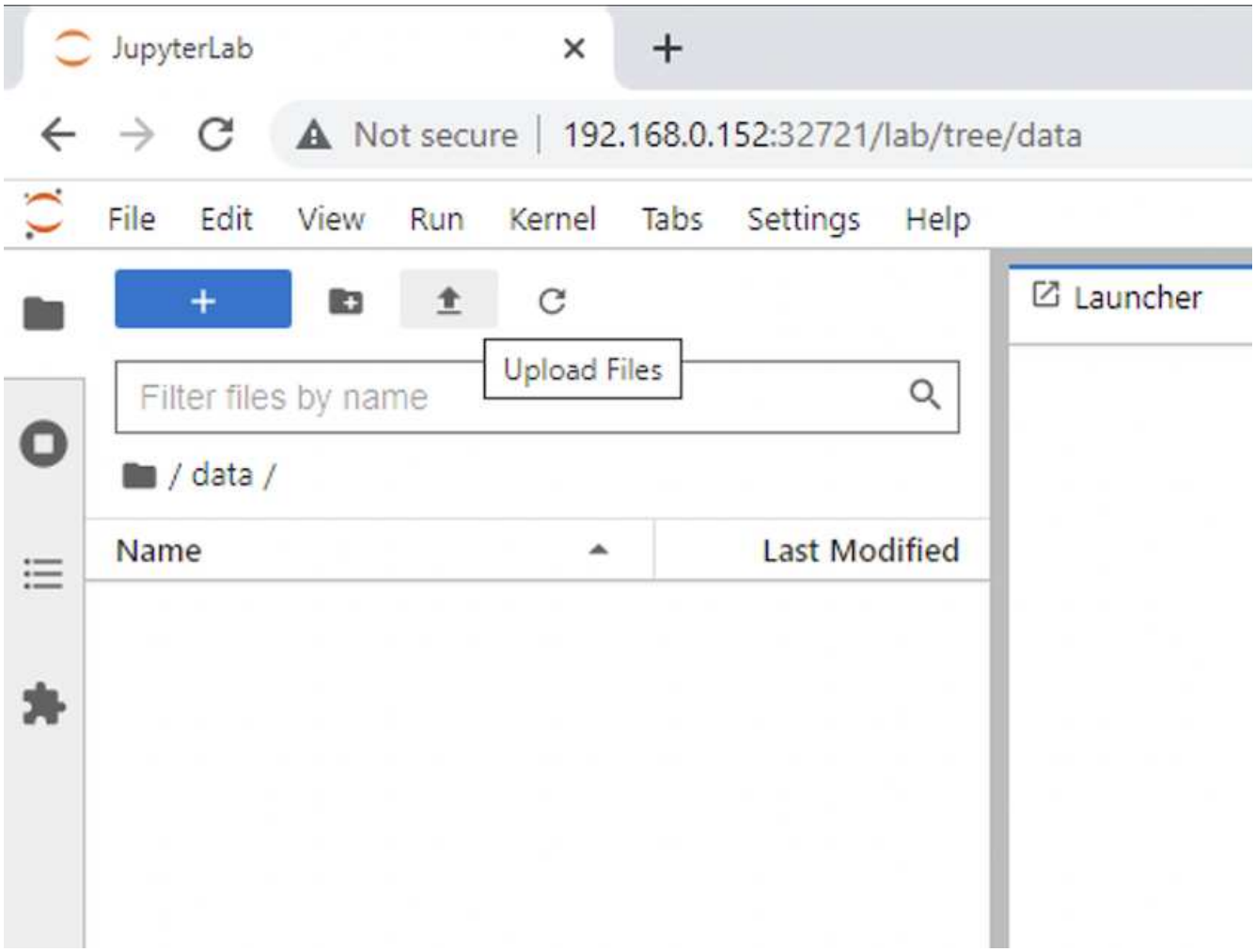
   次の例では、永続ボリューム `inference-data`JupyterLabワークスペースコンテナにマウントされます。 `/home/jovyan/data` 。公式のProject Jupyterコンテナイメージを使用する場合、 `/home/jovyan` JupyterLab Web インターフェース内の最上位ディレクトリとして表示されます。

```
$ 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
```

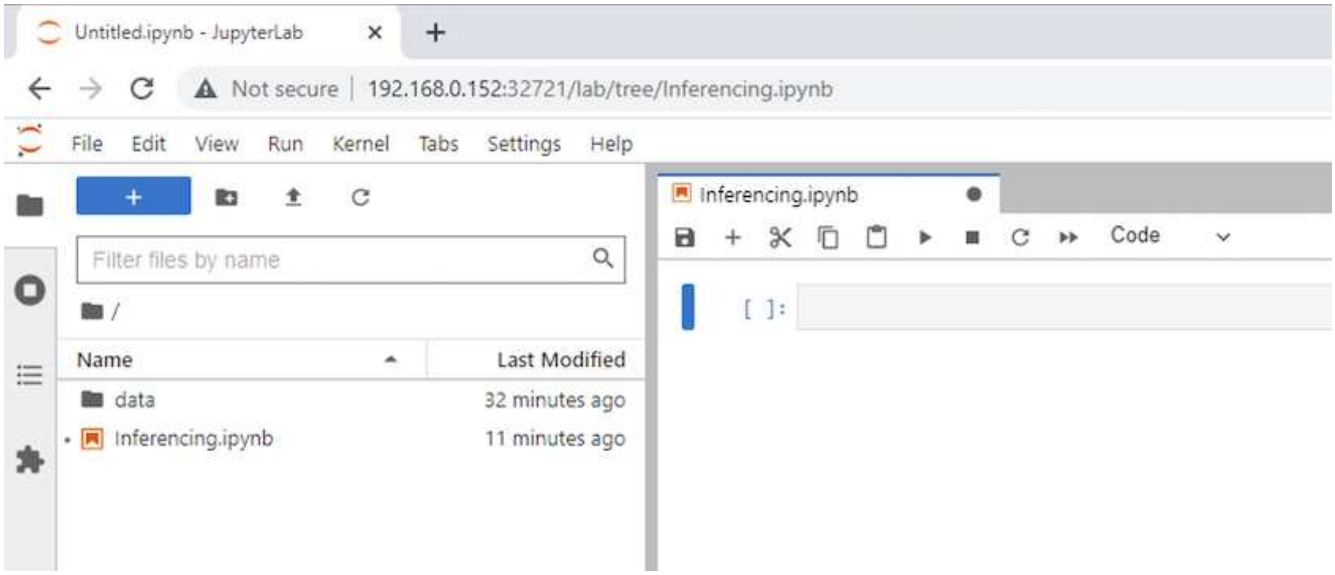
4. 出力に指定されたURLを使用してJupyterLabワークスペースにアクセスします。 `create jupyterlab`指示。データ ディレクトリは、ワークスペースにマウントされた永続ボリュームを表します。



5. 開く `data`ディレクトリを作成し、推論を実行するファイルをアップロードします。ファイルがデータ ディレクトリにアップロードされると、ワークスペースにマウントされた永続ボリュームに自動的に保存されます。ファイルをアップロードするには、次の画像に示すように、「ファイルのアップロード」アイコンをクリックします。



6. 最上位ディレクトリに戻り、新しいノートブックを作成します。



7. 推論コードをノートブックに追加します。次の例は、画像検出ユースケースの推論コードを示しています。

Launcher | image-demo-pytorch.ipynb

Code | Python 3 (ipykernel)

STEP 3-1: Clean (Without obfuscation) detection

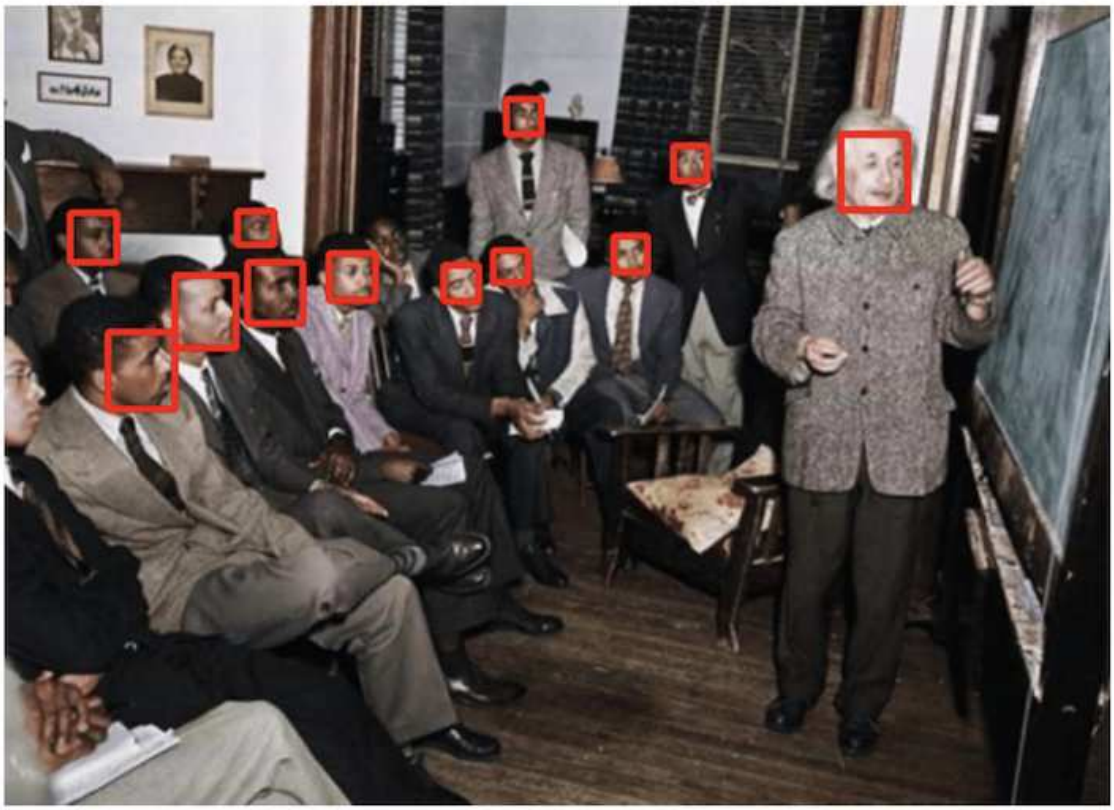
[9]:
```
# get current frame
frame = input_image

# 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. 推論コードに Protopia 難読化を追加します。 Protopia は顧客と直接連携してユースケース固有のドキュメントを提供しており、この技術レポートの範囲外です。次の例は、Protopia 難読化が追加された画像検出ユースケースの推論コードを示しています。

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)
```

Visualize Protopia AI (With obfuscation) detection

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

シナリオ2 – Kubernetesでのバッチ推論

1. AI/ML 推論ワークロード用の Kubernetes 名前空間を作成します。

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

2. NetApp DataOps Toolkit を使用して、推論を実行するデータを保存するための永続ボリュームをプロビジョニングします。

```
$ 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. 新しい永続ボリュームに推論を実行するデータを入力します。

   PVC にデータをロードする方法はいくつかあります。データが現在NetApp StorageGRIDやAmazon S3などのS3互換オブジェクトストレージプラットフォームに保存されている場合は、 "NetApp DataOps Toolkit S3 データムーバーの機能" 。もう1つの簡単な方法は、JupyterLabワークスペースを作成し、JupyterLabウェブインターフェースからファイルをアップロードすることです。これは、「シナリオ1 – JupyterLabでのオンデマンド推論 。"

4. バッチ推論タスク用の Kubernetes ジョブを作成します。次の例は、画像検出ユースケースのバッチ推論ジョブを示しています。このジョブは、画像セット内の各画像に対して推論を実行し、推論精度メトリックを stdout に書き込みます。

```
$ 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. 推論ジョブが正常に完了したことを確認します。

```
$ 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. 推論ジョブに Protopia 難読化を追加します。 Protopia から直接 Protopia 難読化を追加するためのユースケース固有の手順は、この技術レポートの範囲外です。次の例は、ALPHA 値 0.8 を使用して Protopia 難読化を追加した顔検出ユースケースのバッチ推論ジョブを示しています。このジョブは、画像セット内の各画像に対して推論を実行する前に Protopia 難読化を適用し、推論精度メトリックを標準出力に書き込みます。

   この手順を ALPHA 値 0.05、0.1、0.2、0.4、0.6、0.8、0.9、0.95 に対して繰り返しました。結果は以下でご覧いただけます"推論精度の比較。"

```
$ 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. 推論ジョブが正常に完了したことを確認します。

```
$ 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
```

シナリオ3 – NVIDIA Triton推論サーバー

1. AI/ML 推論ワークロード用の Kubernetes 名前空間を作成します。

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

2. NetApp DataOps Toolkit を使用して、 NVIDIA Triton Inference Server のモデル リポジトリとして使用する永続ボリュームをプロビジョニングします。

```
$ 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. モデルを新しい永続ボリュームに保存します。 "形式" NVIDIA Triton 推論サーバーによって認識されます。

   PVC にデータをロードする方法はいくつかあります。簡単な方法は、JupyterLabワークスペースを作成し、JupyterLabウェブインターフェースからファイルをアップロードすることです。これは、「シナリ

オ1 – JupyterLabでのオンデマンド推論 。"

4. NetApp DataOps Toolkit を使用して、新しいNVIDIA Triton Inference Server インスタンスを展開します。

```
$ 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. Triton クライアント SDK を使用して推論タスクを実行します。次の Python コードの抜粋では、Triton Python クライアント SDK を使用して、顔検出ユースケースの推論タスクを実行します。この例では、Triton API を呼び出して推論用の画像を渡します。その後、Triton 推論サーバーはリクエストを受信し、モデルを呼び出し、推論出力を API 結果の一部として返します。

```
# 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. 推論コードに Protopia 難読化を追加します。 Protopia から直接 Protopia 難読化を追加するためのユースケース固有の手順は見つかりますが、このプロセスはこの技術レポートの範囲外です。次の例は、前の手順 5 で示したものと同じ Python コードですが、Protopia の難読化が追加されています。

画像が Triton API に渡される前に、Protopia 難読化が画像に適用されることに注意してください。したがって、難読化されていないイメージはローカル マシンから外に出ることはありません。難読化された画像のみがネットワークを介して渡されます。このワークフローは、データが信頼できるゾーン内で収集されるが、推論のためにその信頼できるゾーンの外部に渡す必要があるユースケースに適用できます。Protopia の難読化がなければ、機密データが信頼できるゾーンから出ることなく、このタイプのワークフローを実装することはできません。

```
# 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)
```

推論精度の比較

この検証では、生の画像セットを使用して、画像検出ユースケースの推論を実行しました。次に、推論前に Protopia 難読化を追加した同じ画像セットに対して、同じ推論タスクを実行しました。 Protopia 難読化コンポーネントの ALPHA の異なる値を使用してタスクを繰り返しました。 Protopia 難読化のコンテキストでは、ALPHA 値は適用される難読化の量を表し、ALPHA 値が高いほど難読化のレベルが高くなります。次に、これらの異なる実行間で推論の精度を比較しました。

次の 2 つの表は、ユースケースの詳細と結果の概要を示しています。

Protopia は顧客と直接連携し、特定のユースケースに適切な ALPHA 値を決定します。

コンポーネント	詳細
モデル	フェイスボックス（PyTorch） -
データセット	FDDBデータセット

プロトピアの難読化	アルファ	正確さ
いいえ	該当なし	0.9337148153739079
はい	0.05	0.9028766627325002
はい	0.1	0.9024301009661478
はい	0.2	0.9081836283186224
はい	0.4	0.9073066107482036
はい	0.6	0.8847816568680239
はい	0.8	0.8841195749171925
はい	0.9	0.8455427675252052
はい	0.95	0.8455427675252052

難読化速度

この検証では、1920 x 1080 ピクセルの画像に Protopia 難読化を 5 回適用し、そのたびに難読化手順が完了するまでにかかった時間を測定しました。

難読化を適用するために、単一のNVIDIA V100 GPU 上で実行される PyTorch を使用し、実行の合間に GPU キャッシュをクリアしました。難読化ステップは、5 回の実行でそれぞれ 5.47 ミリ秒、5.27 ミリ秒、4.54 ミリ秒、5.24 ミリ秒、4.84 ミリ秒かかりました。平均速度は5.072msでした。

まとめ

データは、保存中、転送中、計算中の 3 つの状態で存在します。あらゆる AI 推論サービスの重要な部分は、プロセス全体を通じてデータを脅威から保護することです。推論プロセスでは外部の顧客と推論サービスを提供する企業の両方に関する個人情報が公開される可能性があるため、推論中のデータの保護は重要です。 Protopia AI は、今日の市場における機密 AI 推論のための、目立たないソフトウェアのみのソリューションです。Protopia では、AI には、現在の AI/ML タスクを実行するために不可欠なデータ レコード内の変換された情報のみが供給され、それ以上は供給されません。この確率的変換はマスキングの形式ではなく、キュレーションされたノイズを使用してデータの表現を数学的に変更することに基づいています。

ONTAP機能を備えたNetAppストレージ システムは、ローカル SSD ストレージと同等以上のパフォーマンスを提供し、 NetApp DataOps Toolkit と組み合わせることで、データ サイエンティスト、データ エンジニア、AI/ML 開発者、ビジネスまたはエンタープライズ IT 意思決定者に次のメリットをもたらします。

- AI システム、分析、その他の重要なビジネス システム間でデータを簡単に共有できます。このデータ共有により、インフラストラクチャのオーバーヘッドが削減され、パフォーマンスが向上し、企業全体のデータ管理が合理化されます。
- コストを最小限に抑え、リソースの使用率を向上させるために、コンピューティングとストレージを独立して拡張できます。
- 統合されたスナップショット コピーとクローンを使用して合理化された開発および展開ワークフローにより、瞬時にスペース効率の高いユーザー ワークスペース、統合バージョン管理、および自動展開が実現します。
- 災害復旧、事業継続性、規制要件に対応したエンタープライズ グレードのデータ保護とデータ ガバナンス。
- データ管理操作の呼び出しが簡素化され、Jupyter ノートブックのNetApp DataOps Toolkit からデータ サイエンティストのワークスペースのスナップショット コピーを迅速に取得して、バックアップと追跡が可能になります。

NetAppと Protopia のソリューションは、エンタープライズ グレードの AI 推論の導入に最適な、柔軟なスケールアウト アーキテクチャを提供します。オンプレミスとハイブリッド クラウドの両方の展開において、責任ある AI プラクティスによって機密 AI 推論要件を満たすことができるため、データ保護が可能になり、機密情報のプライバシーが確保されます。

追加情報と謝辞の参照先

このドキュメントに記載されている情報の詳細については、次のドキュメントや Web サイトを参照してください。

- NetApp ONTAPデータ管理ソフトウェア — ONTAP情報ライブラリ

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

- NetAppコンテナ向け永続ストレージ - NetApp Trident

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

- NetApp DataOps ツールキット

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

- NetAppコンテナ向け永続ストレージ - NetApp Trident

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

- プロトピアAI - 機密推論

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

- NetApp BlueXPコピーと同期

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

- NVIDIA Triton 推論サーバー

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

- NVIDIA Triton 推論サーバーのドキュメント

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

- PyTorchのFaceBoxes

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

謝辞

- NetApp主席プロダクトマネージャー、マーク・ケイツ氏
- NetApp 、テクニカル マーケティング エンジニア、Sufian Ahmad 氏
- プロトピアAI最高技術責任者兼教授、ハディ・エスマイルザデ

著作権に関する情報

Copyright © 2025 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用権を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用権については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、http://www.netapp.com/TMに記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。