Scalable and Modular AI Deployment Powered by NVIDIA Clara Deploy

White Paper

WP-09917-001_v01 | April 2020

Authors

This white paper is authored by the following people:

Jesse Tetreault

Rahul Choudhury

Brad Genereaux

Kristopher Kersten

Jiahui Guan

Table of Contents

List of Figures

List of Tables

Introduction

Applications of Artificial Intelligence (AI) are having a profound impact on all walks of life today. The availability of large datasets and graphics processing unit (GPU) based computing have led to breakthroughs in applications that were deemed impossible even a few years ago. The field of medical image analysis is no exception to this trend. Medical image analysis refers to a set of procedures to obtain clinically meaningful information from various imaging modalities, which can then be used to enhance diagnosis and treatment procedures according to the patients' needs.

In the past, one of the popular approaches to analyze medical images was to use machine learning-based algorithms. Machine learning started as a field in computer science to enable algorithms to solve problems without being explicitly programmed. By the end of 1990s, supervised machine learning-based techniques, where labeled training data is used to develop a model, were becoming increasingly popular in medical image analysis. Examples include active shape models for segmentation, atlas methods for organ localization and the use of statistical classifiers such as support vector machines for computer aided detection and diagnosis.

Despite their usefulness for analyzing medical images, traditional machine learning approaches have several limitations. One of the key steps in designing traditional machine learning algorithms is extracting features from datasets. A feature in this context refers to an important, quantifiable aspect of a medical image which is useful for the analysis problem at hand. A major challenge lies in choosing the right features to correctly model a given problem and requires considerable domain expertise in the specific imaging modality. In addition, hand-crafting a feature is often time-consuming. Algorithm developers need to choose from different combinations of features, and degrees of complexity to sufficiently solve a given problem. The difficulty in selecting the right features for medical image analysis often results in brittle system performance during inference time for new imaging datasets. This is one of the main reasons why many machine learning studies have demonstrated great technical potential during the research phase, but only a few have shown actual clinical efficacy.

Recent advancements in deep learning-based artificial intelligence have tremendous potential to remove these hurdles. Deep learning employs multi-layered artificial neural networks to automatically learn the features that optimally represent the imaging data for the problem at hand. This is a major shift in how medical image analysis can be performed - now the burden of feature engineering has shifted from a laborious and brittle manual process to a largely automated process, thus allowing clinical domain-experts to effectively use deep learning for their own research and clinical applications.

Challenges

Although deep learning-based applications have huge potential in the medical imaging space, they can be expensive to design, deploy and maintain. The chasm between deep-learning based research activities in medical imaging and the availability of production-quality medical imaging information systems which make use of such advancements is significant. The following root causes fuel the underlying problems.

Standardized Clinical Imaging Workflows

Modern clinical imaging workflows make use of imaging algorithms, some of which may utilize deep learning-based inference. Multiple such algorithms are often chained together in a pipeline to solve a specific clinical problem. An effective pipeline mechanism would allow decomposing the clinical workflow into elementary activities, defining their control and data flow and allow specification of required computation resources. Today's inference platforms do not offer a standardized way of defining and deploying reusable deep learning-based imaging pipelines, which results in delayed production and higher cost of maintenance.

Scalable Execution of Imaging Workflows

Clinical imaging workflows - when deployed on an inference platform - need a robust queuing and scheduling mechanism so that jobs can be executed efficiently based on the available system resources. This also includes effective utilization of GPUs to get the maximum possible performance during inference. Often these workflows require the ability to support servers that have multiple homogeneous or heterogeneous GPUs. As the number of imaging studies per workflow increases, the inference platform needs to be able to take advantage of additional compute nodes to distribute the increased workload. The lack of a robust inference and workflow execution environment today contributes to the production delay and cost of deploying deep learning-based applications in the hospital enterprise systems.

Maintainable and Reusable Imaging Algorithms

Deep Learning-based imaging algorithms used in clinical workflows are reusable assets for a healthcare organization. The workflows once deployed need to be updated and maintained effectively. However, the healthcare information systems where these applications are deployed are often built on top of a monolithic architecture. Such architecture limits the reuse of components under new contexts. Maintaining operational agility during the repeated deployment of a clinical imaging application, while scaling it at the same time can be a big challenge. Refactoring legacy applications is time consuming and has a huge impact on clinical risk analysis that needs to be performed for any significant change introduced in the system.

Handling Inference Effectively

The type of AI inferences utilized in medical imaging algorithms vary depending on the workflow needs. Post-processing-based workflows typically emphasize throughput over latency. In such scenarios, typically the requirement is to batch hundreds of datasets for inference to achieve optimal throughput on jobs run overnight in a healthcare organization. However, for real-time imaging modalities, large batch sizes also carry a latency penalty. For these usages, smaller batch sizes (as small as a region of interest in a single image or a frame) need to be used, while trading off throughput for lowest latency. A production quality inference platform needs to be adaptive enough to apply different modes of inference based on the underlying needs of the clinical workflows.

Visualizing Results

Making sense of the output from deep learning algorithms in medical imaging often requires use of advanced 2D, 3D, and 4D visualization. Existing 3D visualization systems which are deployed to handle typical clinical workloads are often not designed to handle the outputs from an AI inference platform. In addition, such visualization needs to be built using GPU based hardware for optimal performance. Lack of an ability to meaningfully visualize outputs from AI inference hinders the growth of adoption of AI-driven clinical workflows.

Integration with Healthcare Information Systems

Successful deployment of deep learning-based applications requires meaningful integration with healthcare information systems. The inference platform that is used to execute deep learning-based workflows need to support industry standards such as Digital Imaging and Communications in Medicine (DICOM) for ingestion of input data and publishing of inference results. In addition, some clinical applications may require custom adaptation of modality specific input data before it is used for inference. An inference platform that was not designed with imaging informatics in mind from the ground up often does not have support for healthcare standards and necessary data adaptation capabilities.

NVIDIA Clara Deploy

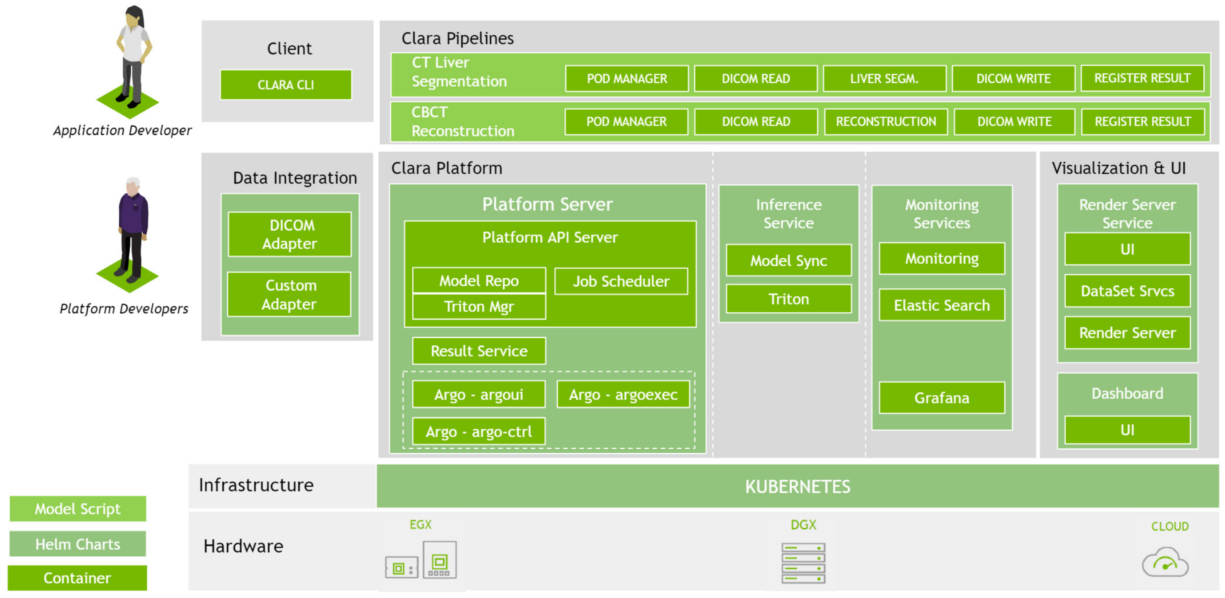
NVIDIA Clara™ Deploy is a reference application framework serving diverse workloads ranging from imaging applications, video batch processing workflows, to genomics. We call this a reference framework because it is built to be customizable. Instrument companies, marketplaces, and startups can build upon it, stand up their healthcare solution or platforms, and decrease their technical engineering debt significantly. By providing these best practices of cloud-native software development, it gives a kick start for developers building production deployment solutions.

In this section we walk through the NVIDIA Clara Deploy platform architecture, range of use cases, and details regarding the underlying components.

NVIDIA Clara Deploy Architecture

The following figure illustrates the NVIDIA Clara Deploy architecture from the underlying hardware to the user-facing tools that facilitate platform configuration and pipeline execution. The modularity of the architecture allows flexibility in defining applications and the ability to reuse components to solve common problems in clinical deployment. The foundation of the platform is Kubernetes orchestration of the deployment and execution of pods on GPU-accelerated compute resources. The choice of an open, cloud-native deployment platform addresses some of the key requirements in deploying medical imaging applications - scalability, resiliency, and the flexibility to deploy on workstations, NVIDIA® NGC-Ready™ for Edge (NVIDIA EGX™) servers, NVIDIA DGX™ platforms, or cloud-based instances.

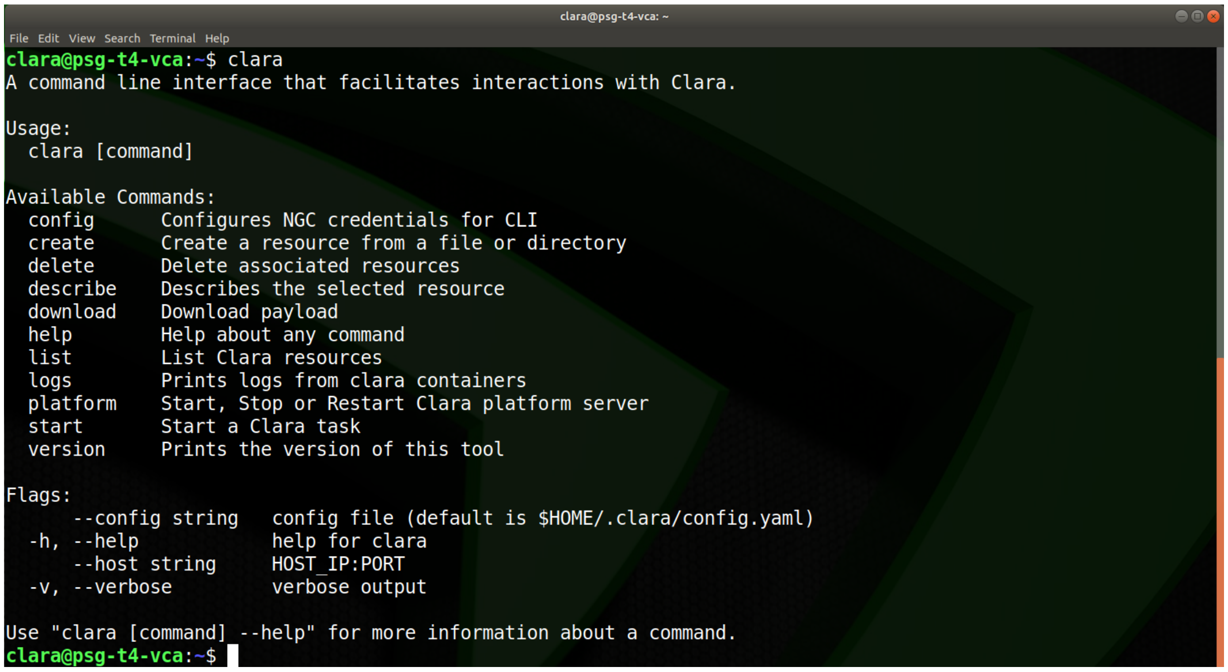
Figure 1. NVIDIA Clara Deploy Architecture



NVIDIA Clara platform provides the core functionality of the NVIDIA Clara Deploy framework, leveraging the underlying Kubernetes infrastructure to manage NVIDIA Clara pipelines and their component containers, payloads, adapters, and services. This includes management of models available to pipelines, priority-based job scheduling, management of Triton inference servers, aggregation of results, as well as health and resource monitoring. The Platform API Server exposes a GRPC-based interface that allows integration with additional first- and third-party services that provide an interface to other systems such as hospital PACS or visualization services.

An NVIDIA Clara pipeline is fundamentally a collection of containers with a defined control flow that executes a medical imaging task or computation. NVIDIA Clara Deploy provides an API that allows definition of these containers and the interface between containers. The platform also provides a DICOM adapter that can be used to integrate with the hospital PACS by receiving DICOM data, defining a pipeline payload, and triggering a pipeline. The results generated by the pipeline execution, or job, are aggregated by the results service and delivered to either external devices or the integrated render server.

Figure 2. NVIDIA Clara Command Line Interface (CLI)



```
clara@psg-t4-vca:~$ clara
A command line interface that facilitates interactions with Clara.

Usage:
  clara [command]

Available Commands:
  config      Configures NGC credentials for CLI
  create      Create a resource from a file or directory
  delete      Delete associated resources
  describe    Describes the selected resource
  download    Download payload
  help        Help about any command
  list        List Clara resources
  logs        Prints logs from clara containers
  platform    Start, Stop or Restart Clara platform server
  start       Start a Clara task
  version     Prints the version of this tool

Flags:
      --config string   config file (default is $HOME/.clara/config.yaml)
  -h, --help            help for clara
      --host string     HOST_IP:PORT
  -v, --verbose         verbose output

Use "clara [command] --help" for more information about a command.
clara@psg-t4-vca:~$
```

All these components of the NVIDIA Clara platform can be controlled using the Clara command line interface (CLI). The CLI provides functionality to interact with the framework such as creating, deleting, and managing pipelines, as well as starting and stopping individual components of the platform. The CLI package contains the `clara`, `clara-dicom`, `clara-render`, `clara-console`, `clara-pull` and `clara-monitor` binaries.

Use Cases and Workflows

NVIDIA Clara Deploy currently enables several different computer vision and image processing tasks including image classification, detection, segmentation, and reconstruction.

- **Classification:** A task that takes an image as input and assigns a discrete class as output. For example, the input may be a chest X-ray, and the desired output would be a binary score where 0 corresponds to 'no detection of pneumonia,' and 1 corresponds to 'likely case of pneumonia.'
- **Detection:** Takes classification one step further by localizing the object or objects of interest. For example, a chest X-ray may be provided as input but instead of a single binary prediction, we are interested in the precise locations of lung nodules. In this case a detection model would be trained to produce coordinates for a 'bounding box' localizing each nodule in the image.
- **Segmentation:** The most fine-grain computer vision task, where every individual pixel or voxel is assigned to a class. In a segmentation task we may want to know the precise shape of a prostate in an MRI image, or the exact shape of a liver and the tumors within. Segmentation is especially prevalent in medical imaging because the granularity of this

information is valuable for a variety of clinical applications. Provided with a reliable segmentation, a downstream task may be to estimate the volume of an organ or tumor or run secondary analysis to check for structural abnormalities.

- **Image reconstruction:** In modalities such as CT and MR, computes the attenuation coefficients of different x-ray absorption paths (ray sum) that are obtained as a set of data (projection). The goal of such reconstruction algorithms is to produce an efficient yet accurate image reconstruction method while keeping the radiation dose to a minimum.

Given the modularity of NVIDIA Clara Deploy and the freedom of not being tethered to a specific library for operator development, there are few applications that could not be made into a scalable and efficient deployment pipeline. Other examples of possible workflows with NVIDIA Clara Deploy SDK:

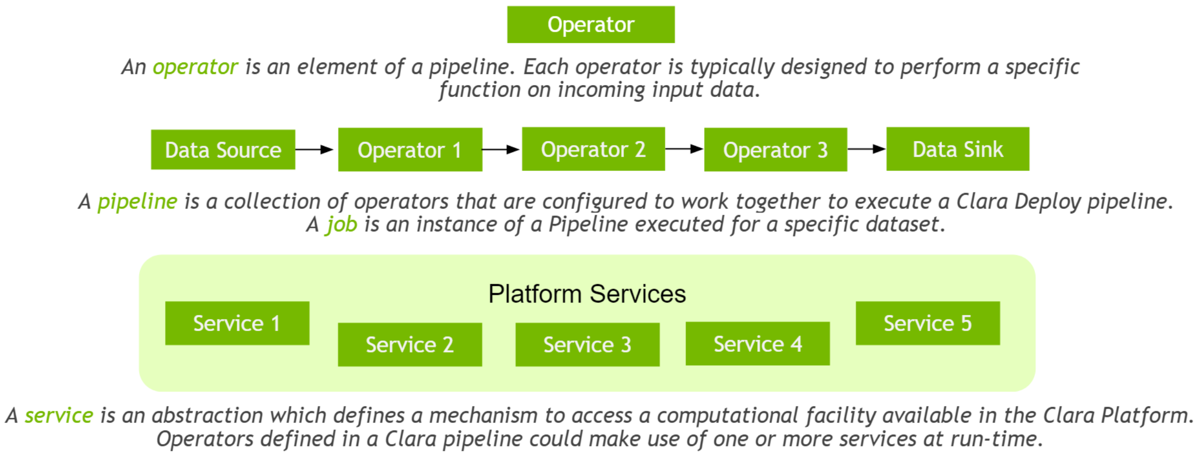
- Image Registration
- Image Super-resolution
- Image Synthesis using GANs
- DeNovo Sequencing

NVIDIA Clara Pipelines and Operators

NVIDIA Clara Deploy applications are built around the concepts of pipelines and operators. An operator is an individual component of an end-to-end NVIDIA Clara workflow. Operators typically execute one individual function such as reading or writing a DICOM image, performing inference with a trained model, or manipulating the data in some other way such as cropping, rescaling, or reformatting. Each operator is a containerized package including all required code and software dependencies so that it can be executed in several different environments and pipelines. The modularity of this approach to application deployment allows for scalable and portable solutions capable of running on multiple hardware platforms.

Beyond the reference operators included in NVIDIA Clara Deploy, custom operators can be designed and integrated into new pipelines and workflows. There is no specific library that an application needs to depend on in order to integrate with and deploy on the NVIDIA Clara Deploy framework. Custom or experimental code that has been written for training a new model architecture should be easily portable to an NVIDIA Clara Deploy operator for inference. See the Clara Deploy Operator Developer Guide for more information.

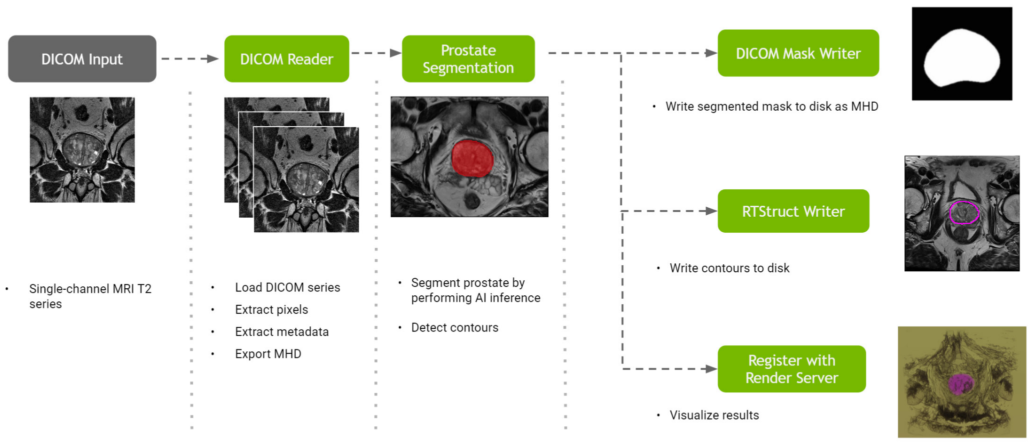
Figure 3. NVIDIA Clara Pipelines and Operators


Operator
An operator is an element of a pipeline. Each operator is typically designed to perform a specific function on incoming input data.
Data Source
Operator 1
Operator 2
Operator 3
Data Sink
A pipeline is a collection of operators that are configured to work together to execute a Clara Deploy pipeline.
A job is an instance of a Pipeline executed for a specific dataset.
Platform Services
Service 1
Service 2
Service 3
Service 4
Service 5
A service is an abstraction which defines a mechanism to access a computational facility available in the Clara Platform.
Operators defined in a Clara pipeline could make use of one or more services at run-time.

Reference Pipelines

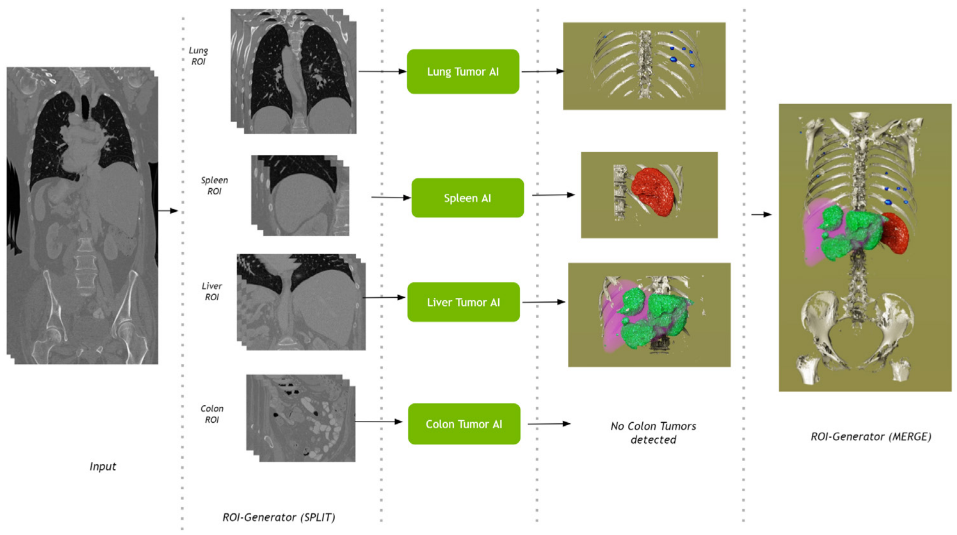
The following figure provides details of the prostate segmentation reference pipeline included with NVIDIA Clara Deploy. The first operator in this pipeline is a DICOM reader that loads the MRI series and converts the incoming data. The segmentation operator ingests this output and performs inference using a model optimized with NVIDIA® TensorRT™, a library that accelerates neural network inference by up to 40x compared with CPU-only platforms. The result is then routed to three operators for post-processing: a DICOM mask writer to create a standard segmented output, a second operator to save the result in RTStruct format, and a final operator to register the result with the NVIDIA Clara Render Server service for further visual analysis.

Figure 4. Prostate Segmentation Pipeline



A more advanced pipeline such as the multi-AI pipeline (Figure 5) shows an even more complex workflow. Using an abdominal CT as input, this pipeline calculates the region-of-interest (ROI) in the input image for each organ, runs inference with four models in parallel on those regions respectively, and then stitches the outputs back together to view the results. Workflows like this show enormous potential for accelerating a physician's time to insight provided only a single medical image.

Figure 5. Multi-AI Pipeline



Visualizing Results with NVIDIA Clara Render

Included in the NVIDIA Clara Deploy framework is the NVIDIA Clara Render Server. This is a viewer that uses GPU to visualize the results of a pipeline. The render server can be particularly useful for visualizing segmentation outputs. Included in this viewer are options for a cinematic view (Figure 6), or a left, right, or top slice-by-slice view, along with the ability to modify the transfer function to control the opacity/transparency of different objects. A snapshot of the user interface is displayed in Figure 7. For a full list of features and capabilities see Clara Dashboard Render Service

Figure 6. NVIDIA Clara Render Visualization

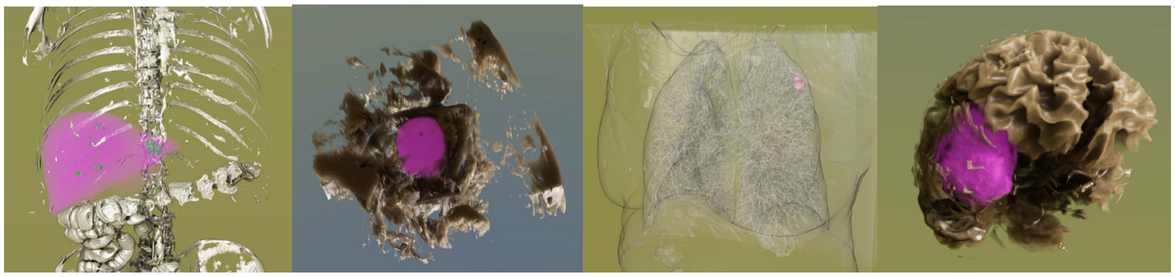
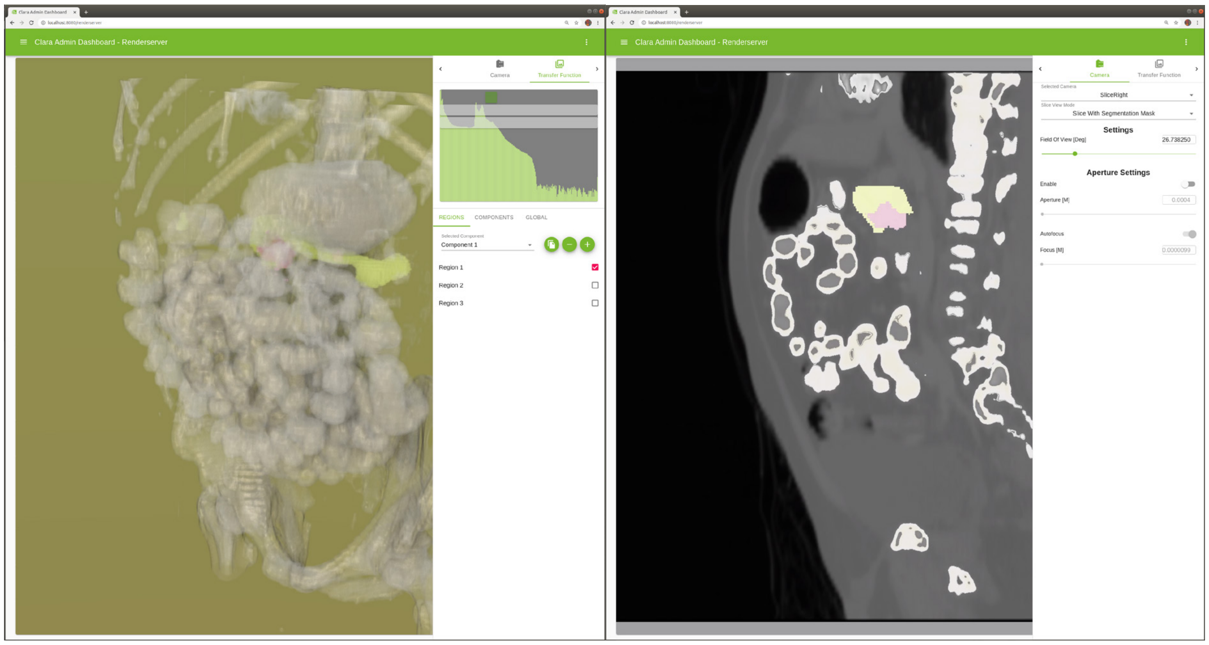


Figure 7. NVIDIA Clara Render User Interface



Selected Component: Component 1

Region 1 [✓]
Region 2 ☐
Region 3 ☐

Selected Camera: SliceRight
Slice View Mode: Slice With Segmentation Mask

Settings

Field Of View [Deg]: 26.738250

Aperture Settings

Enable
Aperture [M]: 0.0004
Autofocus
Focus [M]: 0.0000099

Enabling an AI-Ready Healthcare Institution

Creating an AI-ready healthcare institution is about more than just acquiring some hardware and installing a new package. Supporting this type of endeavor in a manner that can grow and provide value requires selection of a relevant clinical problem, carefully designed and measured goals, and mapping out a workflow to get from experimental design to results in a repeatable way. In this section is one example of a recommended workflow as well as details related to supporting the infrastructure and integration of these workflows at an institutional level.

Selecting a Clinical Problem

The first step towards enabling an AI-ready healthcare institution is to identify an appropriate clinical imaging problem. The right type of problem often has a combination of two criteria:

- An immediate clinical need.
- Can benefit from a deep learning-based model which would provide insights that traditional visualization of imaging datasets cannot provide.

In addition, if the deep learning model has not been trained yet, selecting a problem for which requisite privacy-compliant ground truth data is already available speeds up the deployment process.

Designing a Pipeline

Once a clinical problem has been identified, it's time to break down the problem into multiple and independent subproblems. Each subproblem addresses a compute task that can be handled by an operator. For example, an operator in a pipeline may perform pre-processing of imaging data or may make use of a model to perform inference. Each operator is typically designed to perform a specific function and analysis on incoming input image data. NVIDIA Clara Deploy provides a pipeline definition language using each stage of the workflow that can be defined with appropriate input and output data types and necessary compute resources.

Designing an Operator

Using the NVIDIA Clara Deploy framework, a pipeline is composed of one or more operators. Each operator often ingests one or more volumetric datasets and performs a computer vision task on it. The base operator that comes bundled with NVIDIA Clara Deploy can be used as a template to develop a new operator. The base operator template makes it easy to import an AI model that was trained with the NVIDIA Clara Train framework and make use of it on the NVIDIA Clara Deploy platform. After an operator is unit-tested in the developer's local environment, it can be containerized and incorporated in a pipeline.

Deploying a Pipeline

After all the operators have been incorporated in a pipeline using the NVIDIA Clara Deploy pipeline definition language, the pipeline is ready to be published. Once the pipeline gets published to the NVIDIA Clara Deploy platform, it can be used to trigger new jobs when new DICOM datasets arrive in the platform. NVIDIA Clara Deploy platform comes bundled with a DICOM adapter which offers a DICOM Storage service provider.

A DICOM Storage Service Provider is a server that can receive DICOM datasets from compliant imaging modalities and Picture Archive and Communication Systems (PACS). The DICOM Adapter can be configured so that whenever DICOM datasets are sent to a specific application entity title managed by the NVIDIA Clara Deploy platform, a pre-configured pipeline is used to trigger a job.

Monitoring and Maintaining a Pipeline

An important part of evaluating the success of a deployed pipeline is having the ability to understand how the pipeline (and its constituent operators) performs - how often it is used, the resources utilized by the pipeline over a period of time, clinical validity of the results generated, etc. Based on such data, often one or more operators in the pipeline may need to be modified and re-deployed. This cycle of deploying, understanding performance and usage, modifying the pipeline and redeploying continues until a pipeline is retired from production. Gathering metrics regarding deployed pipelines will help healthcare organizations to move through this iterative cycle faster, as ultimately successful deployment of clinical workflows depends on the adoption of the pipelines and their impact on patient care.

Example Hardware Configurations

The following table provides several recommended base configurations for designing an infrastructure capable of supporting and scaling NVIDIA Clara Deploy pipelines.

Table 1. Hardware Configuration Examples

	Recommended Base Configurations[1]		
Server Component	**V100 NVLink Data Center**	**T4 Data Center**	**T4 Edge Deployment**
Validated GPU configuration	8 GPUs with 2 CPUs using NVIDIA® NVLink®	4 GPUs with 1 CPU	2 GPUs with 1 CPU
CPU cores	64 cores	32 cores	12 cores
CPU speed	2.1 GHz base clock	2.1 GHz base clock	2.1 GHz base clock
System memory	640 GB	160 GB	80 GB
Networking	2 - 200 Gbit NICs	1 - 100 Gbit NICs	1 - 100 Gbit NICs
Storage	One SSD drive per system (either socket)	One SSD drive per system (either socket)	One SSD drive per system (either socket)
Edge hardware			TPM module and Redfish 1.0 compatible IPMI remote management

Note:

[1]Hardware configurations assume a single node deployment of NVIDIA Clara Deploy SDK.

For vendor-specific certified configurations refer to the list of NGC-ready systems. NGC-Ready servers have passed an extensive suite of tests that validate their ability to deliver high performance running NVIDIA® NGC™ containers. NGC-Ready system validation includes tests of:

- Training from scratch and Transfer Learning using NVIDIA Clara Train SDK™
- Single and multi-GPU Deep Learning training using TensorFlow, PyTorch and NVIDIA DeepStream Transfer Learning Toolkit.
- High volume, low latency inference using NVIDIA TensorRT, Triton Inference Server, and DeepStream SDK.
- Data Science using NVIDIA Rapids™ and XGBoost.
- Application development using the NVIDIA® CUDA® Toolkit.

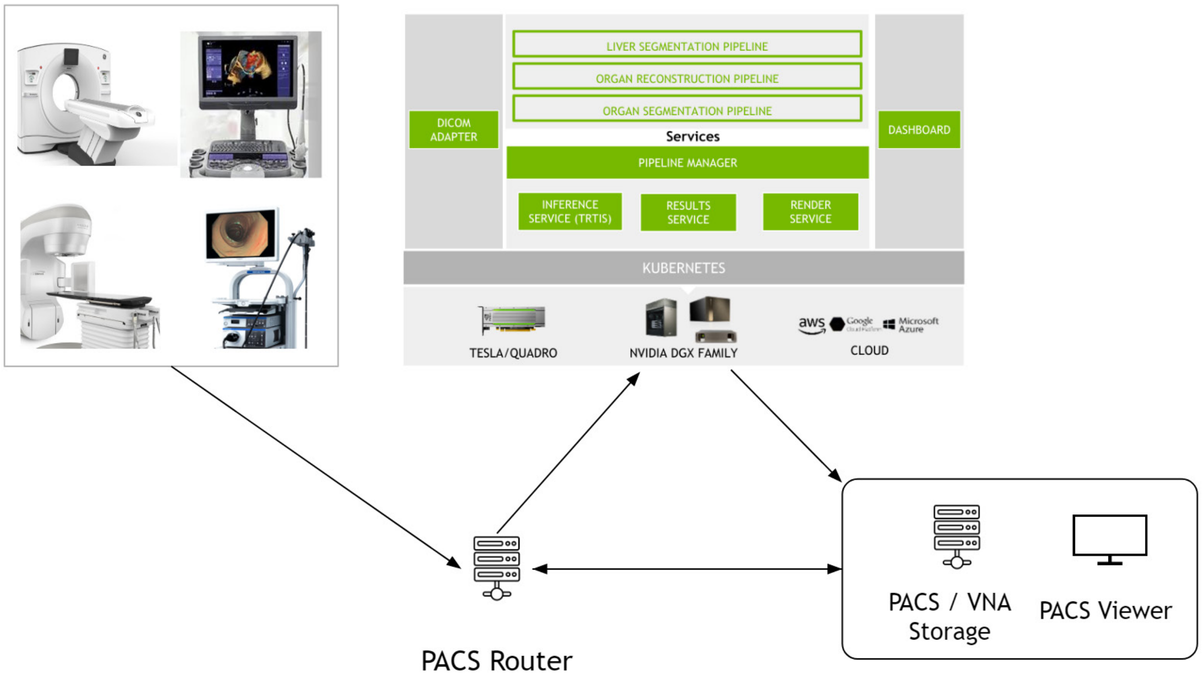
PACS Integration

NVIDIA Clara Deploy provides the pipelines and operators to receive a medical imaging study, transform it and run inference upon it, and then transform and transmit the result. However, this must integrate into the medical imaging ecosystem to be effective. Healthcare has a robust and mature interoperability framework; medical imaging uses a standard called DICOM, and healthcare metadata uses a standard called HL7.

DICOM is a transport and storage specification that every medical imaging device speaks - whether it is the modality or instrument producing and transmitting the medical images, the router that transports medical images, the PACS that stores medical images and drives workflow, or the viewer that displays medical images. When NVIDIA Clara Deploy is used inside of the medical imaging ecosystem, it can be connected in several ways:

- As a post-processing step after the images are generated, where the modality or DICOM router automatically sends it to NVIDIA Clara DICOM adapter.
- As a routing step after the images are stored in PACS, where automatic rules determine whether to send studies to endpoints.

Figure 8. PACS Router Configuration



To send studies for processing to NVIDIA Clara, the integration is similar regardless of which device is transmitting (a modality, a PACS, or a DICOM router). The sender performs a C-STORE to the NVIDIA Clara DICOM adapter. A PACS administrator generally configures this step by logging into the device and adding a send destination to NVIDIA Clara (by specifying the specific IP address, port, and AE title). Once the NVIDIA Clara DICOM adapter receives the DICOM study, processing begins.

To send results back to the medical imaging ecosystem from NVIDIA Clara, the first consideration is to ensure that the downstream system can accept the type of images being generated. For example, some systems might not accept DICOM RTSTRUCT, but may accept secondary capture series. You can confirm the types of objects a device accepts by searching the device manufacturer website for a DICOM Conformance Statement. This lists the objects it can receive. Next, a PACS administrator must configure their PACS to accept objects from NVIDIA Clara Deploy - again, by specifying the AE title on which NVIDIA Clara is sending from. You should know the IP address, port, and AE title of the device you are sending results to.

Some considerations should be taken prior to sending back results. For example, hospitals might not want objects to appear in their production PACS for all users to see (like referring physicians). You might be required to send to a different AE title of a temporary cache for viewing. You should also consider how to handle errors in the workflow, when NVIDIA Clara receives the wrong type of images, which is out of its control. For example, if NVIDIA Clara has a chest x-ray pipeline and a PACS is forwarding chest CT exams.

Software

Setting Up NVIDIA Clara Deploy

NVIDIA Clara Deploy can be installed via Nvidia GPU Cloud (NGC). NGC is the easiest and best way to access and download NVIDIA software, libraries, and SDKs. Many of these resources are distributed in the form of Docker containers. Since NVIDIA Clara Deploy must interface with system-level components such as Kubernetes, the platform is distributed in two (non-containerized) parts: a "bootstrap" package to configure the system and ensure compatible software is installed, and a command-line interface (CLI) package that allows users to interact with the platform.

The bootstrap package is meant to run checks and/or install software needed as dependencies by NVIDIA Clara Deploy. Among other things this includes a compatible version of Kubernetes, Docker, and Helm. Operators in a pipeline are run as individual Docker containers, with their execution orchestrated by Kubernetes. Helm charts provide the framework and instructions for the NVIDIA Clara Deploy platform and services to run and manage workloads.

For more information on installation and getting started, refer to the Clara Deploy Installation Guide and the NGC Bootstrap Model Script.

Deploying a Reference Pipeline

NVIDIA Clara Deploy provides 17 out-of-the-box applications in the form of reference pipelines. These pipelines cover a wide array of use cases such as prostate segmentation, CT reconstruction, liver tumor segmentation, and chest X-ray classification, among others. Each reference pipeline includes a model script defining the execution of pipeline operators, a pre-trained neural network to perform inference, and sample test data. The catalogue of available reference pipelines continues to grow as the NVIDIA Applied Research team tackles problems on the bleeding edge of medical imaging with partners and researchers across the world. The following tables show the included reference pipelines.

Table 2. AI Reference Pipelines

	Segmentation	Classification	Object Detection
CT	Liver and tumor Colon tumor Lung tumor Abdomen Pancreas tumor Spleen Multi-AI		
MR	Hippocampus Brain Prostate		
CR		Chest X-ray	
Pathology		Breast cancer Malaria	
Endoscopy			Simulated object detection on torso model

Table 3. Accelerating Computing Reference Pipelines

	Recon	Other
CT	CT reconstruction	3D cropping using shared memory
Genomics		DeNovo sequencing

Reference Pipeline Performance

We provide approximate performance numbers for each of the reference pipelines included with NVIDIA Clara Deploy. These metrics were collected on a SuperMicro server using a single NVIDIA T4 GPU. The T4 GPU is based on the NVIDIA Turing™ architecture and has 16 GB of GPU memory, and is optimized for inference workloads. Each end-to-end inference time reported in Table 4 is the average of multiple pipeline executions using a single data point as input. Again, note that this is a full pipeline execution, not just running a single model inference.

While these numbers are included for reference, it should be noted that many factors influence the performance of inferencing workloads. The size of the input data, number and complexity of operators performing pre- or post-processing operations, whether the data is batched, and the underlying hardware architecture being used are just a few of these factors.

Table 4. Reference Pipeline Performance

Pipeline	Model	Input Size	End-to-End Inference Time (Sec)
Brain Tumor Segmentation	AHNet	240x240x155 19MB	25s
Hippocampus Segmentation	AHNet	47x37x42 1MB	20s
Spleen Segmentation	AHNet	440x440x515 201MB	88s
Lung Tumor Segmentation	AHNet	512x512x588 78MB	50s
Colon Tumor Segmentation	AHNet	512x512x189 100MB	60s
Malaria Classification	VGGNet	121x130x31 1MB	8s
Pancreas Tumor Segmentation	AHNet	487x487x455 219MB	104s
Chest X-Ray Classification	DenseNet121	256x256x1 <1MB	9s
Liver Segmentation	AHNet	512x512x588 311MB	78s
CT Reconstruction (Iterative FDK)	-	1024	85s
VNet Segmentation Pipeline	VNet	512x512x147 78MB	37s
Histopathology Pipeline	VGGNet	50x50x17 1MB	5s
Prostate Segmentation Pipeline	AHNet	320x320x20 4MB	24s
Multi-organ Pipeline	AHNet	512x512x947 994MB	108s

Supporting an AI Deployment

A crucial factor when considering how to roll out AI at an institution is to make sure there are personnel with the appropriate skill sets to support the endeavor in each phase. In this section we break AI deployment with NVIDIA Clara into different phases and highlight the role of each participant to ensure success.

PHASE 1: Platform Configuration and Installation

Personnel: Principal Investigator, Lead Data Scientist/Engineer, IT/datacenter expert

Success Criteria: Reference pipeline deployed on system

During this initial phase, hardware and software tasks that would be involved in the setup of any new system are performed. The first task is to decide what infrastructure the platform will be deployed on. Ideally this should be a dedicated, NGC-Ready node installed into a datacenter environment as described in "Designing a Pipeline" section. If the platform will be used more for research workflow (24/7 access from the clinic may not be a requirement), the platform could optionally be deployed on a desktop system with NVIDIA® Quadro® or Tesla® GPUs recommended. In situations where ongoing hands-on system administration is not desired or not possible, such as edge deployments in hospitals or clinics, NVIDIA EGX edge platform can be used to install the underlying GPU-enabled Kubernetes infrastructure required by NVIDIA Clara Deploy. Once software installation on the system is complete, success criteria for this phase is to deploy a reference pipeline end-to-end and verify results.

PHASE 2: Integration with Institutional Data

Personnel: Principal Investigator, Lead Data Scientist/Engineer

Success Criteria: Reference pipeline/custom pipeline run end-to-end on custom dataset

Once a reference pipeline has been successfully deployed on test data, the next goal is to either integrate institutional data with an applicable reference pipeline or stand up and verify a custom pipeline depending on workflow requirements. In the case of the former, this should not be a long process and will mainly consist of ensuring that your input dataset is properly configured to be ingested by the reference pipeline. In the case of the latter, the process may be more involved and will require the lead data scientist/engineer to assess what custom operators must be built and integrated into the pipeline before it can be tested.

PHASE 3: Connection to PACS System for End-to-End Workflow

Personnel: Lead Data Scientist/Engineer, IT/datacenter expert, PACS Administrator

Success Criteria: Reference pipeline deployed on system

Once the desired workload has been tested end-to-end on the system, the final step to full deployment is to connect to the PACS. Details for this process are included in the "PACS Integration" of this white paper. A PACS administrator needs to be involved at this stage to help input the specific IP address, port, and AE title of the pipeline and deployment hardware into the system. Success is achieved when an image is pushed from the PACS, processed by NVIDIA Clara, stored back in the PACS and then viewed and verified to be accurate.

Case Studies

NVIDIA works with early adopters and partners to understand the needs of the clinical and research community, and to design software that best meets those needs. NVIDIA Clara Deploy has greatly benefited from these collaborations and feedback. We provide the outcomes of several such partnerships in the following sections.

SUNY Upstate

Researchers at SUNY Upstate in Syracuse, NY have deployed a T4 server in their data center that runs a pipeline to ingest MRI images, perform whole prostate segmentation, and receive the output in RTStruct format. NVIDIA worked with the SUNY team to build a custom RTStruct operator that is now available for reuse on NGC, as well as the entire end-to-end pipeline.

Table 5. Prostate Segmentation Pipeline

Application	Modality	Hardware	GPU
Prostate RTSTruct Segmentation	MRI	Dell PowerEdge server	2x T4

National Institutes of Health Clinical Center

Researchers at the National Institutes of Health Clinical Center have successfully implemented Liver Tumor Segmentation and Whole Prostate Segmentation pipelines using NVIDIA Clara Deploy and integrated the results into their CareStream PACS system. Researchers can now push a DICOM series directly from the research PACS system to an NVIDIA DGX-2™ for processing and receive the segmented output in the PACS for further analysis.

Table 6. Liver Tumor and Prostate Segmentation Pipelines

Application	Modality	Hardware	GPU
Liver Tumor Segmentation	CT	NVIDIA DGX-2	16x V100
Whole Prostate Segmentation	MRI	NVIDIA DGX-2	16x V100

CCDS

The Center for Clinical Data Science (CCDS), a joint center of the Massachusetts General Hospital (MGH) and Brigham Women's Hospital (BWH) and part of the broader Partners HealthCare System in Boston Massachusetts has successfully implemented an end-to-end kidney stone pipeline using NVIDIA Clara Deploy. The inference workflow is initiated through a DICOM Router (Compass, produced by Laurel Bridge), which sends the relevant DICOM series to the NVIDIA Clara Deploy DICOM adapter with the AE title matching the kidney stone segmentation pipeline. This kickstarts the requested inference pipeline in NVIDIA Clara Deploy, the result of which is sent to a database of choice (the relevant entity within the Partners HealthCare network) and made available for Visage visualization.

Table 7. Kidney Stone Segmentation Pipeline

Application	**Modality**	**Hardware**	**GPU**
Kidney Stone Segmentation	CT	NVIDIA DGX-1™	8x P100

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com