Hewlett Packard Enterprise

Document type

CONFIDENTIAL | AUTHORIZED HPE PARTNER USE ONLY

COMPREHENSIVE TRAINING ON PRIVACY-PRESERVING DISTRIBUTED DEEP LEARNING AND ACCELERATED AI

CONTENTS

HPE Official - Not for circulation

HPE Official – Not for circulation

HPE Official - Not for circulation

MODULE 1

PRIVACY-PRESERVING DISTRIBUTED LEARNING - SWARM LEARNING

HPE Official – Not for circulation

1.1 Introduction

With data as its cornerstone, the global economy is becoming increasingly digital. But before we can fully harness the potential of data, we must create the algorithms, models, and systems necessary to extract hidden insights from the data and put them into practice. Numerous applications, including image recognition, object detection, and machine translation, have seen considerable success with machine learning (ML) and Deep Learning (DL) Architectures. Inferencing is typically performed using models trained on aggregated data in public or private clouds. These models, which are being used in the cloud or at the edge, are bringing about significant adjustments in several industries, including healthcare, agriculture, retail, and transportation.

The training data is gathered in one place and used to create, train, and test machine models in conventional techniques. This centralized strategy, though, is faced with growing socioeconomic and technical difficulties. The enormous amount of data needed to train ML models can be difficult to transfer and aggregate due to issues with data sovereignty, security, and privacy. A central infrastructure that hosts and processes the aggregated data can be expensive to build.

We have seen how to build Neural Networks or Artificial Neural Networks that require very less human intervention. Also, Deep learning is used when there are millions of datapoints available for analysis compared to the traditional machine learning techniques, which can process thousands of data. Also, to achieve human-level action, it is required to train a neural network with a huge amount of data. Data that is not only huge but may not be adequate for complete analysis by an organization.

Many machine learning applications, most of them utilizing the centralized learning approach, and one of the major challenges in this approach is the increasingly distributed nature of the data. Data is produced at speed and in high volume, and at a distributed location.

HPE Official - Not for circulation

Nowadays, there is a trend of edge computing that is moving computing and intelligence closer to data. Data privacy and security are important for an organization that generates its own data. Swarm Learning tries to overcome the limitation.

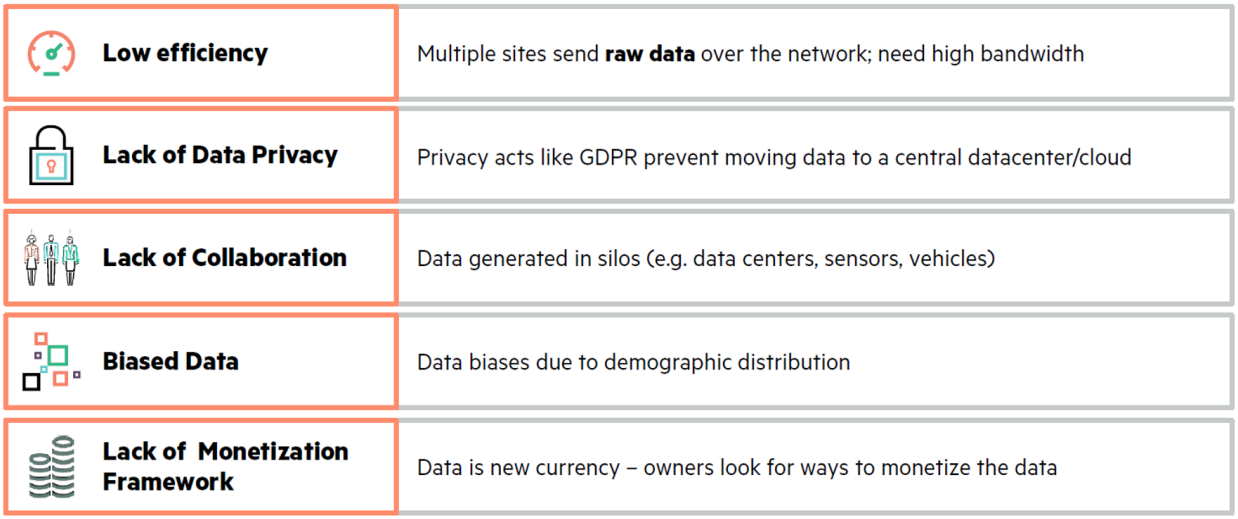


Figure 1.1 Challenges in ML

1.2 Model Performance Optimization

An optimization problem is a problem of finding the Best Solution / Minimal Cost Solution from all the feasible solutions. Let us have a quick look at various optimization algorithms that can be utilized for data-driven learning

1.2.1 Classical Numerical Optimization

Y = credit = a0 + (a1 * age) + (a2 * income) + (a3 * education)

For this set of problems, we go for Gradient Descent

Gradient Descent: It is an iterative first-order optimization algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning (DL) to minimize a cost/loss function (e.g., in linear regression).

A lot can go wrong with GD:

HPE Official - Not to

- Local minima

1.2.2 Amoeba Learning

Pre-req [Simplex Function]. It is a popular optimization algorithm for linear programming.

In Amoeba, instead of one solution to a problem, we've many; here it is 3.

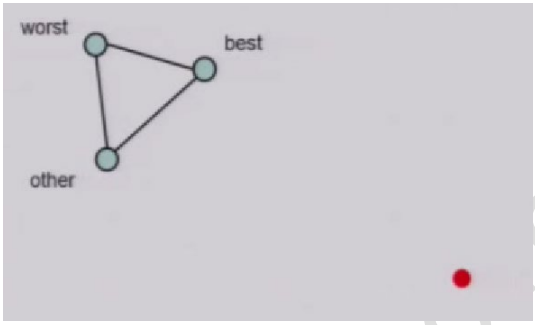


Figure 1.2 Simplex Function

What are these points

- Each point has [position, associated error]

Steps

- Initialize 3 points to a random position
- Start a Loop
    a. Compute the centroid based on "best" and "other" points as explained
    b. Compute Expanded

        If expanded better than worst, replace the worst

    c. Compute Reflected

        If reflected better than worst, replace the worst

    d. Compute Contracted

        If reflected better than worst, replace the worst

HPE Official - Not for circulation

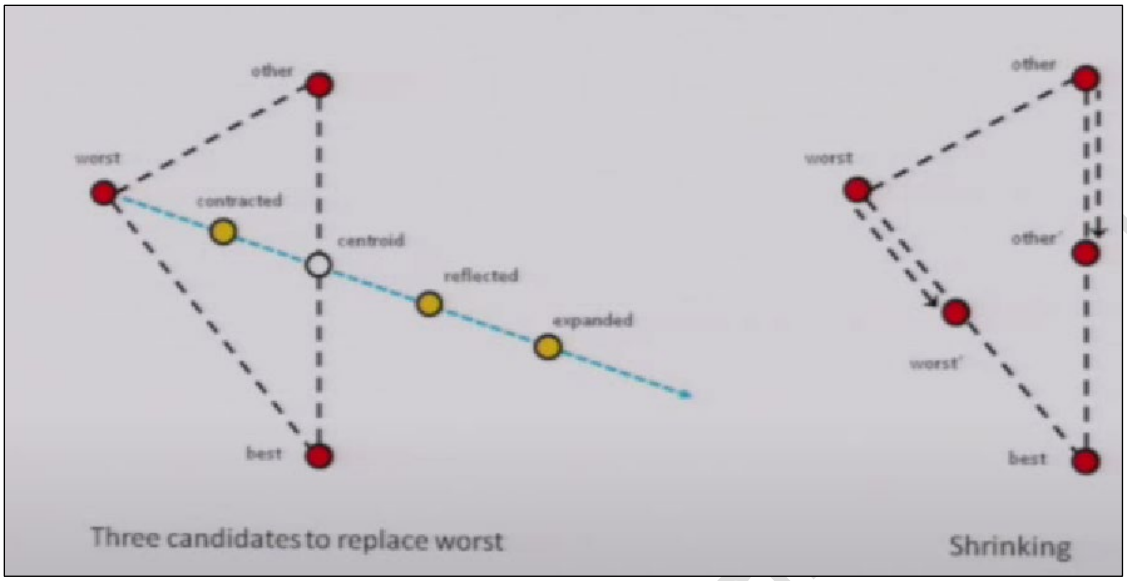


Figure 1.3 Three Candidates to replace worst

e. Shrink “other” and “worst” toward the best
f. End Loop

1.2.3 Particle Swarm Learning

Particle Swarm Optimization was proposed by Kennedy and Eberhart in 1995. As mentioned in the original paper, sociobiologists believe a school of fish or a flock of birds that moves in a group “**need to profit from the experience of all other members**.”

Particle Swarm Optimization is inspired by the social foraging behavior of some animals, such as the flocking behavior of birds and the schooling behavior of fishes.

The goal of the algorithm is to have all the particles locate the optima in a multi-dimensional space, initially assigned with random position and random velocity, gradually advancing towards the local optima using exploration and exploitation of good, known positions in space.

In this optimization problem, we find the maximum and minimum of a function. Let us suppose f(X). where X is the coordinate in the XY plane as (x,y).

Overview of the Logic

Suppose we have ‘**P’** particles; each particle will have the following attributes

HPE Official - Not for

- Position
- Velocity
- Error
- Best Position
- Best Error

The two global parameters would be

- Overall best position
- Overall best error

PSO Algorithm

At the start, the Positions are randomly initialized

Next, there will be a loop for a certain number of epochs

In Each Epoch

a. The next direction depends on the current direction, the best position by any particle
b. Use the new direction to compute the new position
c. Check if the new position is as best or swarm best
d. END for loop

Reports the best position by any particle

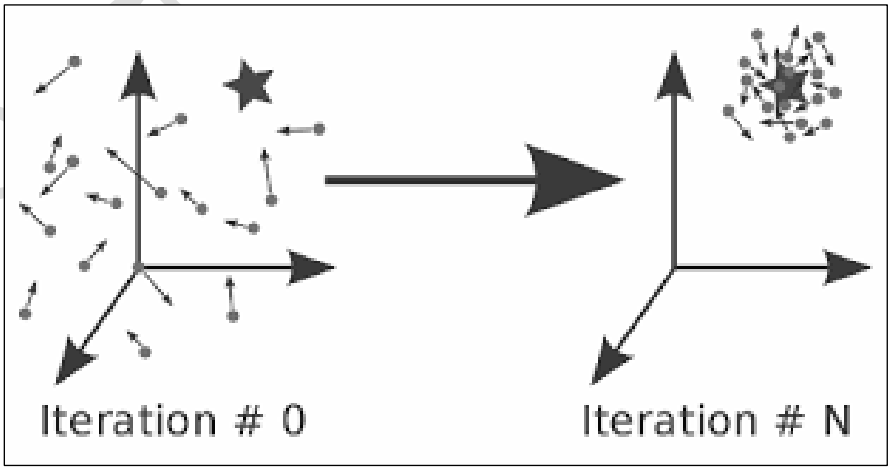


Figure. 1.4 PSO Algorithm

HPE Confidential - Not for circulation

1.3 Different approaches in Machine Learning

There are four different approaches to Machine Learning.

1. Disconnected Learning
2. Centralized ML
3. Federated ML
4. Swarm Learning

1.3.1 Disconnected Learning

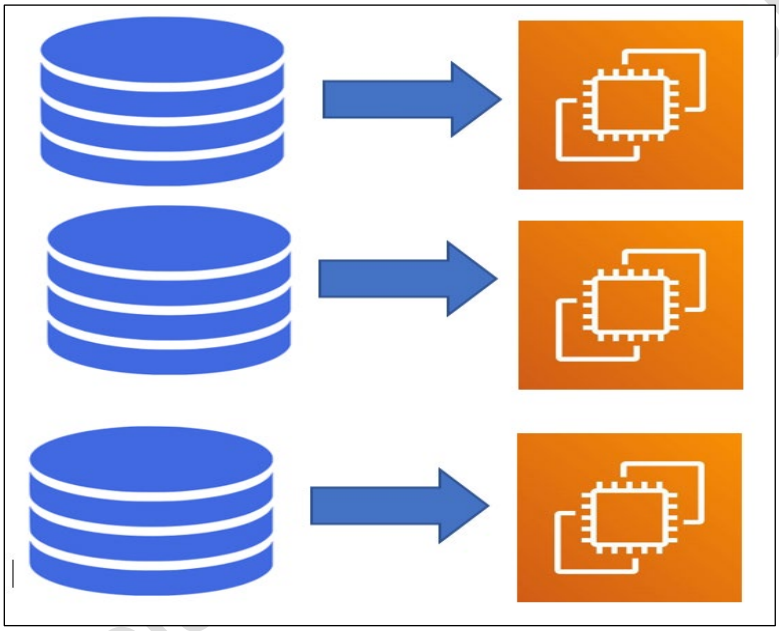


Figure 1.5 Disconnected Learning

From the figure, there is a set of individual machines that are independent. Each machine is having their own set of data and is also trained independently. The same data can be trained using different machine-learning techniques at the same time.

But there is a different limitation in this approach as follows:

a. Data Availability.

   Not all data can be contained in a single place as it need large storage space for data accumulation, which may not be feasible.

b. Cost in Data Migration to each local space.

HPE Official Publication

Data Transaction costs to each place may be high and time-consuming.

c. Not Adequate Data

Data required for training a model may not be complete and adequate.

d. It can only be used for simple workloads.

1.3.2 Centralized ML

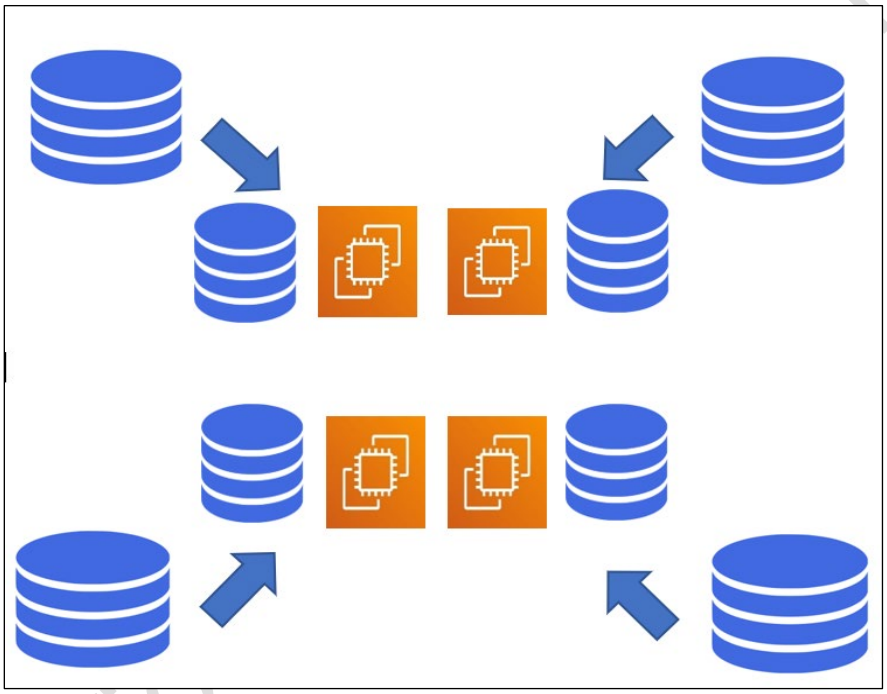


Figure 1.6 Centralized Learning

From the figure, there is a central repository where the data is available.

A central server is used to orchestrate the different steps of the algorithms and coordinate all the participating nodes during the learning process. Hence the Data and Parameters are stored and available centrally. The training data is aggregated to a centralized location, where machine models are developed, trained, and tested.

HPE Offic lation

Data sovereignty, security, and privacy can all create barriers to aggregating the huge amount of data required to train ML models. In addition to these costs of a central infrastructure.

1.3.3 Federated Learning

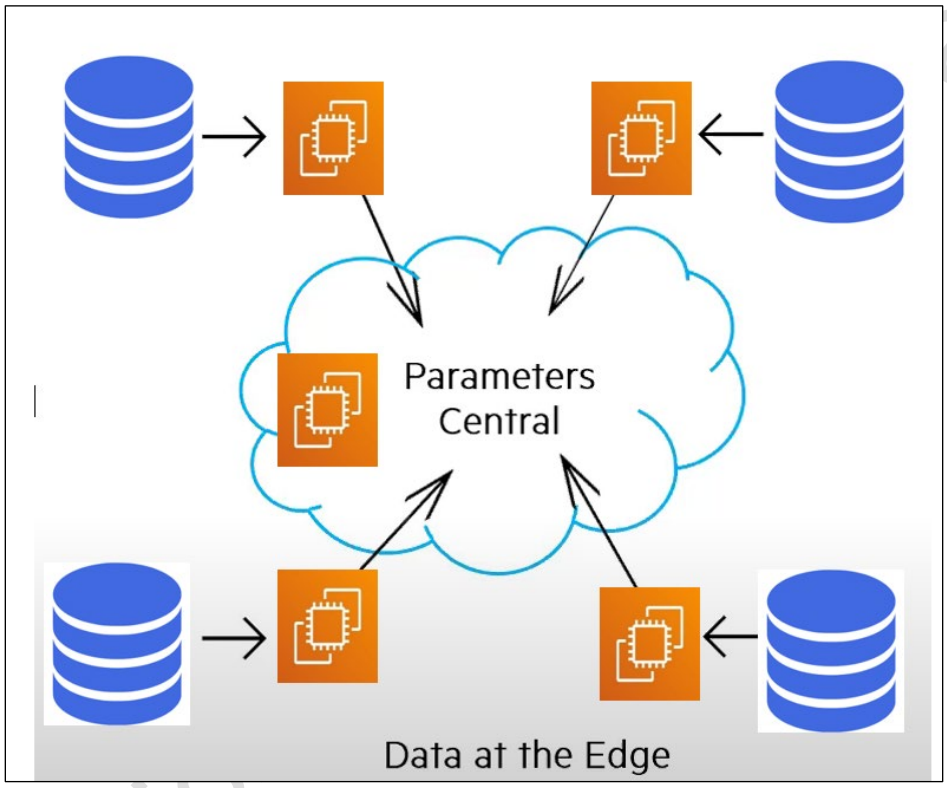


Figure 1.7 Federated Learning

It is a Distributed Machine Learning Framework. Federated learning (also known as collaborative learning) is a machine learning technique that trains an algorithm across multiple decentralized edge devices or servers holding local data samples without exchanging them. In this kind of Learning, Models are trained at the edges, and parameters are learned. Next, these Parameters are merger by a central coordinator.

Hence, now our centralized machine learning application will have a local copy on all devices, where users can use them according to their needs. The model will now gradually learn and train itself on the information available locally from time to time. Then the

HPE Offic ation

devices share the training results, not the data, from the local copy of the machine learning app, into the central server. The developer now updates the model to a newer version

Benefits over other

- Ensures Data Privacy
- A lot of cost savings in transferring data to a central storage and Storage cost etc.

Drawback

- Security, as there is still a third party who controls the central server and has access to the updated parameter.
- Lack of proper Monetization on parameter sharing.

1.4 Problem with Centralized and Traditional Techniques

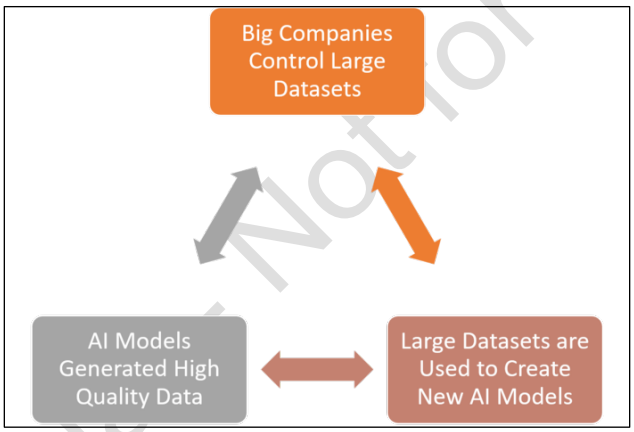


Figure 1.8 Problem with Centralized and Traditional Techniques

AI is not only an intelligence problem but a data problem. Today, large datasets relevant to AI problems are controlled by a small number of large organizations.

HPE Official - Not for circulation

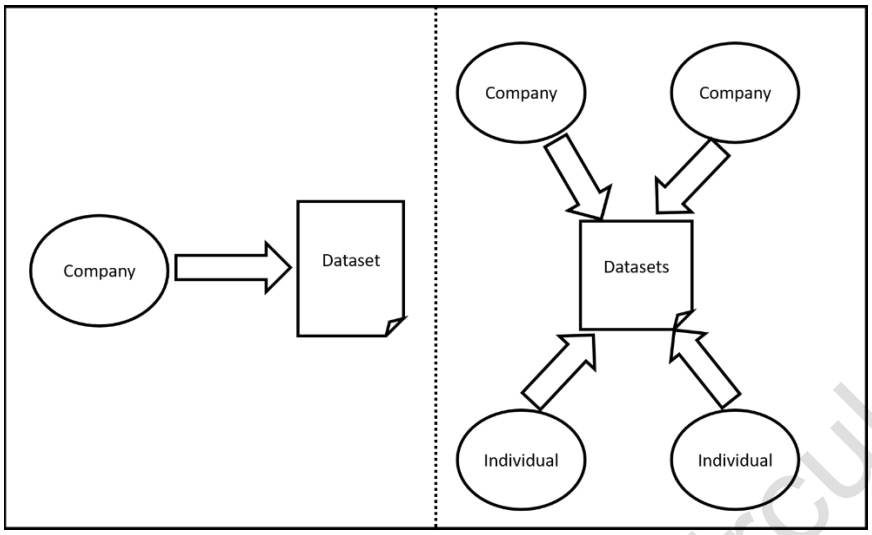


Figure 1.9 Centralized Data

Data Security: Imagine a healthcare AI scenario in which any participant in an experiment could contribute their own data with the right security and privacy guarantee. As data is shared at a center location hence, the third party has access to all the information.

Model Centralization Problem: Imagine that you want to evaluate a different model for the Scenario across the globe. And the data is available in a central repository**.** Data transfer and storage cost is high.

Training Centralization Problem: Model Overfitting will happen frequently, or model bias, as it is trained on the vast amount of data with the aim of minimizing the error and making a low-bias model.

Model Optimization problem: We rely on centralized processes for the optimization and regularization of AI models that very often use the same data that created the models. The model is optimized on the available set of data, which may result in a lack of model generalization.

Lack of Standard Monetization: To the collaboration of different organizations for data sharing, there are not any standard measures for monetizing content sharing.

HPE Official – Not for circulation

1.5 Introduction to Swarm Learning

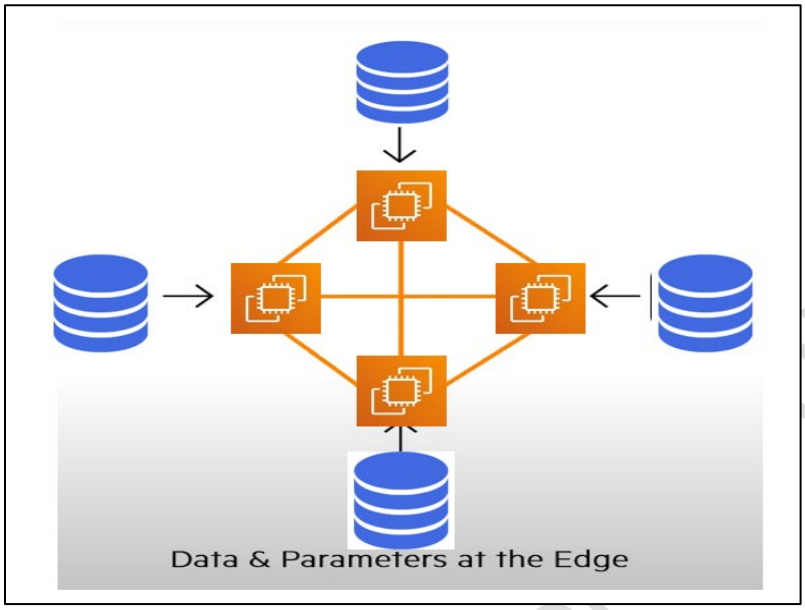


Figure 1.10 Swarm Learning

Swarm Learning is a framework designed for creating a set of nodes in which each node possesses some training data locally to train a common Machine Learning model without sharing data. Instead, the parameter learned by each node is shared and merged to obtain a global model.

One important thing is that there is not a static central merger, but it may be any node in the swarm network that got elected for parameter merging.

A Swarm is a configuration of tens of thousands of individuals that have chosen their own will to converge on a common goal. Swarm Intelligence is the Complex Collective, Self-Organized, Coordinated, Flexible, and Robust Behaviors of a group following simple rules.

HPE Official - Not for

1.6 Swarm Intelligence

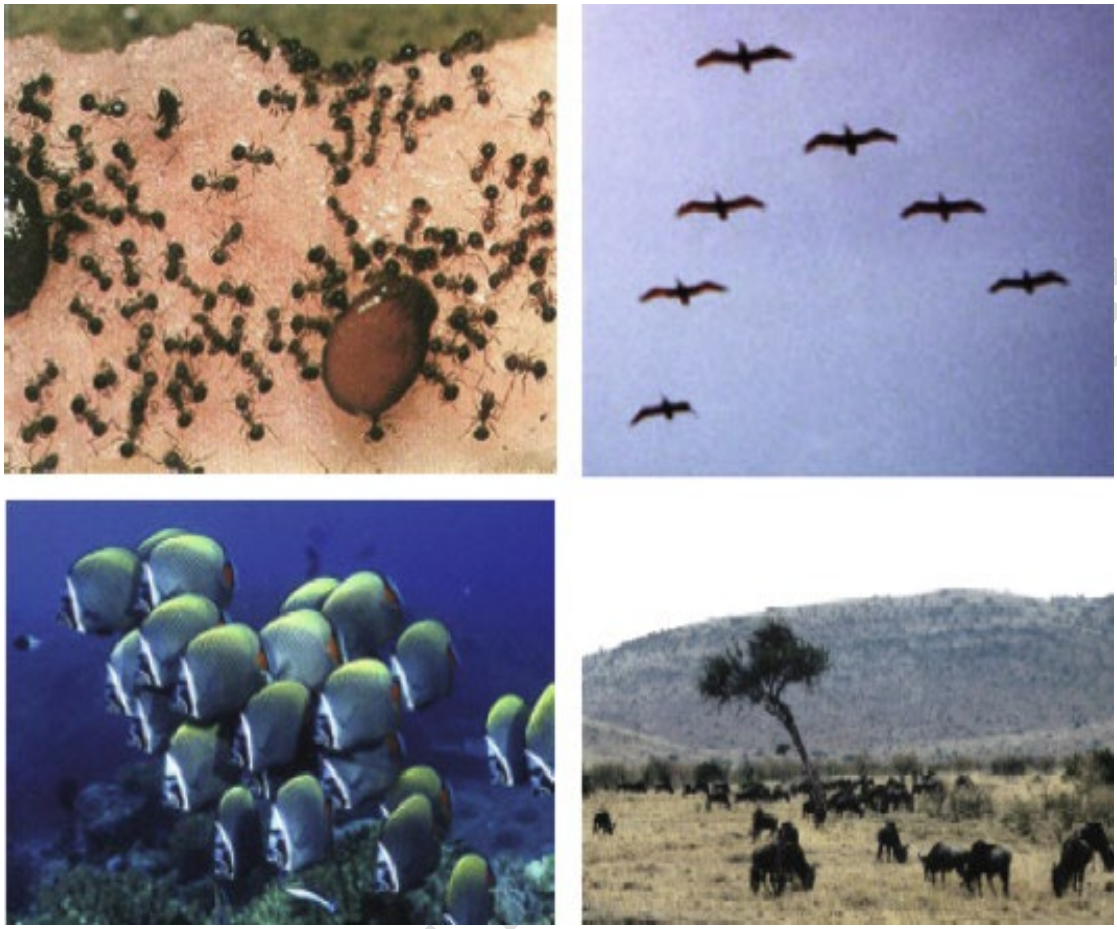


Figure 1.11 Swarm Intelligence

SI systems typically consist of a population of simple agents or bodies interacting locally with one another and with their environment. The inspiration often comes from nature, especially biological systems. The agents follow quite simple rules, and although there is no centralized control structure dictating how individual agents should behave, local and random interactions between such agents lead to the emergence of "intelligent" global behavior unknown to the individual agents

HPE Official

1.7 Artificial Swarm Intelligence

1.7.1 Swarm Learning Architecture

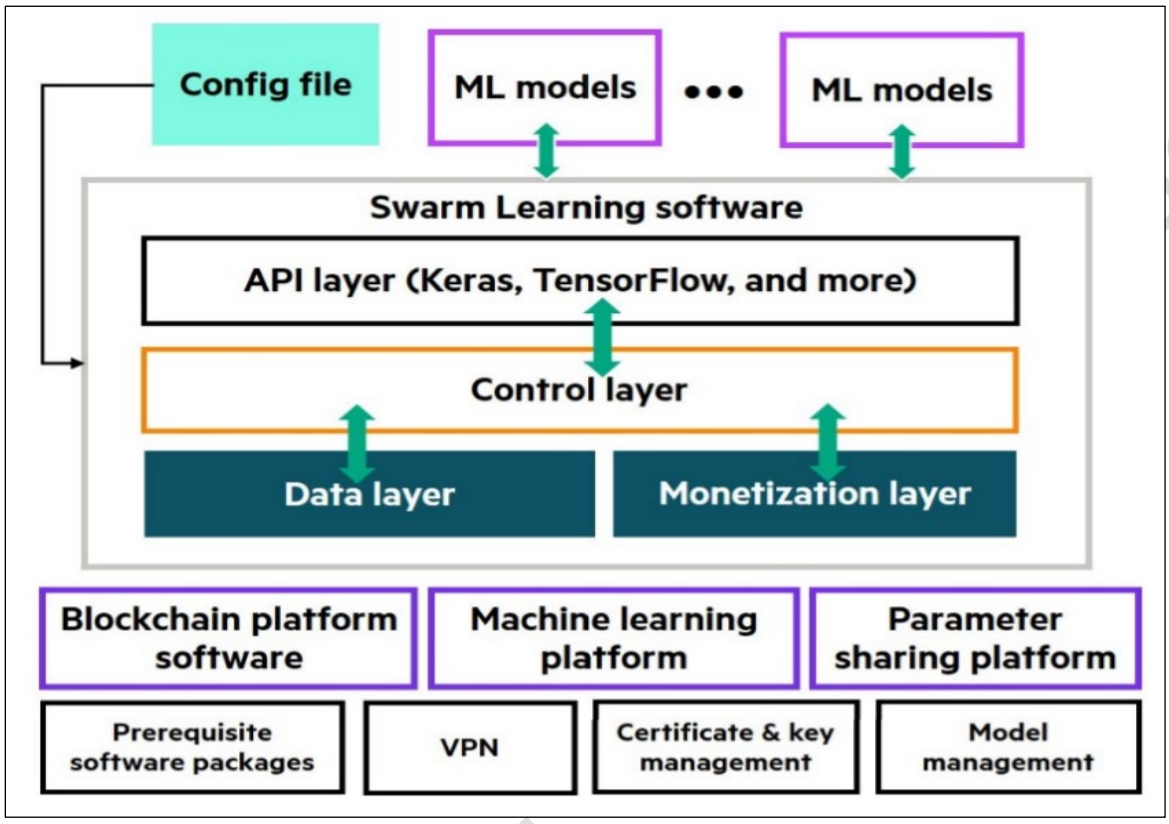


Figure 1.12 Swarm Learning Architecture

The architecture of Swarm Learning can be divided into the following four layers

- API
- Control
- Data
- Monetization

API layer

Swarm Learning work as an API library and supports TensorFlow, Keras, and PyTorch. These are famous and commonly used training APIs in the data-science domain. Each node seamlessly exchanges parameters at the end of each model training epoch and subsequently continues the training after resetting the local models to the globally merged parameters

HPE Official -

Control Layers

This layer makes the swarm network a globally consistent state and involves blockchain technology. The state comprised information such as the current epoch, current members of the Swarm with their IP addresses and ports, and URIs for parameters files. The set of operations here also involves logic to elect the leader node to merge parameters.

Data layer

The data layer takes charge of the reliable and secure sharing of model parameters across the Swarm Network. It supports HTTPS and TLS.

Monetization Layer

This layer meters data usage and contribution during the model training process to calculate the monetary rewards for each of the participant's nodes, which are dispensed at the end of the training.

How Swarm benefits your Business?

The root of SWARM ML's purpose is solving the dilemma of an explosion of data and the technical challenges of extracting values from data.

The following are the usual benefits:

1. **Efficiency**

Cost of data transfer to a central location for processing faces costs of data transfer and investment in storage and computing capability.

2. **Privacy and Security**

There are regulatory compliance, and legislative efforts on data privacy and parameter sharing as the company must pay a huge amount on settlement of privacy violation

HPE Official - Not for circulation

3. **Fault Tolerant**

Swarm ML is a decentralized method for both data storage and learning that avoids single-point failure and also does not have any bottleneck point. This also deals with problem-biased data from various sources.

1.7.2 Monetization model

The sensitive information never leaves the system of the user or organization's control. Services depend on the use of these data in containerized and authorized by the data owner. There is an incentive designed to facilitate data sharing and collaboration for both the user and the organization.

1.8 Components of Swarm ML

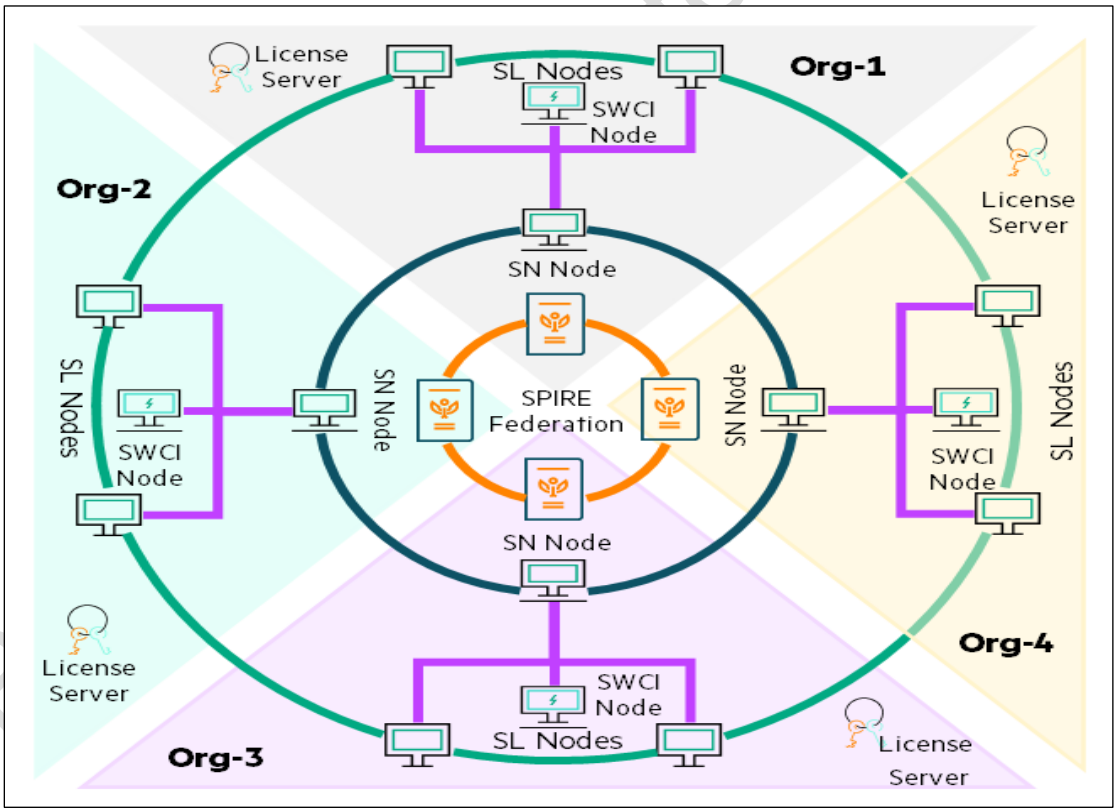


Figure 1.13 Components of Swarm ML

HP for circulation

- Swarm Learning Node
- Swarm Network Node
- Swarm CLI
- Spiffe Spire Node

Swarm Learning Node

These nodes run a user-defined Machine Learning algorithm. Here your deep learning model is created and trained on the local data available. This node is Responsible for updating the model over an epoch. The ML algorithm can be based on different frameworks such as TensorFlow, keras, and pytorch.

Swarm Network Node

The nodes form the blockchain network. This node interacts with each other in the platform and maintains the global state information about the model that is being trained and gathers the metadata written in the blockchain, and progressively share information. In starting, each node is to be registered in the network.

Swarm Learning CLI

It is a command line interface tool to the Swarm Learning framework. Here you can manage the swarm learning framework. You can connect and manage nodes using the port.

SPIFFE SPIRE Server Nodes

Secure Production Identity Framework for Everyone. This node provides security to the Swarm Learning Network. This identifies the framework for workload in an organization. It can establish a TLS connection or sign and verify the network.

HPE Official - Not for circulation

1.9 Steps for Swarm Learning

1. Node Registration
2. Train
3. Parameter Sharing
4. Parameter Merging
5. Repeat

Register

Nodes register to Swarm Network. The process begins with enrollment, or registration, in the Swarm smart contract by each node. This is a one-time process.

Each node subsequently records its relevant attributes in the contract, such as the uniform resource identifier (URI) from which its own set of trained parameters can be downloaded by other nodes.

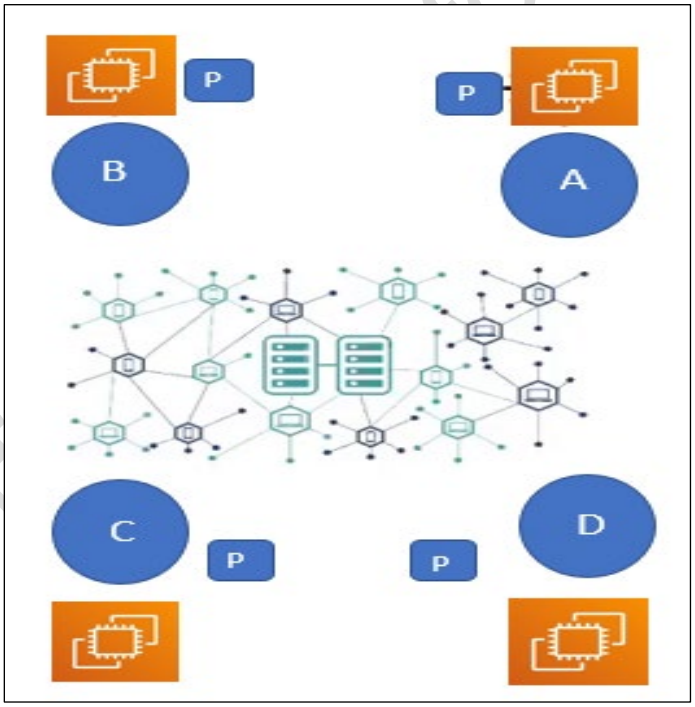


Figure 1.14 Register

HPE Confidential for circulation

Train

Nodes train a model on local data for certain epochs. In this node, proceed to train the local copy of the model iteratively over multiple epochs. During each epoch, every node trains its local model using one or more data batches, after which it exports the parameter values in a file and uploads it to a shared file system for other nodes to access.

Node signals other nodes that it is ready for the parameter-sharing step.

Parameter Sharing

Once the number of nodes that are ready for the parameter-sharing step. It begins with the process of electing the epoch leader, whose role is to merge the parameters derived after local training on all nodes. It is a star topology where a single leader performs the merge.

Parameter Merge

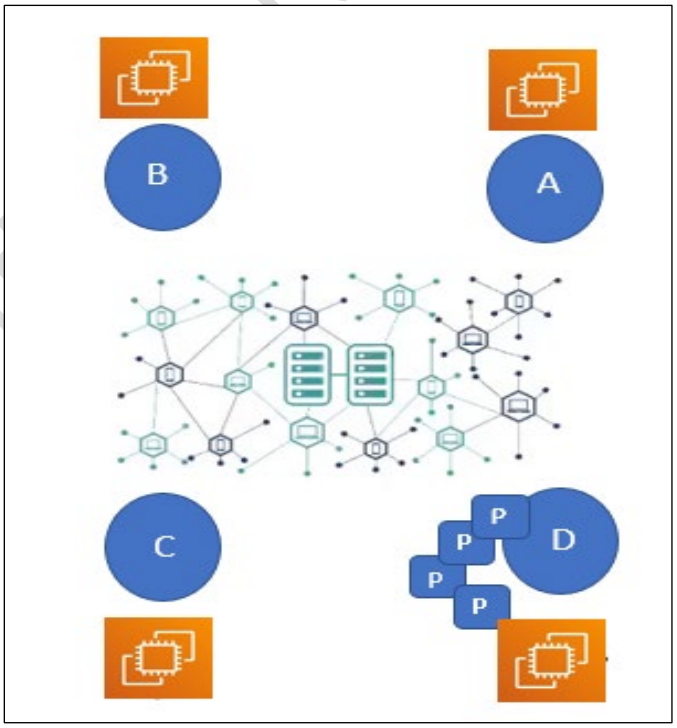


Figure 1.15 Parameter Merge

HPE Offi Not for circulation

Nodes share and merge the trained models at elected Node; here it is D. The framework supports multiple merge algorithms such as mean, weighted mean, median, and so on. Each node then downloads the file from the leader and updates its local model with the new set of parameter values.

1.10 Application of Swarm Intelligence

1.10.1 Ant-based routing

Inspiration

- The use of swarm intelligence in telecommunication networks has also been researched in the form of ant-based routing.
- It is inspired by the pheromone communication of the blind ants regarding a good path between the colony and the food source in an environment, the phenomenon known as stigmergy.
- The probability of the ant following a certain route is not only a function of pheromone intensity but also a function of distance to that city, the function known as visibility.

Strategy

- The objective of the strategy is to exploit historic, i.e., pheromone-based and heuristic information to construct candidate solutions each in a probabilistic stepwise manner and fold the information learned from constructing solutions into the history.
- The probability of selecting a component is determined by the heuristic contribution of the component to the overall cost of the solution, and the quality of the solution and history is updated proportionally to the quality of the best-known solution.

HPE Official - Not for circulation

Use Case

Route Optimization of Unmanned Aerial Vehicles (UAV).

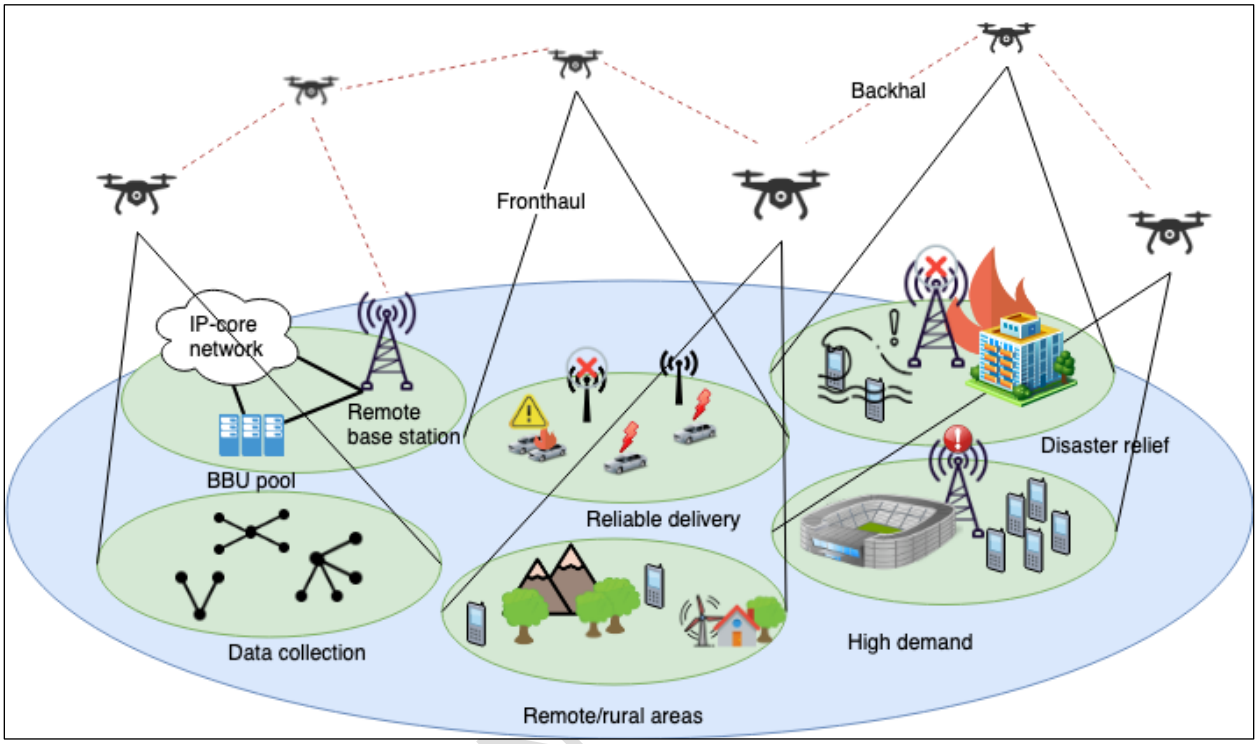


Figure 1.16 Route Optimization of Unmanned Aerial Vehicles (UAV)

- UAV is an aircraft without the onboard presence of pilots.
- It includes software and hardware agents that communicate or displace in an optimal manner.
- The UAVs are engaged in a simulated area coverage scenario with a defined set of waypoints. The objective is to find the shortest route that connects all the waypoints to optimize the time and the cost of the UAV's flight.
- Ant system algorithm is used for UAVs route optimization.
- It is being used in commercial applications such as telecommunications, ground traffic control, search and rescue operations, and crop monitoring, among others.
- UAVs assist with frost protection, irrigation, and crop management in agriculture.

HPE Official - N

- Together with Mobile Ground Station systems, UAVs offer persistent surveillance, enhanced situational awareness, and actionable intelligence to law enforcement and security personnel.

1.11 Blockchain: Element of Swarm Learning

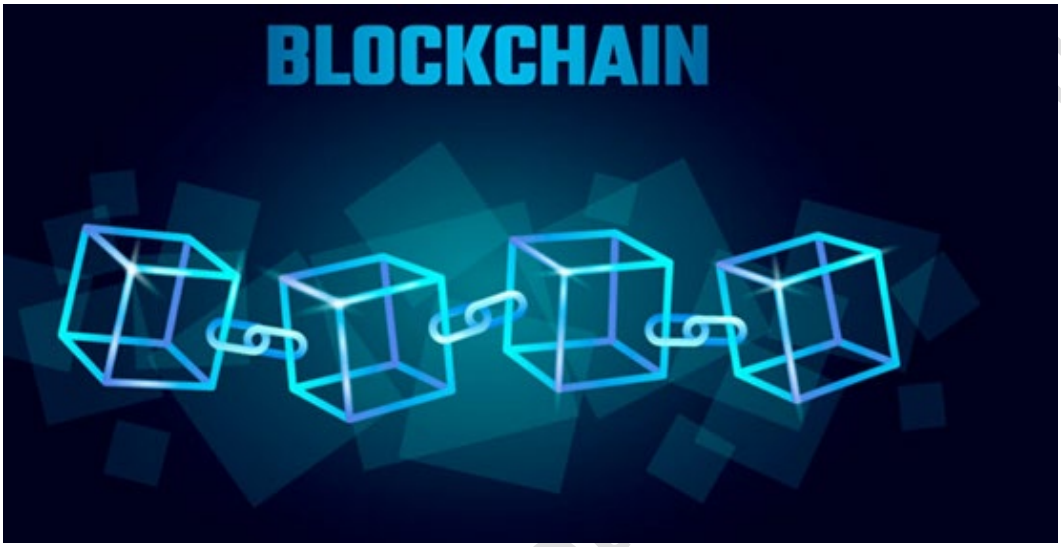


Figure 1.17 Blockchain

1.11.1 Major problems

Machine learning systems face a couple of problems.

- Risk of data confidentiality breach/data privacy regulations.
- Slowness and Cost for data duplication due to requiring a central server.

SL has the goal of facilitating the integration of any medical data from any data owner worldwide without violating privacy laws.

In Swam Learning, parameters are shared across the Swam Learning Network; also, what if data to train a model to detect special diseases is not sufficient or enough? Though in swarm learning, there is no central location in Swarm learning (SL). The training is performed locally/at the edge, just like in federated learning. But in the case of SL, not even the learnings are shared via a central dedicated server.

The Learning node will exchange the information using Block Chain Technology.

HPE Official - Not for Distribution

1.11.2 What Is a Blockchain?

A blockchain is a distributed database that is shared among the nodes of a computer network. As a database, a blockchain stores information electronically in a digital format. Blockchains are best known for their crucial role in cryptocurrency systems, such as Bitcoin, for maintaining a secure and decentralized record of transactions. The innovation of a blockchain is that it guarantees the fidelity and security of a record of data and generates trust without the need for a trusted third party.

One key difference between a typical database and a blockchain is how the data is structured. A blockchain collects information together in groups, known as blocks, which hold sets of information. Blocks have certain storage capacities and, when filled, are closed and linked to the previously filled block, forming a chain of data known as the blockchain. All the latest information that follows that freshly added block is compiled into a newly formed block that will then also be added to the chain once filled.

1.11.3 Key-Point

- Blockchain is a type of shared database that differs from a typical database in the way that it stores information; blockchains store data in blocks that are then linked together via cryptography.
- As new data comes in, it is entered into a fresh block. Once the block is filled with data, it is chained onto the previous block, which makes the data chained together in chronological order.
- Diverse types of information can be stored on a blockchain, but the most common use so far has been as a ledger for transactions.
- In Bitcoin’s case, blockchain is used in a decentralized way so that no single person or group has control—rather, all users collectively retain control.
- Decentralized blockchains are immutable, which means that the data entered is irreversible. For Bitcoin, this means that transactions are permanently recorded and viewable to anyone.

HPE Official - Not for circulation

1.11.4 How does blockchain work?

The goal of blockchain is to allow digital information to be recorded and distributed, but not edited. In this way, a blockchain is a foundation for immutable ledgers or records of transactions that cannot be altered, deleted, or destroyed. Therefore, blockchains are also known as distributed ledger technology (DLT).

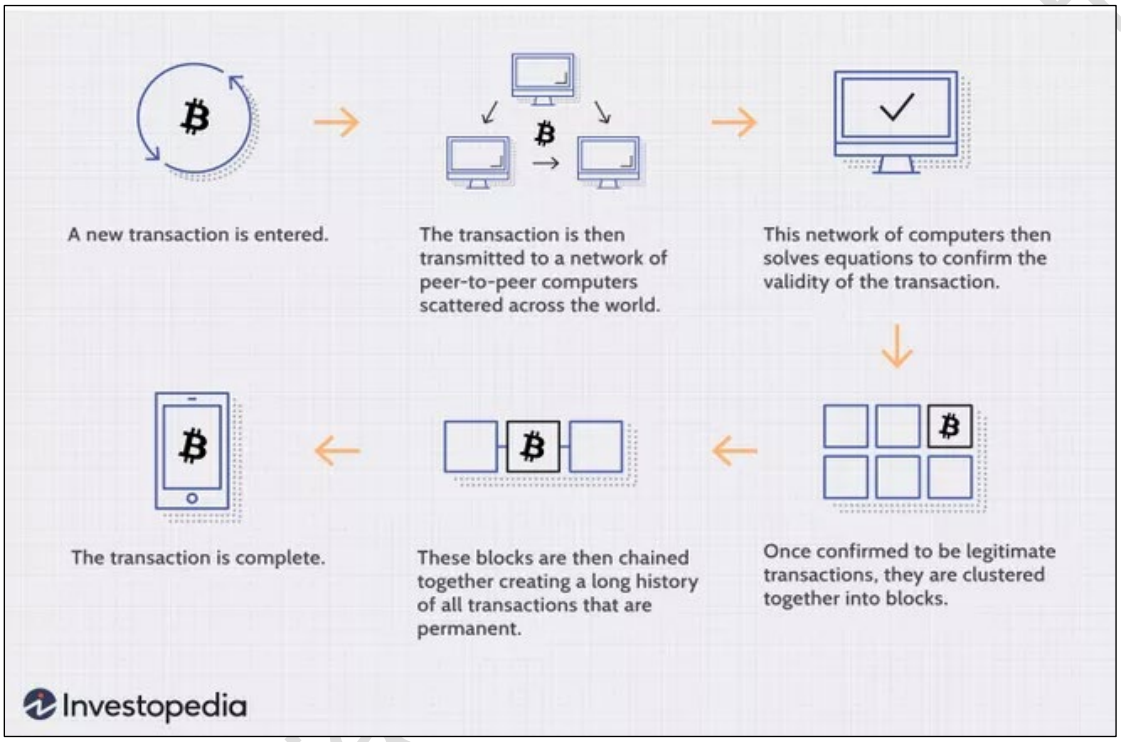


Figure 1.18 How does a Blockchain Work

SL has the goal of facilitating the integration of any medical data from any data owner worldwide without violating privacy laws. The approach involves leveraging a group of technologies, including edge computing and blockchain, to process data while removing the need for centralized coordination, thus preserving the confidentiality of the underlying information and allowing for collaboration that was not previously possible.

Public blockchains use compute-intensive consensus algorithms, like Proof of Work, to ensure the transactional guarantee across untrusted participants. For example, with a public blockchain, the consensus of truth is established by applying huge computing resources in a competition to solve a puzzle. Private, permissioned blockchain

HPE Official

implementations do not need such consensus algorithms as the participants are validated through identity.

Swarm Learning uses a private, permissioned blockchain that does not require such a consensus and therefore does not suffer from compute-intensive consensus algorithms.

1.12 Machine Learning Landscape

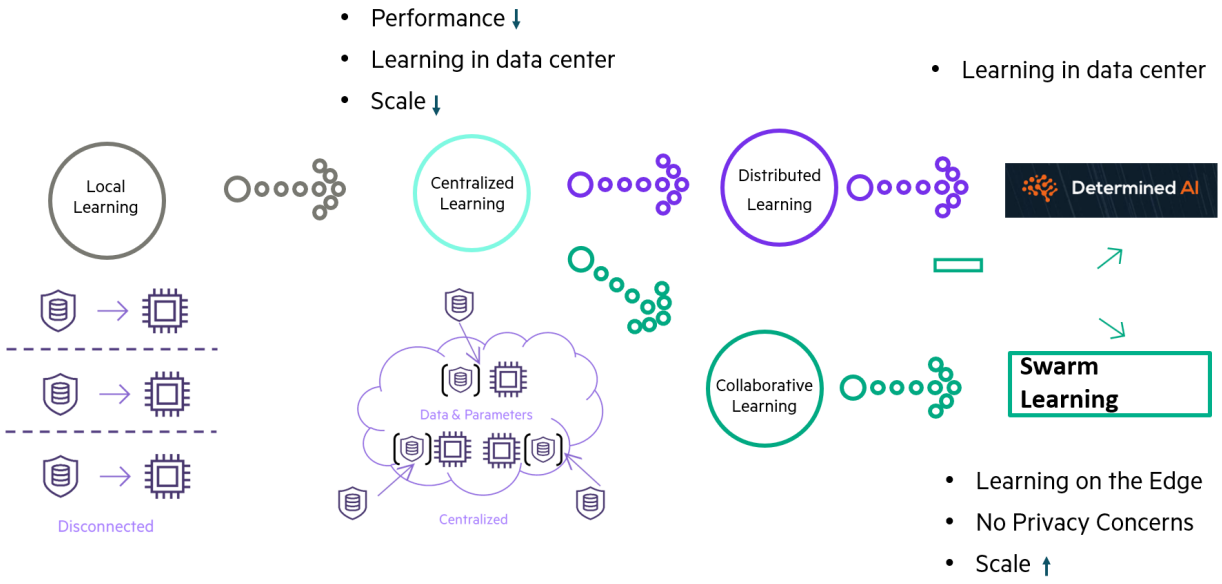


Figure 1.19 Machine Learning Landscape

The demerits of centralized learning can be approached in two ways; distributed learning is approached by determine.ai. The second approach is collaborative learning; the HPE solution is Swarm Learning.

1.12 Swarm Learning Value Proposition

The following table is a summary of value propositions by swarm learning.

HPE Official

Improved efficiency	Saving in network bandwidth; Near real-time learning & inferencing
Privacy Preserved ML	Data does not leave the origin
Collaborative Learning	Decentralized network of swarm nodes based on blockchain; Dynamic consortium membership
Learning with Biased Data	Localized learning with global state merge
Monetization Framework	Enable data owners to monetize the insights – Framework to enable incentives for sharing the model parameters

Figure 1.20 Swarm Learning Value Proposition

HPE Official - Not for ch

MODULE 2

END-END MACHINE LEARNING

HPE Official - Not for circulation

2.1 Pipeline:

A series of steps or actions put together is called a pipeline. Each step in a pipeline consumes the output of the previous step and generates an output that is provided to the subsequent step. In an ML Pipeline, each step may include actions related to data processing, model training, model testing, and the like. An example of an ML pipeline with four steps is shown in FIG. 1.

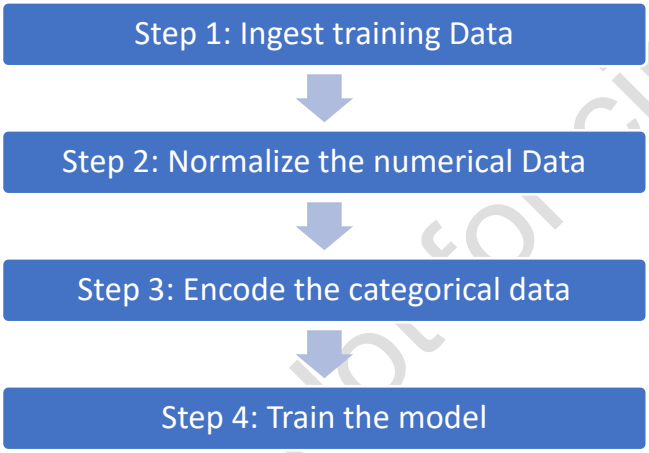


Figure. 2.1: Example of an ML Pipeline

In a Spark pipeline, at each step, new objects are created. For example, if the data frame comprises 5 columns and if we normalize 2 columns out of 5 columns at step 2 as shown in Fig. 3.1, we get 2 new columns as the output of step 2 as a new data frame. This is because of the transformation in the Spark do not occur in a place, and objects such as data frames are immutable.

Further, a Spark pipeline comprises a series of Estimators or Transformers at each step. The transformer uses an incoming data frame to produce a new data frame. For example, a trained ML model takes a data frame of features and produces a data frame of predictions. The Estimator uses an incoming data frame to produce a Transformer. For

HPE Official - Not for circulation

example, an ML algorithm takes a data frame of features and produces a trained ML model. In reference to Fig. 3.1, steps 1, 2, and 3 are regarded as Transformers in the pipeline, and step 4 is the Estimator in the pipeline.

2.2 Training Pipeline

The training pipeline refers to the process of creating a machine-learning model. The training pipeline comprises one or more stages relating to data processing, such as normalization, data cleaning, featuring engineering, encoding categorical features, etc., followed by a model training stage. At the end of the training pipeline, the trained model is generated that is used to make predictions on the live data either in real-time or in batches.

Example: Processing a Text Document

Step 1: Split the text in the document into words.

Step 2: Generate a feature vector for each of the unique words in the document.

Step 3: Train a prediction model using the feature vectors and the corresponding labels.

Figure. 2.2:Training Pipeline

In Fig. 3.2, the tokenizer and the hashing TF blocks are Transformers, and the Logistic Regression is the Estimator. However, the pipeline is an Estimator in the Spark framework. Therefore, after fitting the pipeline, we obtain a pipeline model Transformer. The pipeline

HPE Official - Not for circulation

model is used during the inference. In the case of multiple models in the pipeline, the transformer is called after fitting the model in the pipeline.

2.3 Inference Pipeline

The inference pipeline is a process of utilizing the trained machine learning model to make a prediction for the live data. The inference pipeline, when deployed, refers to a task "operationalizing an ML model." The deployed inference pipelines help to achieve the business goal/objective.

Continuing with the example of Processing a Text Document, the pipeline_model shown in Fig. 3 is the inference pipeline. The inference pipeline has the same number of stages as compared to the training pipeline. But the inference pipeline comprises only Transformers. All the Estimators in the training pipeline are converted into Estimators in the inference pipeline.

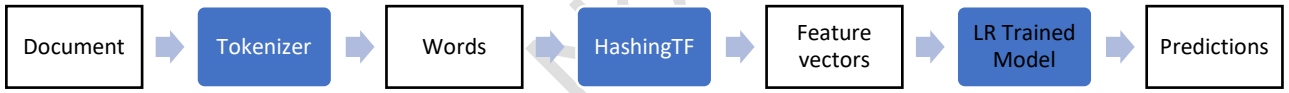


Figure. 2.3:Inference Pipeline

2.4 Model Deployment

2.4.1 Microservices

The ML models are deployed with the Microservices architecture when the solution to the business objective is realized via two or more ML models. For example, consider an e-commerce application that allows users to browse the products and select and purchase one or more products. The organization hosting the e-commerce application collects the data relating to the user browsing, purchases made by the user, etc. Further, the organization wishes to deploy multiple solutions such as recommending new products to

HPE Official - Not for circulation

the user based on the browsing history, providing discounts based on the purchase pattern of the user, forecasting the type of products that will be in demand, etc. In such a situation, if the ML models are independently built with different code bases, or running on different infrastructures, then microservices architecture can be used to deploy the ML models.

The advantages are:

- Each service or ML solution can be debugged, patched (i.e., modified), and deployed individually and independently. Therefore, the entire system need not be disturbed.
- This will overcome the issues of single-point failure.
- Eases the managing and maintaining of the services.
- Seeds up the development of additional and complex services.

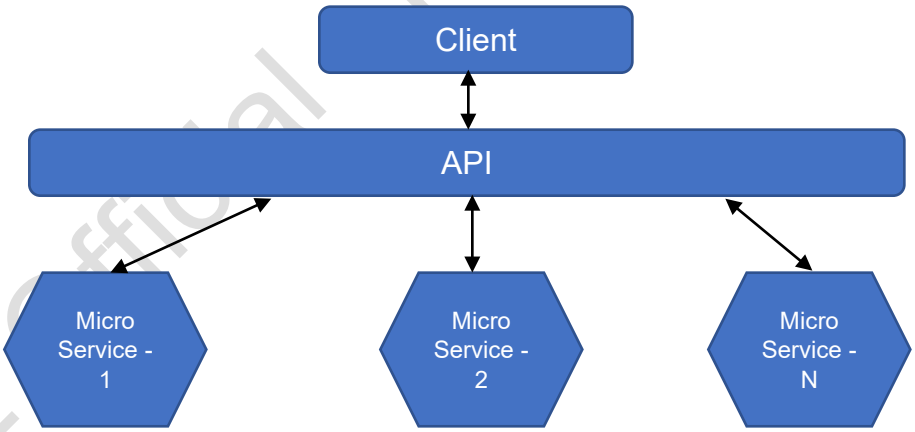


Figure 2.4 Microservices

HPE Official - Not for circulation

2.4.2 Batching

This is one of the most widely used architectures in ML deployment. The batching technique is used when the data for prediction arrives at regular time intervals or batches. The ML models are scheduled to run/generate predictions for the batches of data. This technique is suitable when the applications do not have a low-latency requirement. For example, consider a clinical laboratory where all samples are collected in the morning, and the data derived from the samples are sent in batches for predicting sugar levels, disease prediction, etc.

The advantages are:

- Efficient management of computer resources.
- A large amount of data at runtime enables to performance of tasks such as anomaly detection, clustering, outlier detection, etc.
- The batch data can be optimized for efficiency using parallel and distributed computing.

2.4.3 Event-based

The individual actions in the ML pipeline trigger other actions in the system. When the input data is stochastic, and the predictions of the first model need to be consumed by the second model, the event-based architectures are useful. For example, in a gaming application, the server needs to track the user's actions for making decisions on the gameplay. There are two types of event-based architecture, namely, Event streaming and Publisher/Subscriber (Pub/Sub).

The Pub/Sub model makes use of a broker where the publisher communicates asynchronously with the subscriber. The Publishers send data to the event broker, who in turn shares the data with the appropriate subscriber service. For example, Apache Kafka.

HPE Official - Not for circulation

In the event streaming model, the continuous data is processed in near real-time. The data is processed by the system as and when it is received without being stored (persisted) in a database. For Example, Spark streaming.

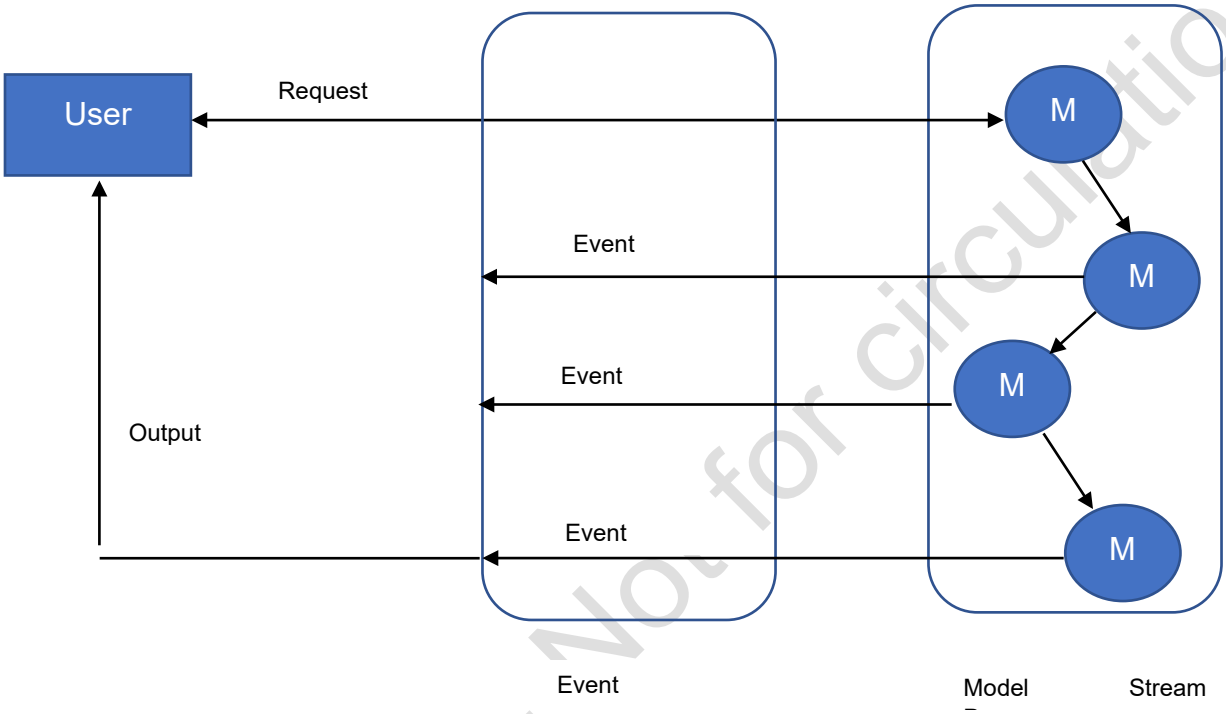


Figure 2.5 Streaming Process

2.5 Containerization

The ML models built using Python language need to be aware of the environmental requirements before deploying the service. This is because python programs are not exported as standalone executables. The Python interpreter needs the relevant libraries and support packages for executing the code. Therefore, containerization helps to package the Python program along with the dependencies as a standalone unit and can be executed on any platform. For example, Docker.

HPE Official - Not for circulation

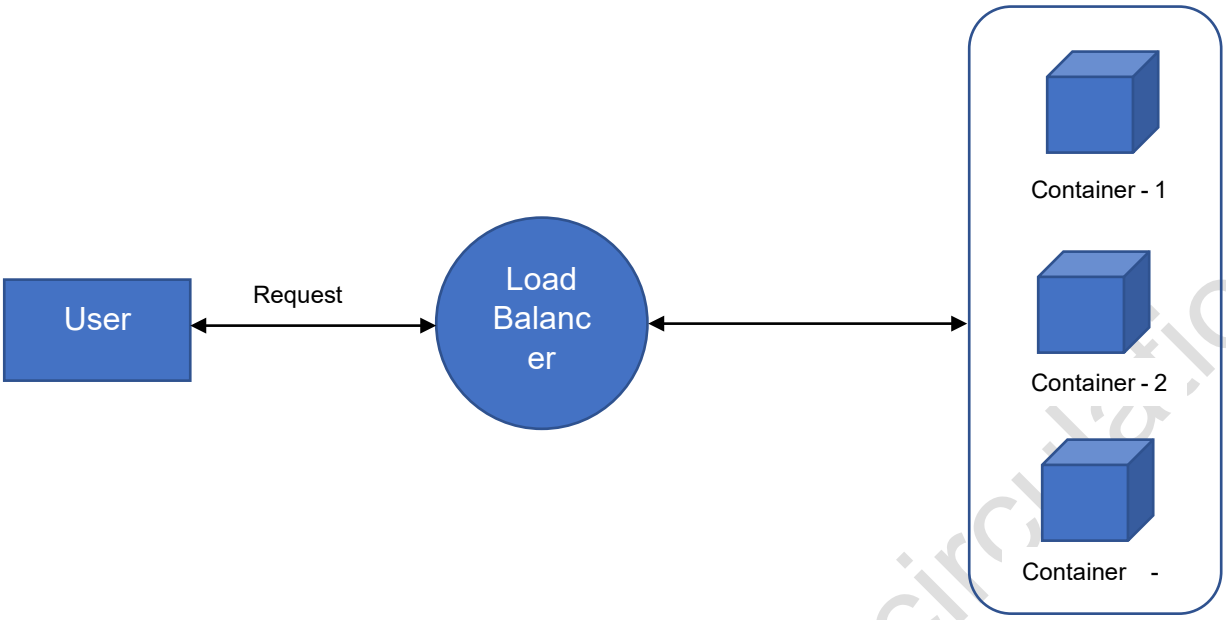


Figure. 2.6:Containerization

2.6 MLOps in Spark

MLOps (Machine Learning Operations) is the practice of combining the lessons learned from DevOps with Machine learning for the productionalization of the trained models. MLOps are essential for standardizing and streamlining the steps involved in the life cycle management of a machine learning model. The role of MLOps is to fill the gap between the data scientist and the machine learning consumers. The MLOps is required for the following reasons:

- Reproducibility of ML/DL
- Automation ML/DL pipelines
- Versioning ML/DL models and Data
- ML/DL Lifecycle management
- Monitoring ML/DL.

HPE Official - Not for circulation

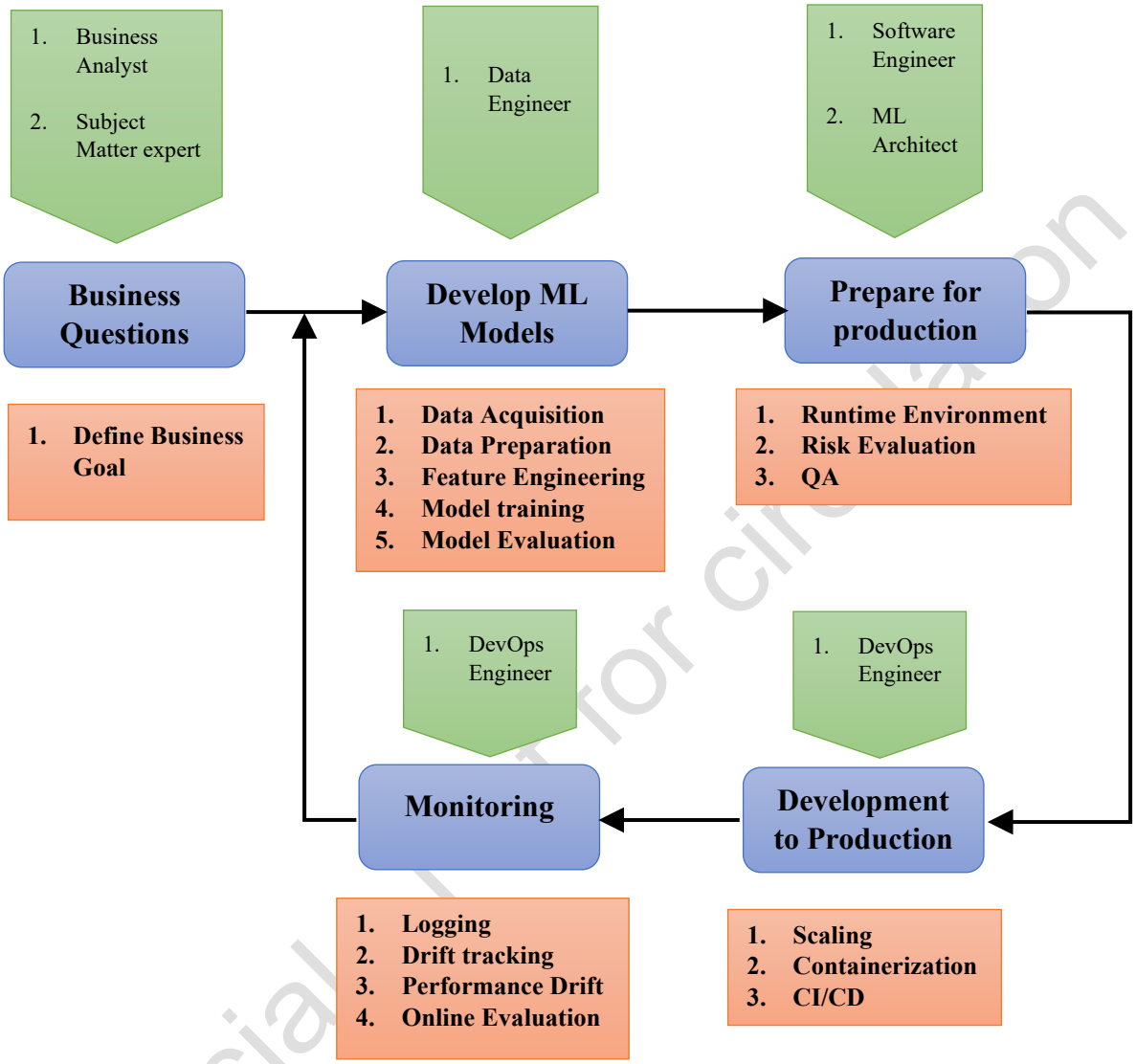


Figure.2.7 Life Cycle of a Machine Learning Model

The process of developing a machine learning model starts with a need to solve a business problem/objective. For example, reducing fraudulent financial transactions to a value lesser than 0.1%, and recognizing the faces from a social media post. The business goals are characterized by the Key Performance Indicators (KPIs). For example, Reducing the server downtime to reduce the revenue loss by $N. The KPIs are used to frame the ML model and evaluate the performance of the ML model in production.

HPE Official Use for circulation

The business analysts and subject matter experts provide business goals, KPI for training an ML model and evaluate and ensure the model performance aligns with the business needs. Further, they flag the results that are not aligned with the business objectives.

The Data Engineer/Data Scientist/ML Architect is responsible for managing the flow of data into and outside of the ML model, optimizing the retrieval of data and use of data, investigating the data pipeline, assessing the quality of the ML model, performing model packaging, and the like.

The Software Engineer integrates the trained ML models with the applications of the organization. They ensure that the ML models work seamlessly with other applications hosted by the organization. They perform automated testing and versioning of the models.

The DevOps Engineer creates and maintains a CI/CD pipeline, integration of MLOps with the strategy of the organization, deploys the ML pipeline, and test the security, performance, and availability of the system.

The process involved in the ML lifecycle management is detailed below.

2.6.1 Develop the models

The different components in building an ML model are choosing the type of algorithm for the given task, good training data, the performance metric to be optimized, hyperparameters for the ML model, etc.

Some of the tasks in developing the ML models having good generalization capacity are data exploration, feature engineering, and selection, bias-variance tradeoff, evaluating and comparing the models, responsible AI, etc.

2.6.2 Prepare for Production

The developed ML model is not guaranteed to work in a real-world scenario. The reason being the production environment might be different from the development environment. The things to consider before sending the model to production are given below.

The model should be provided in the format required for production. For example, the model developed using python might require conversion to Java-based environment. The

HPE Official - Not for circulation

reason for the development and runtime environment to be different maybe performance considerations such as latency, target device for deployment such as low-power devices, etc. Therefore, model quantization, pruning, and distillation may be required.

Accessing the data in real time during inferencing requires access to a database. For example, look up at the average prices of an apartment associated with a zip code. Thus, network connectivity, appropriate drivers, libraries required to communicate with the database, authentication credentials, etc., must be provided in the production environment.

2.6.3 Development to Production

The principles and practices of agile software development include continuous integration and continuous delivery for faster release of applications/updates and better quality and risk control.

The deployment process includes the following steps:

Integration: It is a process of combining/adding one or more additional features to the existing model and performing some complex tests.

Delivery: It is the process of building a fully packaged and validated version of the ML model. The packaged model is ready for deployment.

Deployment: It is a process of running a new model on the target infrastructure. The two main modes of deployment are batch scoring and real-time scoring.

Release: It is a process of directing the production or real-time workload to the deployed model.

Monitoring: The deployed model must be monitored as the performance of the ML model can degrade with time. Monitoring at the resource level includes ensuring that the ML model is running correctly in the production environment, ensuring the system is alive, ensuring the CPU, RAM, network usage, and disk space is available, ensuring that the requests are being processed at the expected rate by the ML model.

HPE Official - Not for circulation

Monitoring at the performance level, includes monitoring the pertinence of the model over time, ensure that the model is still an accurate representation of the pattern of new incoming data, ensuring that the model is performing well as it did during the development phase.

The key challenges in managing the life cycle of machine learning models are

- Data Drift
- Change is business needs
- Ensure the model in production meets the original goal

Training:

a. Train-run

In the train-run technique, the training and the prediction are in the same process. The model can be trained either in batch mode or incremental mode. The entire training process is run before generating the predictions. Therefore, applications requiring low-latency cannot follow the train-run process. However, in applications where the machine learning models are light-weight or require use of recently available data before making predictions, or applications where batch process in executed infrequently the train-run technique is used. The advantages of train-run technique are

- Overcome the problems associated with Drift (performance and data).
- Save development time.
- Reduced complexity.

HPE Official - Not for circulation

•

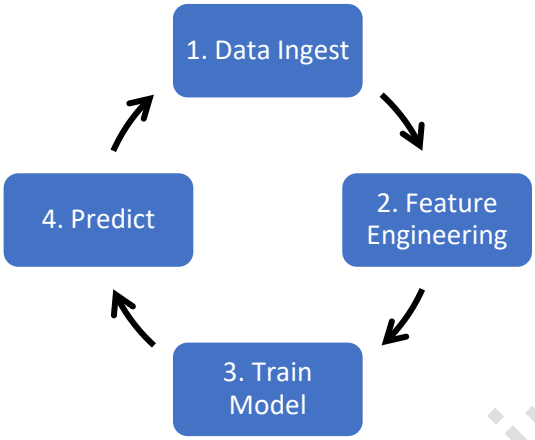


Figure. 2.8 Train – Deploy

b. Train-persist

In the train-persist technique, the model is trained in batches, and the predictions can be generated in any mode (batch or real-time). The model used for predictions is received from the storage.

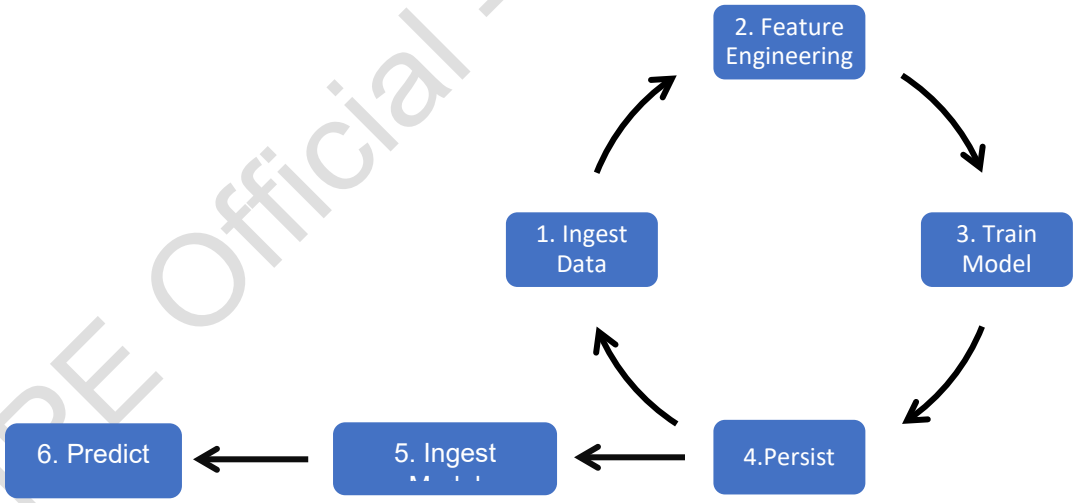


Figure. 2.9 Train Persist Model

HRE Official - Not for circulation

Re-training the models:

The performance of the model can degrade over time. The main reason for the performance degradation is “Drift.” There are two main types of drifts, namely, Concept Drift and Data Drift.

a. The concept of drift refers to changing the relationship between the features associated with the data and predictions performed by the model. The concept of drift is also known as covariate drift.
    - Example 1: At the training time, the relationship between one or more features and the predictions was a linear relationship. But post-deployment, with a lot of input data, the relationship turns out to be a non-linear relationship.
    - Example 2: The measurements made by sensors and valves in an industrial process can vary with time. The accuracy and precision of the measurements are reduced.
b. Data drift occurs when the statistical properties of the input data change with time.

Registering:

An ML Model can be registered with the Model Registry. A registered ML model has a unique name and contains versions, associated transitional stages, model lineage, and other metadata.

Each registered model can have one or many versions. When a new model is added to the Model Registry, it is added as version 1. Each new model registered to the same model's name increments the version number.

Before you can add a model to the Model Registry, you must log in using the log model methods of the corresponding model flavors. Once a model has been logged, you can add, modify, update, transition, or delete a model in the Model Registry through the UI or the API.

Registering a Model Using MLFlow

a. Using MLFlow spark log model:

HPE Official - Not for circulation

This approach is used to register a model that is trained using the MLflow run. If a registered model with the name does not exist, the method registers a new model and creates Version 1. If a registered model with the name exists, the method creates a new model version.

b. Using mlflow. register_model:

This approach is used when all the experiment runs are complete and the best model is selected for registering. The experiments must be run with MLFlow. If a registered model with the name does not exist, the method registers a new model, creates Version 1, and returns a ModelVersion MLFlow object. If a registered model with the name exists, the method creates a new model version and returns the version object.

c. Using create_registered_model:

This approach is used after running the experiments. The experiments must be run with MLFlow. If the model's name exists, this method will throw a MlflowException because creating a new registered model requires a unique name.

d. For a model trained outside of MLFlow:

Log the model by setting a tracking uri (mlflow.set_tracking_uri) and use the log_model function for registering.

Fetching a Model Using MLFlow

The registered model can be fetched using the function mlflow. spark.load_model, with URI as the input. The model version number and staging can also be used to fetch the specific model.

Deleting a Model Using MLFlow

The registered model, along with all the versions, can be deleted using the function delete_registered_model. To delete only the specific versions of the registered model, use the function delete_model_version.

Advantages of versioning and registering:

HPE Official - Not for circulation

- ML models can be retrained based on new training data.
- Models can be quickly rolled back to a previous serving version.

Hyperparameter tuning

This is a process of finding the best set of parameters for the Estimator, such as ML model, or for the entire training pipeline. The creation of the pipeline enables the user to tune all the stages of the pipeline at once. Without the pipeline, the best set of parameters for each stage must be determined individually.

Further, for selecting the best model for a given task, the spark framework provides CrossValidator and TrainValidationSplit techniques. For both techniques, an Estimator such as a pipeline or an ML model, a set of parameters to choose from, and an Evaluator, i.e., a metric for evaluating the pipeline, is required.

The general process of hyperparameter tuning works as follows,

- The input data is split into one or more training and testing sets.
- For each <Train, Test> pair of sets, the Estimator is iterated over the set of parameters.
- The performance of the model is evaluated on the test set.
- The best-performing model is selected based on the metric provided by the Evaluator.

Cross-Validation:

The dataset is split into pairs of training and testing sets, based on the value of “k.” In each pair, the dataset is divided into “k” partitions of equal size and one of the partitions is used for testing and the remaining “k-1” partitions are used for training. The testing partition is sequentially selected in pairs of training and testing sets. For example, if the value of “k = 4”, then 4 pairs of training and testing sets are created. In each pair, the dataset is divided into 4 partitions. Further, the first partition in the first pair is used for testing and the remaining three are used for training. In the second pair, the second

HPE Official - Not for circulation

partition is used for testing and the remaining three is used for training. Similarly, in third and fourth pairs.

One model is trained using each pair of training and testing sets and the average of the evaluation metrics is computed for the model. This is used with the average evaluation metric of the other models for selecting the best model.

Train-Validation Split:

In this approach, the dataset is split into one pair of training and testing sets based on the value of “trainRatio” (a value between 0 and 1). Further, one model is trained using the training and testing sets. For example, if the “train Ratio = 0.73,” then 73% of the dataset is used for training and 27% of the dataset is used for testing.

Parameter Grid:

The parameter grid comprises a set of values to be tried for each hyperparameter during the model training. Each combination of the values in the parameter grid is tried on the pipeline. For example, if there are 2 hyperparameters say H1 and H2. If H1 has 2 values and H2 has 3 values, then with trainvalidationsplit approach 6 models are trained. With the cross-validation approach if the value of “k = 3”, then 18 models are trained and averaged to obtained 6 models.

Further, the set of values in the parameter grid are evaluated serially by default. However, parallel evaluation can be performed by setting a value to the “parallelism” parameter based on the size of the cluster deployed.

Evaluator:

The Evaluator can be one of a RegressionEvaluator, BinaryClassificationEvaluator, MultiClassClassificationEvaluator, MultilabelClassificationEvaluator, and a RankingEvaluator. The Evaluator includes a default performance metric for selecting the best set of parameters. This can be modified by using the setMetricName in the Evaluator.

HPE Official – Not for circulation

Publishing

With MLFlow:

Start the mlflow tracking server by executing the command “mlflow server” in the /tmp directory. Next, run the required model using the python command.

Web service:

When there is a need to respond to user requests in real-time, the ML model can be deployed as a Web Service. The ML model is set up as a web endpoint and the services and applications can invoke the endpoint.

Some of the python packages required for building a Web endpoint are “requests” for providing functions for GET and POST commands. The “flask” package for Enabling functions to be exposed as HTTP endpoints. The “gunicorn” package that enables hosting Flask apps in production environments. The “mlflow” package that provides model persistence. The “pillow” package that is a fork of the Python Imaging Library. The “dash” package that enables writing interactive web apps in Python. The

Steps involved in deploying the ML model as an HTTP endpoint:

1. Set up a web service using Flask, Gunicorn, and Dash.
2. Load the pre-trained ML models.
3. Create the feature vector required to be provided as input to the ML model.
4. Obtain and append the model output under the “response” key.

Containerization:

A container is a unit of software the comprises all the code and its dependencies required to perform a given task. The containers reproduce the same environment used for training the ML models during the deployment as well. It is an alternative to Virtual Machines. For example, docker, Elastic Container Service (ECS), Elastic Kubernetes Services (EKS) can be used for containerized deployment of the ML models.

HPE Official - Not for circulation

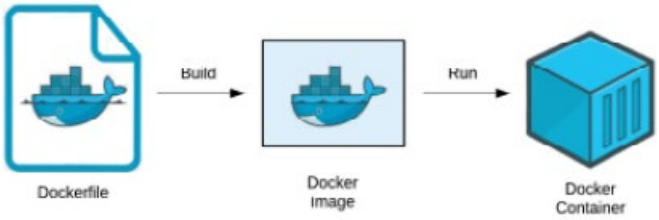


Figure 2.10 Docker

Using docker, a docker image is built using a docker file. The docker file is a text document containing a set of docker commands provided by the developer. The docker commands are used to build the docker image. The docker image is a collection of runtime libraries, filesystems, etc. The docker container is an instantiated image that is runs the application.

Steps in deploying the trained model over HTTP using docker:

1. Create a requirements.txt file with the requirements dependencies
2. Create a Docker file to install the dependencies from the requirements file, mention the directory for the web application.
3. Place the web application files in the working directory along with the trained model files.
4. Build the container.
5. Verify that the image is created
6. Run the container.

2.7 Pachyderm

Data scientists can build an end-to-end machine learning workflow using the Pachyderm data science platform, which covers the key phases of the machine learning life cycle from data ingestion to production.

The most crucial Git and Pachyderm ideas share many similarities if you are familiar with Git, a version control and life cycle system for code. For thousands of engineers worldwide, version control tools like Git and its hosted variant GitHub have become industry standards. Git gives you the ability to record the changes you have made to your code and go back at any time. A tool that allows data scientists to track the versions of their research is something they should have.

HPE Official - Not for circulation

2.8 Pachyderm Pipeline Specification

An automated approach called a machine learning (ML) pipeline enables you to run the same code repeatedly against various sets of data and settings. A pipeline makes sure that each cycle is automated and follows the same set of procedures. A single configuration file known as the pipeline specification, or the pipeline spec, defines an ML pipeline in Pachyderm just like it does in many other technologies.

The most crucial configuration in Pachyderm is the Pachyderm pipeline specification since it specifies what your pipeline performs, how frequently it runs, how the work is distributed among Pachyderm employees, and where to output the result.

2.9 Creating Repo and Pipeline using Pachyderm

A repo is the highest-level data primitive in Pachyderm. Like many things in Pachyderm, it shares its name with primitives in Git and is designed to behave analogously. Repos should be dedicated to a sole source of data such as log messages from a particular service, a user's table, or training data for an ML model. Repos are dirt cheap so do not be shy about making tons of them. For this demo, we will simply create a repo called images to hold the data we want to process:

```
$ pachctl create-repo images

# See the repo we just created
$ pachctl list-repo
NAME                CREATED             SIZE
images              2 minutes ago       0 B
```

Adding Data to Pachyderm

Now that we have created a repo it is time to add some data. In Pachyderm, you write data to an explicit commit (again, like Git). Commits are immutable snapshots of your data which give Pachyderm its version control properties.

Files can be added, removed, or updated in each commit.

Let us start by just adding a file, in this case an image, to a new commit. We have provided some sample images for you

that we host on Imgur.

HPE Online - Not for circulation

We will use the put-file command along with two flags, -c and -f. -f can take either a local file or a URL which

it will automatically scrape. In our case, we will simply pass the URL.

Unlike Git though, commits in Pachyderm must be explicitly started and finished as they can contain massive amounts of

data and we do not want that much “dirty” data hanging around in an unpersisted state. The -c flag specifies that we

want to start a new commit, add data, and finish the commit in a convenient one-liner.

We also specify the repo name “images,” the branch name “master,” and what we want to name the file, “liberty.png.”

```
$ pachctl put-file images master liberty.png -c -f http://imgur.com/46Q8nDz.png
```

Finally, we check to make sure the data we just added is in Pachyderm

```
# If we list the repos, we can see that there is now data
$ pachctl list-repo
NAME                CREATED             SIZE
images              5 minutes ago   57.27 KiB

# We can view the commit we just created
$ pachctl list-commit images
REPO                ID                                PARENT              STARTED
↪         DURATION            SIZE
images              7162f5301e494ec8820012576476326c  <none>              2 minutes
↪ago      38 seconds          57.27 KiB

# And view the file in that commit
$ pachctl list-file images master
NAME                TYPE                SIZE
liberty.png         file                57.27 KiB
```

We can view the file we just added to Pachyderm. Since this is an image, we cannot just print it out in the terminal, but the following commands will let you view it easily.

```
# on OSX
$ pachctl get-file images master liberty.png | open -f -a /Applications/Preview.app

# on Linux
$ pachctl get-file images master liberty.png | display
```

Create a Pipeline

Now that we have some data in our repo, it is time to do something with it. Pipelines are the core processing primitive in Pachyderm and they are specified with a JSON encoding. For this example, we have already created the pipeline for you and you can find the code on GitHub. When you want to create your own pipelines later, you can refer to the full

Pipeline Specification to use more advanced options. This includes building your own code into a container instead of the pre-built Docker image we will be using here.

For now, we are going to create a single pipeline that takes in images and does some simple edge detection.

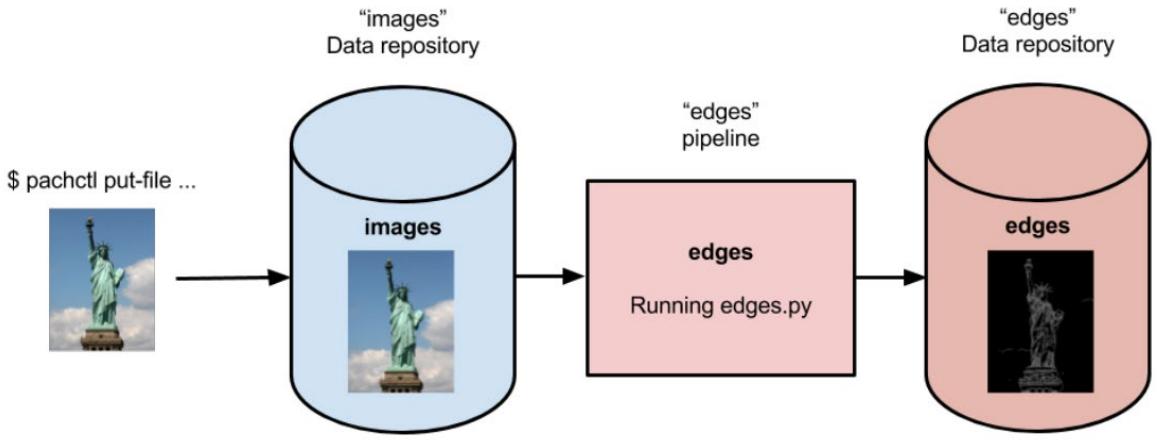


Figure 2.11 Pipeline Structure

Below is the pipeline spec and python code we are using. Let us walk through the details.

```
# edges.json
{
  "pipeline": {
    "name": "edges"
  },
  "transform": {
    "cmd": [ "python3", "/edges.py" ],
    "image": "pachyderm/opencv"
  },
  "input": {
    "atom": {
      "repo": "images",
      "glob": "/*"
    }
  }
}
```

Our pipeline spec contains a few simple sections. First is the pipeline name, edges. Then we have the transform

which specifies the docker image we want to use, pachyderm/opencv (defaults to DockerHub as the registry),

and the entry point edges.py. Lastly, we specify the input. Here we only have one "atom" input, our images repo

with a particular glob pattern.

The glob pattern defines how the input data can be broken up if we wanted to distribute our computation. /* means

that each file can be processed individually, which makes sense for images. Glob patterns are one of the most powerful

features of Pachyderm so when you start creating your own pipelines, check out the Pipeline Specification.

```
# edges.py
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os

# make_edges reads an image from /pfs/images and outputs the result of running
# edge detection on that image to /pfs/out. Note that /pfs/images and
# /pfs/out are special directories that Pachyderm injects into the container.
def make_edges(image):
    img = cv2.imread(image)
    tail = os.path.split(image)[1]
```

```
    edges = cv2.Canny(img,100,200)
    plt.imsave(os.path.join("/pfs/out", os.path.splitext(tail)[0]+'.png'), edges, cmap␣
↪= 'gray')

# walk /pfs/images and call make_edges on every file found
for dirpath, dirs, files in os.walk("/pfs/images"):
    for file in files:
        make_edges(os.path.join(dirpath, file))
```

Our python code is straight forward. We are simply walking over all the images in /pfs/images, do our edge detection and write to /pfs/out. /pfs/images and /pfs/out are special local directories that Pachyderm creates within the container for you. All the input data for a pipeline will be found in /pfs/ and your code should always write out to /pfs/out. Pachyderm will automatically gather everything you write to /pfs/out and version it as this pipeline's output. Now let us create the pipeline in Pachyderm:

```
$ pachctl create-pipeline -f https://raw.githubusercontent.com/pachyderm/pachyderm/
↪master/doc/examples/opencv/edges.json
```

2.10 What Happens When You Create a Pipeline

Creating a pipeline tells Pachyderm to run your code on the data currently in your input repo (the HEAD commit) as

well as all future commits that happen after the pipeline is created. Our repo already had a commit, so Pachyderm

automatically launched a job to process that data.

This first time Pachyderm runs a pipeline job, it needs to download the Docker image (specified in the pipeline spec)

from the specified Docker registry (Docker Hub in this case). As such, this first run this might take a minute or so,

depending on your Internet connection. Subsequent runs will be much faster.

You can view the job with:

```
$ pachctl list-job
ID                               OUTPUT COMMIT                                STARTED
↪DURATION   RESTART PROGRESS  DL       UL       STATE
490a28be32de491e942372018cd42460 edges/bc2d20d0c23740f397622a62b0978c57 2 minutes ago
↪35 seconds 0       1 + 0 / 1 57.27KiB 22.22KiB success
```

Yay! Our pipeline succeeded! Notice, that there is an OUTPUT COMMIT column specified above. Pachyderm creates a corresponding output repo for every pipeline. This output repo will have the same name as the pipeline, and all the results of that pipeline will be versioned in this output repo. In our example, the “edges” pipeline created a repo called “edges” to store the results.

```
$ pachctl list-repo
NAME                CREATED             SIZE
edges               2 minutes ago       22.22 KiB
images              10 minutes ago      57.27 KiB
```

Reading the Output, we can view the output data from the “edges” repo in the same fashion that we viewed the input data

```
# on OSX
$ pachctl get-file edges master liberty.png | open -f -a /Applications/Preview.app

# on Linux
$ pachctl get-file edges master liberty.png | display
```

The output should look like:

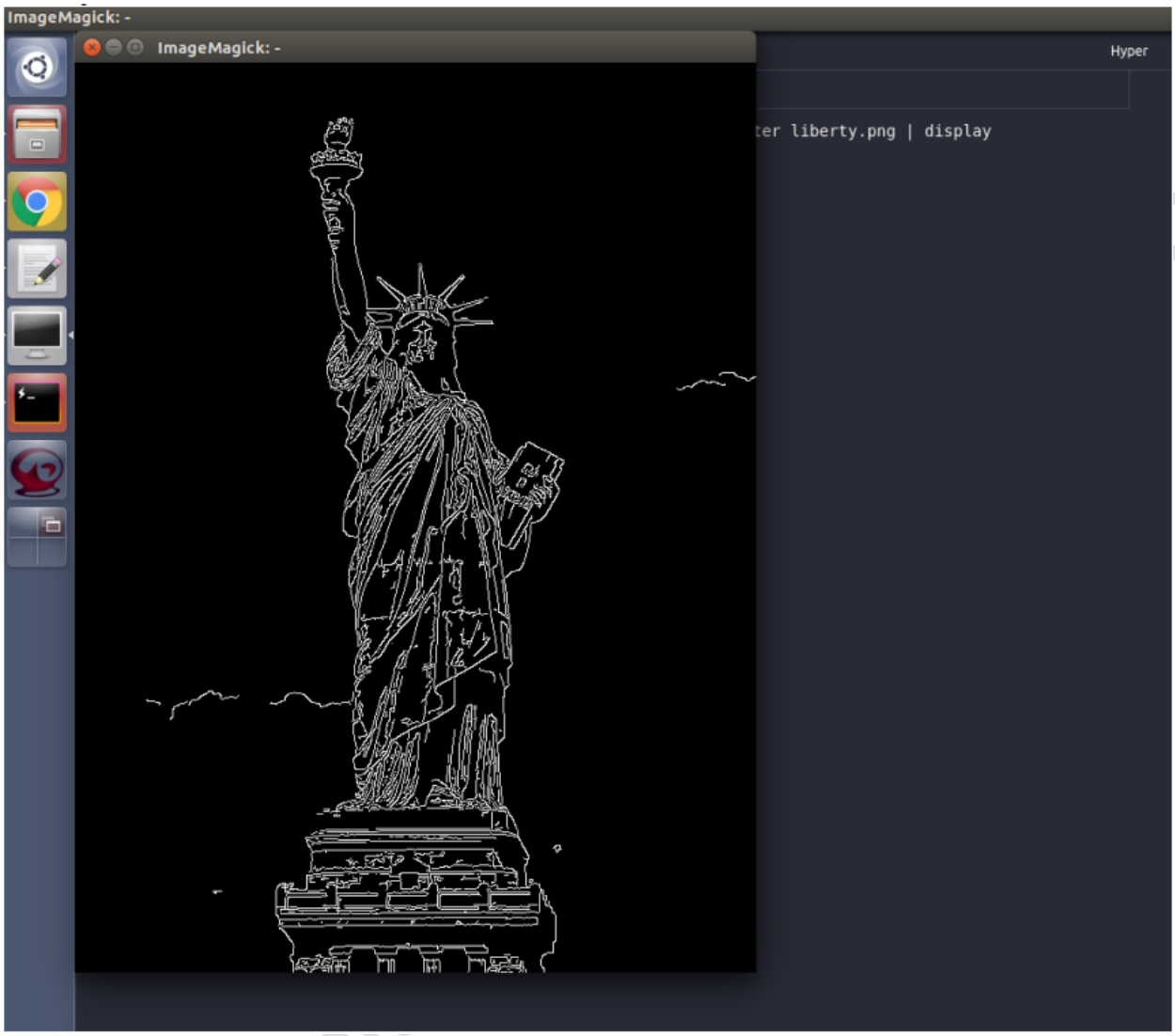


Figure 2.12

Processing More Data

Pipelines will also automatically process the data from new commits as they are created. Think of pipelines as being subscribed to any new commits on their input repo(s). Also, like Git, commits have a parental structure that tracks which files have changed. In this case we are going to be adding more images. Let us create two new commits in a parental structure. To do this we will simply do two more put-file commands with -c and by specifying master as the branch, it will automatically parent our commits onto each other. Branch names are just references to a particular HEAD commit.

HPE Offic

```
$ pachctl put-file images master AT-AT.png -c -f http://imgur.com/8MN9Kg0.png

$ pachctl put-file images master kitten.png -c -f http://imgur.com/g2QnNqa.png
```

Adding a new commit of data will automatically trigger the pipeline to run on the new data we have added. We will see

corresponding jobs get started and commits to the output “edges” repo. Let us also view our new outputs.

```
# view the jobs that were kicked off
$ pachctl list-job
ID                                OUTPUT COMMIT                           STARTED
 ↪ DURATION           RESTART PROGRESS  DL       UL       STATE
81ae47a802f14038b95f8f248cddbed2 edges/146a5e398f3f40a09f5151559fd4a6cb 7 seconds ago
 ↪ Less than a second 0       1 + 2 / 3 102.4KiB 74.21KiB success
ce448c12d0dd4410b3a5ae0c0f07e1f9 edges/c5d7ded9ba214d9aa4aa2c044625198c 16 seconds
 ↪ago Less than a second 0       1 + 1 / 2 78.7KiB  37.15KiB success
490a28be32de491e942372018cd42460 edges/bc2d20d0c23740f397622a62b0978c57 9 minutes ago
 ↪ 35 seconds         0       1 + 0 / 1 57.27KiB 22.22KiB success
```

```
# View the output data

# on OSX
$ pachctl get-file edges master AT-AT.png | open -f -a /Applications/Preview.app

$ pachctl get-file edges master kitten.png | open -f -a /Applications/Preview.app

# on Linux
$ pachctl get-file edges master AT-AT.png | display

$ pachctl get-file edges master kitten.png | display
```

ADDING ANOTHER PIPELINE

We have successfully deployed and utilized a single stage Pachyderm pipeline, but now let us add a processing stage

to illustrate a multi-stage Pachyderm pipeline. Specifically, let us add a montage pipeline that take our original and

edge detected images and arranges them into a single montage of images:

HPE Official

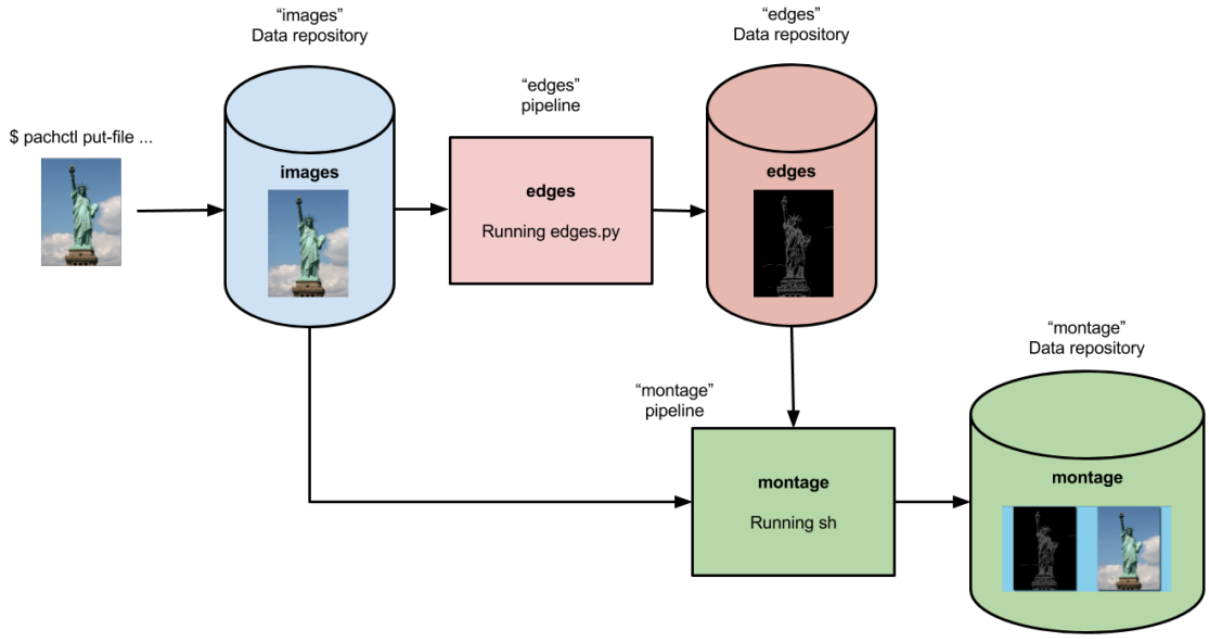


Figure 2.13 Modified Pipeline

Below is the pipeline spec for this new pipeline:

```
# montage.json
{
  "pipeline": {
    "name": "montage"
  },
  "input": {
    "cross": [ {
      "atom": {
        "glob": "/",
        "repo": "images"
      }
    },
    {
      "atom": {
        "glob": "/",
        "repo": "edges"
      }
    } ]
  },
  "transform": {
    "cmd": [ "sh" ],
    "image": "v4tech/imagemagick",
    "stdin": [ "montage -shadow -background SkyBlue -geometry 300x300+2+2 $(find /pfs␣
↪-type f | sort) /pfs/out/montage.png" ]
  }
}
```

This pipeline spec is like our edges pipeline except, for montage: (1) we are using a different Docker image that has imagemagick installed, (2) we are executing a sh command with stdin instead of a python script, and (3) we have multiple input data repositories. In this case we are combining our multiple input data repositories using a cross pattern. There are multiple interesting ways to combine data in Pachyderm, which are further discussed here and here. For the purposes of this example, suffice it to say that this cross pattern creates a single pairing of our input images with our edge detected images. We create this next pipeline as before, with pachctl:

```
$ pachctl create-pipeline -f https://raw.githubusercontent.com/pachyderm/pachyderm/
↪master/doc/examples/opencv/montage.json
```

This will automatically trigger a job that generates a montage for all the current HEAD commits of the input repos:

```
$ pachctl list-job
ID                                OUTPUT COMMIT                                STARTED
↪    DURATION           RESTART PROGRESS  DL       UL       STATE
92cecc40c3144fd5b4e07603bb24b104 montage/1af4657db2404fcfba1c6cee6c71ae16 45 seconds
↪ago 6 seconds          0       1 + 0 / 1 371.9KiB 1.284MiB success
81ae47a802f14038b95f8f248cddbed2 edges/146a5e398f3f40a09f5151559fd4a6cb   2 minutes
↪ago  Less than a second 0       1 + 2 / 3 102.4KiB 74.21KiB success
ce448c12d0dd4410b3a5ae0c0f07e1f9 edges/c5d7ded9ba214d9aa4aa2c044625198c   2 minutes
↪ago  Less than a second 0       1 + 1 / 2 78.7KiB  37.15KiB success
490a28be32de491e942372018cd42460 edges/bc2d20d0c23740f397622a62b0978c57   11 minutes
↪ago 35 seconds         0       1 + 0 / 1 57.27KiB 22.22KiB success
```

And you can view the generated montage image via:

```
# on OSX
$ pachctl get-file montage master montage.png | open -f -a /Applications/Preview.app

# on Linux
$ pachctl get-file montage master montage.png | display
```

HPE Confidential - circulation

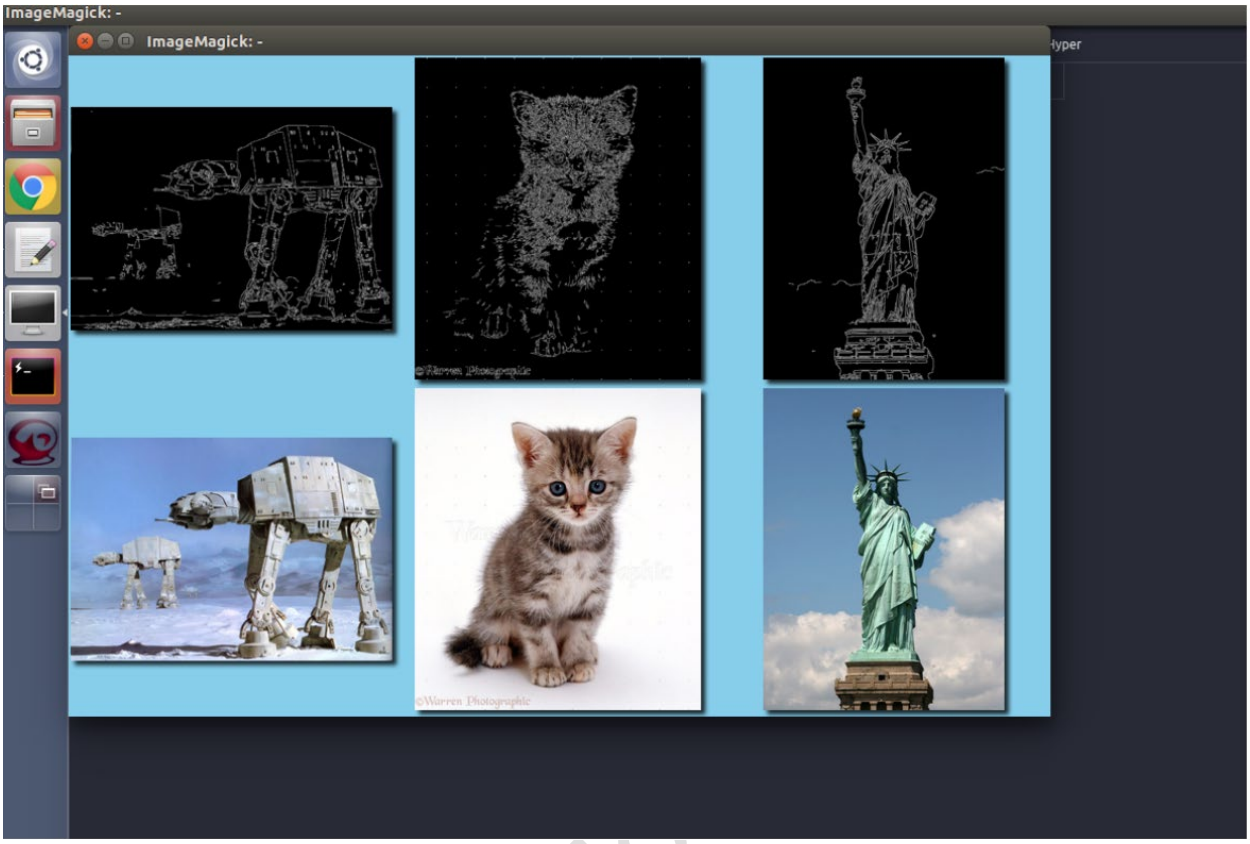


Figure 2.14 Modified Image

2.11 Seldon.CORE

With Seldon Core you can take your model and put it directly into the production using our flexible Model Servers. Using the so-called Reusable Model Servers, you can deploy your models into Kubernetes cluster in just a few steps:

1. Data Scientist prepares ML model using state of the art libraries (mlflow, dvc, xgboost, scikit-learn just to name a few).

2. Trained model is uploaded to the central repository (e.g., S3 storage).

3. Software Engineer prepares a Reusable Model Server using Seldon Core which is uploaded as Docker Image to the Image Registry.

4. Deployment manifest (Seldon Deployment CRD) is created and applied to the Kubernetes cluster.

5. Seldon Core Operator creates all required Kubernetes resources.

HPE Official - Not

6. Inference requests sent to the Seldon Deployment are passed to all internal models by the Service Orchestrator.

7. Metrics and tracing data can be collected by leveraging our integrations with third party frameworks.

2.12 E2E Serving with Model Servers with seldon

With Seldon Core you can take your model and put it directly into the production using our flexible Model Servers.

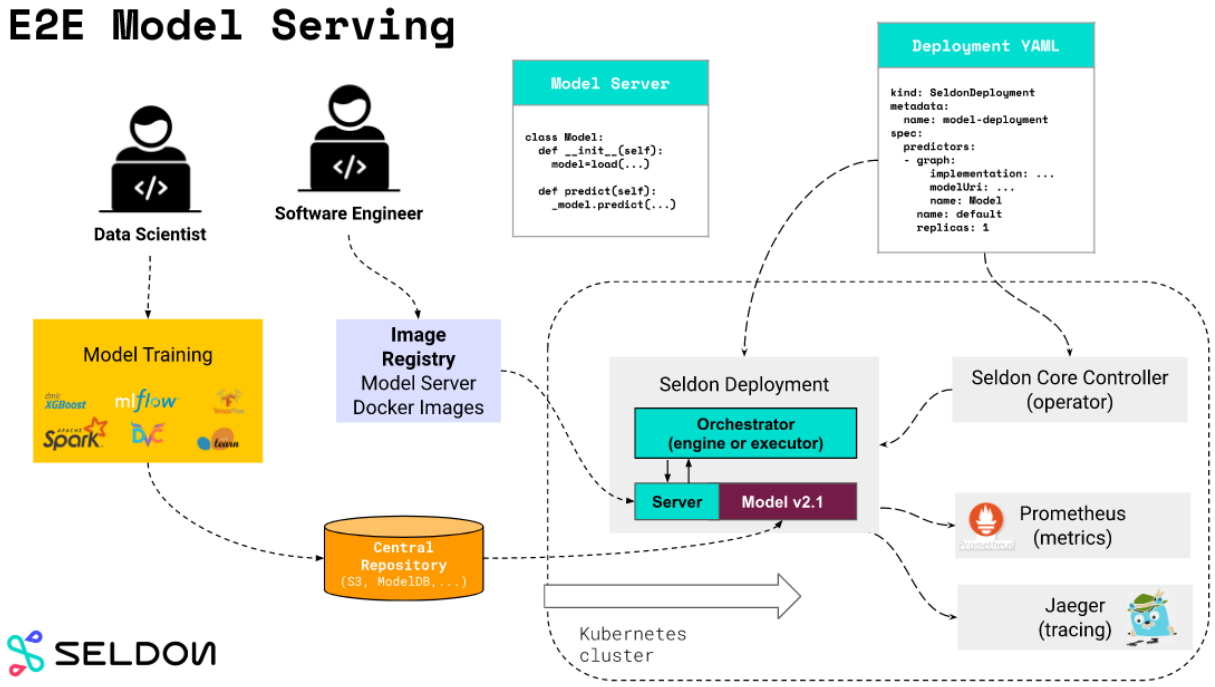


Figure 2.15 Seldon Architecture

2.13 Two Types of Model Servers

With Seldon Core you can build two types of servers: reusable and non-reusable ones. Each of these are useful depending on the context and the actual use case.

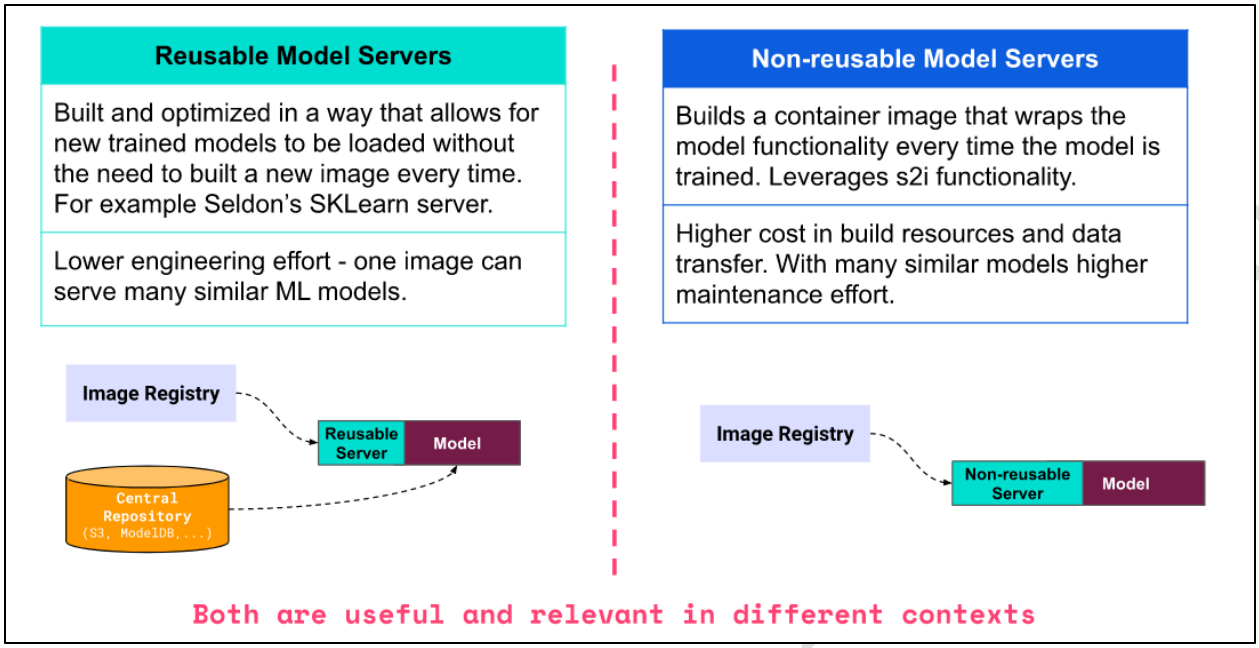


Figure 2.16 Model Servers

• Reusable Model Servers: Often referred to as prepackaged model servers. Allow to deploy a family of similar models without the need to build a new server each time. They often fetch models from a central repository (like your company's S3 storage)

• Non-Reusable Model Servers: Specialized server meant to serve a single model. Does not require the central repository but requires a build of a new image for every model.

2.14 Conceptual overview of machine learning deployments / inference graphs

A machine learning deployment (or inference graph) refers to a group of components in the Seldon ecosystem of a type associated with Seldon (Seldon Deployments). It represents a workflow, grouping the components of a machine learning system into a logical pipeline. The ML Deployment contains the configuration of the components and the definitions of the inputs and outputs of the system, and of each component.

Model

A component within a machine learning deployment which holds the representation of learned information from the training data.

HPE Official - Not for

Language Wrapper

A language wrapper is a model which enables cross language and/or runtime interoperability with a particular programming language. Language wrappers allows Seldon Core users to build Reusable and Non-Reusable model servers. As you will

see, the entire process is amazingly simple and requires user to only define logic that loads models and perform inference prediction as well as the required runtime dependencies.

Pre-packaged Inference Server

Conceptual overview of pre-packaged inference servers¶

A pre-packaged inference server is one of:

- SKLearn Server
- XGBoost Server
- Tensorflow Serving
- MLflow Server

Servers which can be used to deploy a trained model.

Pre-packaged inference servers come built into Seldon Core, to allow users to go easily from artifact (i.e., serialised model) to ML deployment regardless of toolkit. Please see the following docs pages for users looking for instructions on how to create their own “pre-packaged” inference servers:

Graph

A graph represents machine learning components as nodes with edges representing the inputs and outputs of an operation being passed from one component to the next.

HPE Official - Not for circulation

Request

A request represents a single call to a model for a prediction. The request will be a payload which holds prediction data (often in the form of an array) passed over a particular protocol. It is expected to follow a particular format.

Request Logging

Request logging is a feature of Seldon to be able to track the requests that have been submitted to a model. In the default setup requests are logged and stored in Elasticsearch.

Seldon Deployment CRD

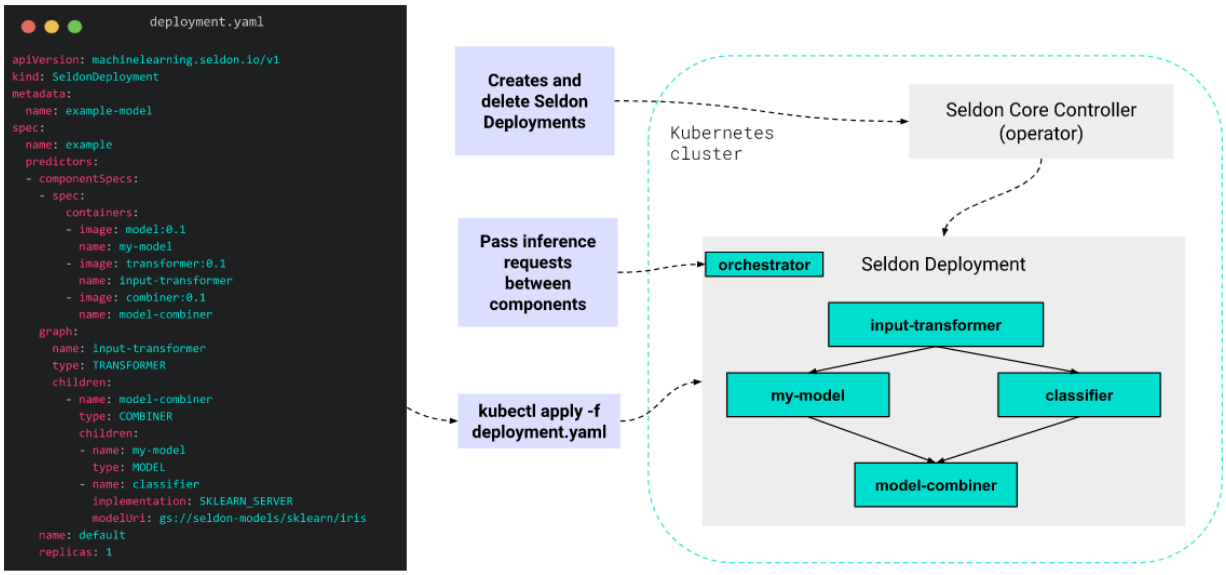


Figure 2.17 Seldon Deployment CRD

Seldon Deployment CRD (Custom Resource Definition) is the real strength of Seldon Core. It allows you to easily deploy your inference model to the Kubernetes cluster and handle some real production traffic!

HPE Confidential circulation

Custom Resources are extensions of the Kubernetes API. They allow one to create a custom combination of basic Kubernetes objects that acts together. In Seldon Core we use CRDs to define the inference graph through the manifest yaml files.

The manifest file that you write is enormously powerful yet simple. You can easily define what models do you want in your deployment and how they are connected in the inference graph.

2.15 SELDON PACHYDERM INTEGRATION

Pachyderm is a data layer with built-in version control and lineage for your data science applications. to put it another way, it integrates data version control with pipelines so that you can coordinate and monitor complex ml workflows. you can create a directed acyclic graph (DAG) that is automatically versioned by joining pipelines, allowing you to track the history of any run. insert new data; retrieve a model that has just been trained. The open-source Seldon Core machine learning engine's enterprise product, Seldon Deploy, is created to deploy, and monitor ML graphs at scale using a dashboard or API. Pachyderm, HPE Machine Learning Development Environment and Seldon Deploy can be integrated to implement an end-to-end ML pipeline.

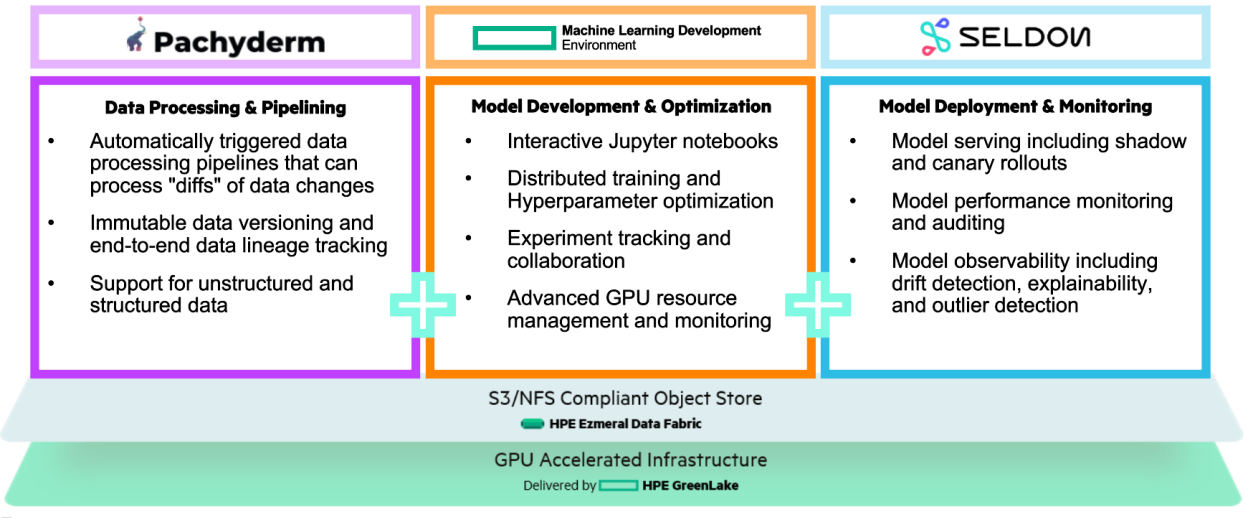


Figure 2.18 Seldon- HPEMLDE- Pachyderm Integration

HPE Confidential - Not for circulation

MODULE 3

AI APPLICATIONS – COMPUTER VISION AND ACCELERATING THE HPC SIMULATIONS

HPE Official - Not for circulation

3.1 Introduction

Artificial intelligence is the field of study that deals with mimicking human intelligence into machines. With the help of artificial intelligence, machines should be able to communicate with humans either through the text that is typed on the computer terminal, understand what is spoken, or generate speech (Natural Language). It should be able to achieve goals by executing a sequence of actions(planning) and should have the ability to see using cameras(vision) like human eyes. Machines should be able to conclude (inference) by remembering complicated facts. Machines should have the ability to analyze various situations based on complicated rules and offer us advice. (Expert systems).

With reference to the Figure 1.1, we can identify that AI should possess human like capabilities in learning from experiences, reasoning from what is learned, recognize images, solve complex problems based on reasoning, understand natural language, and create perspectives based on different contexts. Sometimes, decisions taken based on contextual information and rational thinking are beneficial than any logical decision taken as per predefined algorithms. Humans can take rational decisions which is expected in AI and efforts are taken to fulfill the same. It has been successful to a great extend so far.

Based on the current scenario of implementation of AI on various platforms, there are 6 diverse types of capabilities which are made use of. They are

- Personalization and profiling
- Predictions
- Pattern recognition and anomaly detection
- Natural language Processing
- Object identification
- Goal achievement

Some of the systems which use the capabilities of AI, make use of combinations of above capabilities to perform in a successful manner. Let us investigate each technique one by one.

HPE Official - Not for circulation

3.2 Personalization and profiling

In this method, each person is treated as a unique individual, based on his interests, a profile is created, and that is used for giving suggestions on various applications. We can see such personalized suggestions in platforms like Amazon shopping, Netflix, Spotify, Youtube etc. The profile of the person gets updated each time his interest changes. Based on his current preferences predictions are made on his likes and dislikes. These kinds of profiling are used to check if a loan is to be sanctioned for a person or not. It is also useful in preparing a database of criminals and terrorists and can be used in monitoring and tracking them.

3.3 Prediction

The aim of a prediction system is to interpret that there is a good chance of this event happening given the data under consideration. This kind of a probability prediction helps humans to take good decisions in various situations. Such predictions are used in marketing to understand customer behavior and sell products. It is used in medical field to predict the probabilities of certain diseases and act accordingly. It is also used to identify the kind of impact a certain pandemic can make in the upcoming future and this data can be used to take preventive measure and equip the society with required medical facilities.

3.4 Pattern recognition and anomaly detection

Here, from the machine can understand patterns from the data and can give suggestions. This is the method used to give us suggestions on what to type next while typing a mail or helps a person to complete a task with more suggestions on what he is looking for in a search engine or any other application. Anomaly detection is also possible from the given data. For example, from a set of images of brain, the system can recognize one with tumor and categorize it. Image analysis based on the patterns can be carried out for this purpose.

HPE Official - Not for circulation

3.5 Goal achievement

In this scenario, machines try to learn from the experiences. It calculates rewards and penalties of certain moves from the experiential learning. This approach is being used in playing games with systems. It tackles our movements and takes decisions accordingly.

3.6 Natural Language processing

This is a technique by which machines can communicate with humans in the language of humans. This method even enables the device to analyze human emotions and act accordingly. AI should be capable of recognizing human speech and text and respond based on the requirements. Speech analysis, Language understanding, Text analysis, Speech to Text conversion and vice versa are some of the skills AI must possess to process natural language in various contexts. Siri, Google Assistant etc. are examples which uses this technology to communicate with human and take decisions accordingly.

3.7 Object detection

With the help of machine learning algorithms which are trained on identifying certain objects, these systems can be used in identifying suspicious objects. It can be used to identify faces using face recognition algorithms. It can also be used to identify persons with criminal background or any kind of objects which can be a potential threat to any situation in hand.

Like humans interact, the AI entity should be able to interact with humans. Humans should not be able to recognize that they are interacting with an AI entity. We need to carry out the following tests/approaches to AI entity to achieve this

The AI entity can be corelated to human-likeness based on following tests

- Turing Test
- The Rational Agent Approach
- The Cognitive Modelling approaches
- The Law of Thought Approach

HPE Official - Not for circulation

As humans interact, the AI entity should be able to interact with humans. Humans should not be able to recognize that they are interacting with an AI entity.AI entity should have the following capabilities to achieve this.

3.7.1Turing Test

- To communicate successfully Natural Language Processing should be used.
- Memory enabled by knowledge representations.
- Automated Reasoning to answer questions and draw conclusions.
- Should detect patterns and adapt to new circumstances using Machine Learning.

3.7.2 Cognitive Modelling Approach

Human cognition skills must be built to Artificial Intelligence Model.

There are three approaches:

- Introspection: Model should be built based on observing human thoughts.
- Psychological Experiments: Observing the behavior of humans and model it.
- Brain Imaging: Observe the brain functions using MRI and replicate it using code.

3.7.3The Laws of Thought Approach

This approach takes into consideration human thought process. There are many situations where humans take decisions without being 100% sure, based on context. The thought process will vary depending on the situations. It is highly impossible to code this contextual situation as solving a problem in practice and in actual situations will be different as there would be many parameters. The entity must behave according to some logical statements.

3.7.4 The Rational Agent Approach

Many times, there will not be logical right thing to do, would depend upon the circumstances. The rational agent approach tries to make the best choice depending upon the circumstances. The objective is to design an agent which is adaptable to the dynamic environments.

HPE Official Not for circulation

After analysing various broad capabilities of AI, and the tests conducted to measure it, let us investigate some of the specific capabilities that are made use of, in various AI related applications in detail. Most of the data generated consists of images. Processing of these images are required in various applications like number plate identification, medical image analysis or any image related applications. The following section describes the basics of image, how images are being analysed, how texts are extracted from documents, techniques used for Image classification, object detection and form recognition.

3.8 Image Analysis

In real world, an image can be considered as a function of two real variables, b (x, y) where b is the brightness at the coordinate position (x, y). Region-of-interest or Regions are the sub-images of an image. An analog image b (x, y) in a 2D space is converted into a digital image b [m, n] through a sampling process which is frequently known as digitization. An image consists of N rows and M columns as shown in Figure 1.2. Pixel is an intersection of rows and columns. In digital images, number of rows is known as vertical resolution. Horizontal resolution is the number of columns. Pixels are the basic element of any image, also known as picture elements.

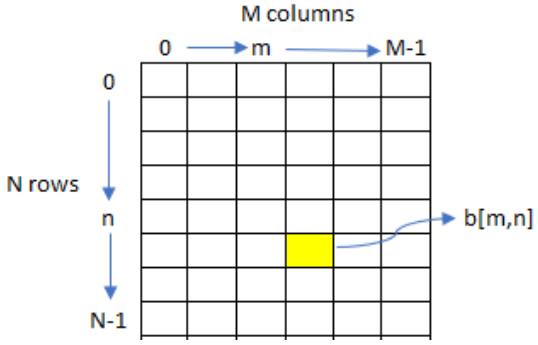


Figure 3.1 Image representation as pixels

The average brightness rounded to nearest integer value is the value assigned to every pixel. Pixel values are binary words of length *k* and can represent up to 2^k different values. The value *k* is known as bit depth or depth of the image. Commonly assigned values to digital image are shown in the Table 1.1.

HPE Offi - Not for circulation

Table 3.1: Common digital image parameters value

Parameter	Typical values	Symbol
Columns	256,512,768,1024,1920	M
Rows	256,512,525,625,1024,1080	N
Gray Levels	2,64,256,1024,4096,16384	L

The several types of images are binary, grayscale and Color images.

3.8.1 Binary Images

Binary images pixel values can take only two values black or white as shown in Figure. These are encoded using a single bit (0/1) per pixel. Binary images are usually used to represent in electronic printing, fax transmissions encoding, line graphics and archiving documents.

Figure 3.2: Binary image

3.8.2 Grayscale Images (Intensity Images)

The brightness, intensity or depth of the image is represented in a grayscale image. The image data are positive whole numbers in the range 0……2^k -1 and it cannot be a negative value as it represents intensity or density of light energy. For example, a grayscale image k may be equal to 8 bits (1 byte) per pixel and intensity can fall in the range of 0, ……..,255

HPE Official Not for circulation

where 0 represents minimum brightness(black) and maximum brightness is given by value 255 which represents white.

Figure 3.3: Grayscale Image

3.8.3 Color Images

Color images are based on primary colors red, green, and blue (RGB). The range is around [0,255] and makes use of 8 bits for each color component. Each pixel requires 3 x 8 =24 bits to encode all three components. By superimposing three colors, red, green, and blue human eyes can be “fooled” into seeing most of the colors which can be recognized perceptually. The intensity or level of these three colors determines what we perceive.

HPE Official - Not for circulation

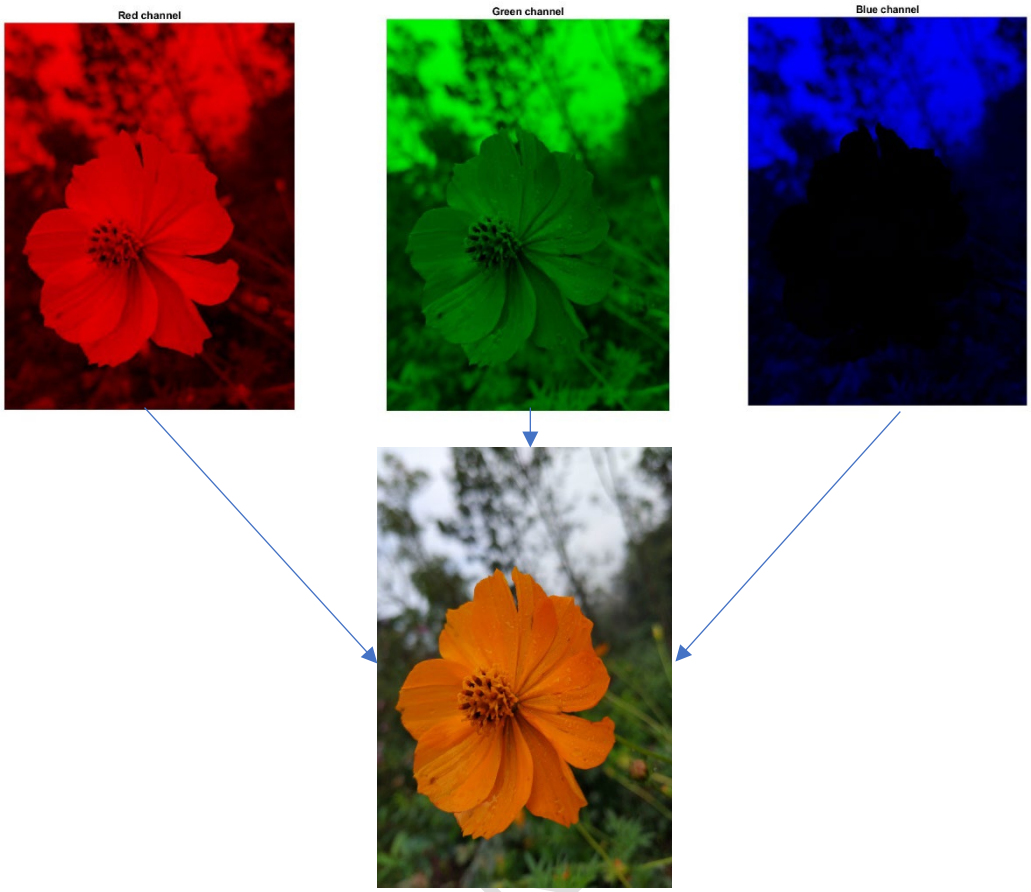


Figure 3.4 : RGB channel images and color image

If the image is made up of a grayscale image of any of the primary colors, it is known as a channel as shown in figure 1.5. The combination of RGB channels will result in the original image. The primary colors can be combined to form different colors as shown in Table 1.2.

Table 3.3. Colors encoded as RGB triples

(1,0,0) = red	(0,1,0) = green	(1,1,0) = yellow	(0,0,1) = blue
(0,0,0) = black	(0,1,1) = cyan	(1,1,1) = white	(0.5,0.5,0.5) = grey

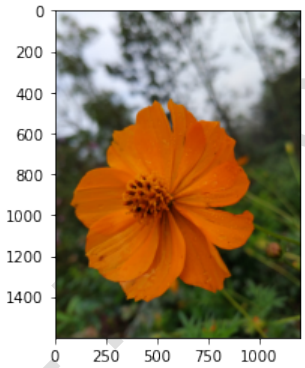
If the image is of dimension 240x400 and each pixel is of 8 bits, then the size of the image will be

240x400x8 =768000 bits. As 1 bit = 8 bytes, this value can be converted to bytes by 768000/8 =96000 bytes.

An image can be read and displayed using the following commands

```
#import matplotlib modules
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

#Read Images
img = mpimg.imread('PHOTO-2021-12-01-21-08-48.jpg')

#Display image
plt.imshow(img)
```

Output:

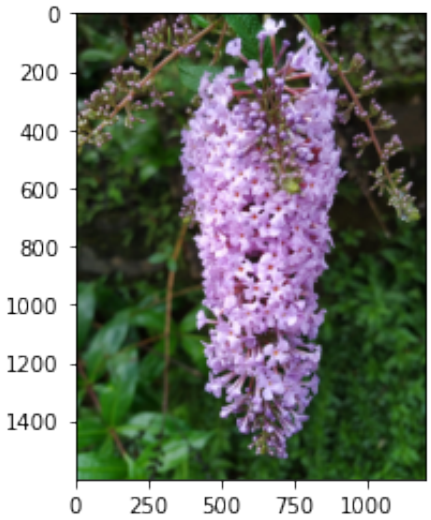


An image can be written to a folder by following commands and dimension of the image can be displayed.

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np

#Open image and convert image into array
im = np.array(Image.open('flower2.jpg'))

#Save the file as filename.jpg
plt.imsave('filename1.jpeg',im)

#Display the image written to filename.jpg
plt.imshow(im)

#Display the dimensions of image
print(im.shape)
```

Output:

(1600, 1200, 3)



3.8.4 Histograms

The statistics of an image can be visualized and interpreted with the help of histograms. Histograms describe the intensity values frequency in an image or are the frequency distributions. For a typical 8-bit grayscale image I in range I (m, n) Ɛ [0, K-1], the histogram holds K entries where K =2^8 =256.

Histogram entry can be defined as

h(i) = For intensity i, the number of pixels in I for all 0 ≤ I < K.

Thus h (1) denotes the number of pixels with the value 1 and h (2) the number of pixels with value 2 and so on. H (255) is the maximum intensity value i.e number of white pixels i.e., 255 = K -1 as shown in figure below.

HPE Official - Not for

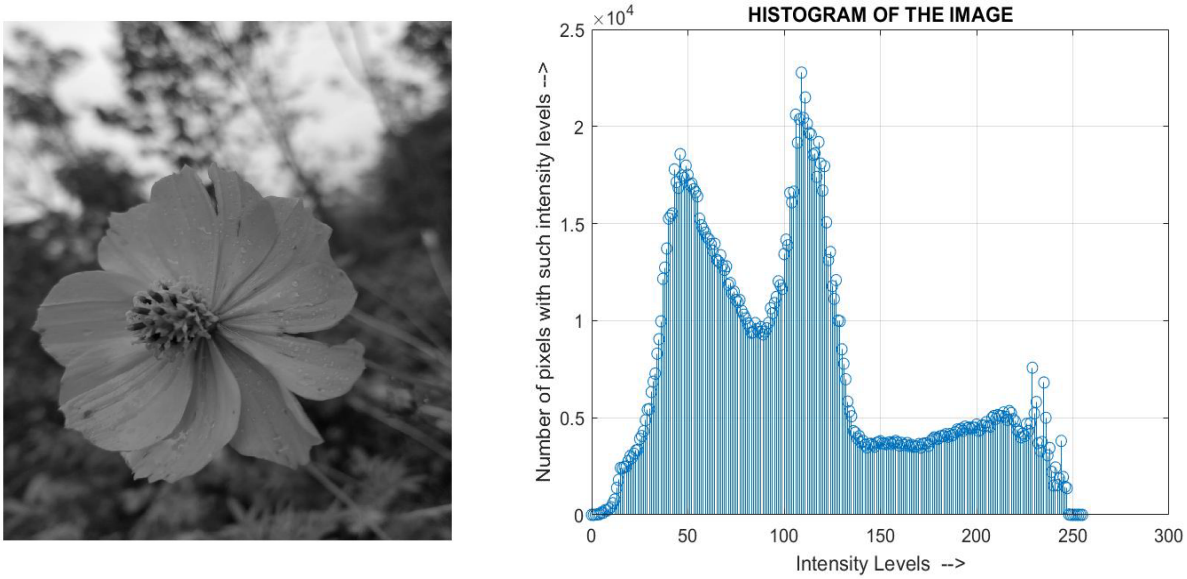


Figure 3.5: Histogram of a grayscale image

Another popular technique in image processing is image inversion. Image inversion or color inversion is achieved by subtracting maximum intensity from the given RGB color value. The resultant formula would be $img\_inv = I_{max} - I(m, n)$

where img_inv is the pixel that is inverted, I_{max} is the intensity maximum value and I (m, n) is the coordinate intensity values. The color inverted image for the original image is as shown in figure

HPE Official - Not fo

Figure 3.6. Image Inversion

The several topics discussed above give you an overview of the processing techniques for images. The Human Visual system can recognize images and videos. Image Recognition or (Computer Vision) is the ability of computers to recognize images and videos and extract valuable information. Image Analysis is an application of computer vision. The main objective is to make computers understand images and videos. Image Processing Techniques are used to achieve this. Images could be of distinct types like color and grayscale images, multispectral and hyperspectral images, or a video sequence.

With advancements in technology, the amount of data that needs to be processed is increasing day by day. Consider social media monitoring where the detail of a particular brand must be analyzed. It is impossible for humans to search for images of the brand. With the help of image analysis, all details about the image will be available within a few seconds. Image Analysis has seen significant growth with the advances in software frameworks and hardware platforms. Image Analysis could be used to find the internal structure (constituents of the product) and external characteristics (size, color, shape, and texture) of food products. Image Analysis is quite popular in all areas of science and technology. The main objective of image analysis is to interpret real images and to identify and reconstruct properties like texture characteristics, scene illumination, color

HPE Official - Not for circulation

characteristics and shape information. In image analysis, a caption will be generated about the details of the image.

Computer vision system is like human visual system as shown in figure 1.2.

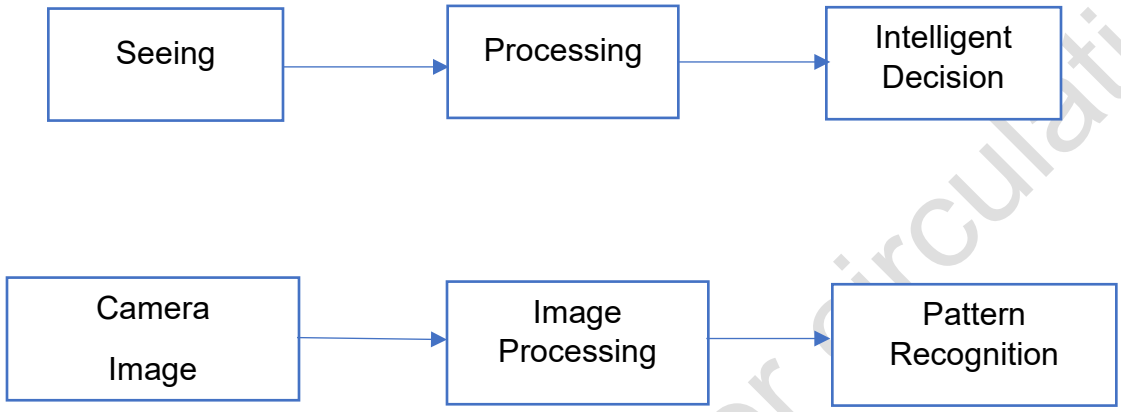


Figure 3.7 Human Visual system Vs Computer vision

In both the systems photons are converted to a signal by light sensors, the signals are processed, and the signals are interpreted (object recognition).

In image analysis, the goal is not to alter or enhance the appearance of an image, but to extract meaningful information from its contents like finding bar code in a milk carton, or object from its background.

3.9 Text Extraction from document

Text extraction from images and documents play a vital role in information retrieval in present world as vast number of resources are available in the digital medium. Much information from journals, records, scanned documents, web pages, tv programs, currencies, business cards etc. can be obtained if they are retrieved, filtering out the unwanted areas. Usually, such documents are converted to images and various techniques are used to obtain information from them. It can be used in applications like document analysis, identifying number plates of vehicles, video analysis, detecting objects, content retrieval etc. The text available in any scanned document are available in

HPE Official - Not for circulation

various fonts, alignment, shapes, colors etc. Sometimes the background color may be a complex color form which extracting text may be a complicated process. There are several types of texts present in documents. Images in documents are categorized into document image, scene image and caption image. Usually, captions are imposed upon videos and images to narrate the summary on the content or to highlight what is being spoken on the video. This information if extracted can give a lot of insights about the content. Scene texts are those texts which are present in the scene itself when the video or image is being recorded/created. A camera captures it and there will be a lot of information available on them which can be extracted. Extracting scene text could be difficult as it may appear in all complex manner with different lighting, backgrounds, orientation etc.

Images and pdf documents consist of lots of texts. Many times, this documents clarity would be very less. Optical character Recognition (OCR) is the technique used to extract text from documents. OCR may be generated accurate results if the document is blurry or unstructured. OCR software considers various attributes like text density, text structure, Font, and Artifacts. OCR has its own limitations. So current technologies use OCR based on artificial intelligence. As it incorporates AI, the technology can learn and analyze when extracting from huge databases of documents and nurturing the ability to think on its own.

Let us now investigate the methods used to extract texts from documents like images for analysis. Many times, informative texts are present in images, cards, pdfs etc. To detect them from any type of document for performing a deep analysis requires some tactics. There are five types of methods used for text extraction. They are

HPE Official - Not for circulation

3.9.1Region-based method

- Flower
- Detection

-
- Feature Extraction
-

Figure 3.8. Example of region-based text extraction

In this method a sliding window is used to detect text from any kind of image. Numerous factors like contour, edge, shape, and color are used in this approach. The property of uniformity in color or grey scale in the areas of text in an image compared to its background is utilized for its detection. Giving proper thresholds to the images can help in distinguishing the areas with text from its background. Thresholding of the image can be performed by using an intensity level in between the color of the text and that of the background. Once the thresholding is performed, the regions of texts are identified, and the subsections are merged, and they are differentiated with the help of bounding boxes.

Usually, this technique works based on three levels. In the first level, segmentation of the document / image into small parts is carried out to differentiate textual characters from its background. In level 2, merging and grouping of small areas is performed to form the words and sentences. In the last level, differentiation of text with respect to other elements is performed.

HPE Official - Not

The Region-based method can be divided into two categories as shown below.

a) **Connected component based (CC- based)**: Small sections of images are grouped to form large components. All the regions of images are identified this way by continuously performing the grouping. The area without texts is filtered by using spatial coordinates of the sections and identifying the boundaries of the regions with texts.

b) **Edge Based Method:** In edge-based method, the edges of every letter and digit are detected. To develop a high-level contrast between background and text, this method is used. The contrast between the objects and background in images is used to identify the boundaries or edges. Using a filter for the edges, the background is eliminated for carrying out further processing.

3.9.2 Texture-Based method

Texture and its properties are used to extract the text. Here, various parameters of an image or a document which define its texture is used to distinguish the text. Correlation, contrast, entropy etc. are some parameters which define the texture. Based on the texture properties a classifier can be trained to divide the regions. Here machine learning technique is being used for classification. This method is more generalized compared to the earlier methods, but it is computationally expensive.

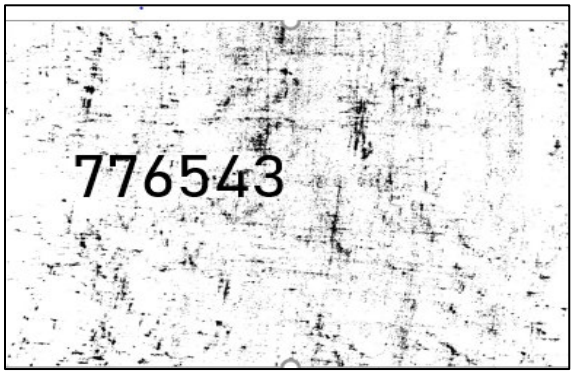


Figure 3.9. Example of texture-based text extraction

Official - Not for circulation

3.9.3 Hybrid Technique

Hybrid technique uses a combination of above two techniques. Text is detected using Region based method and features are extracted using texture-based method. It enjoys the advantages of both the methods. Region based method helps in identifying the presence of texts whereas the texture-based method extracts the features of the text, which in turn can be used in processing the language for obtaining deeper insights.

3.9.4 Morphological Method

The text related features are extracted from processed images using morphological methos. This method removes background noise first and then analyses the segmented part based on size, shape etc. It positions a particular structure at certain locations of the image, and it is compared with the nearby regions. This method is well suited for textures, as it is good at finding spatial relations among various pixels. It possesses numerous tools to extract the parameters related to the structure. The tools can be expressed in Boolean algebra and hence can be used on dedicated hardware. Morphological/ structural features of texts of a particular language can also be used to differentiate the background of the document from the text area.

Figure3.10. Example of Morphological method for text extraction

The statistical aspects of various words and usages of a particular languages can be used for comparing with the structure of other images or backgrounds present in a particular document. The special structures and features of certain words in a language can also be used for analyzing sentiments in a text as well.

Text extraction also known as keyword extraction, which utilizes machine learning can automatically scan text and extract relevant keywords and phrases from various documents like news articles, receipts, or any unstructured documents. The extracted data is converted to a file of suitable format which can be used by the end user.

Analyzing and interpreting texts from the users play a key role in various business development, tracking of illegal intentions, detecting offense etc. Of late, there has been a lot of improvement in this area as the enormous amount of data generated and Natural Language Processing came into picture. Before getting into the essence of this advanced technology, let us understand the basics of text extraction. The document available from various devices and medias may not be structured most of the times. Various data available on documents are text, images, handwritten scanned notes etc. In such scenario, there is a need of converting those unstructured documents into proper formats for conducting different analysis.

The basic steps involved in preparation of documents to proper formats before deep analysis are

- Language Identification
- Tokenization
- Sentence Breaking
- Part of Speech Tagging
- Chunking
- Syntax Parsing
- Sentence Chaining

(i)Language Identification

Primarily, step in preparing the document for processing is to identify the language in which the document is written. Each language has its own idiosyncrasies, which makes it an important task to understand the language of the text.

(ii) Tokenization

HPE Official - Not for circulation

Tokens are words, numbers, punctuations, or sentences, which help in providing valuable information regarding the text. They are the individual meaningful units obtained from the document. Tokenization is the process of breaking text documents into different tokens. Tokens are often specific to languages. It is easy to tokenize languages like English, which use alphabets. They have specific ways to break up words, sentences etc. by using space, punctuations etc. So, rule-based algorithms are enough to tokenize such languages. But certain languages like Chinese, Japanese languages use logos and such languages do not have space breaks between the words itself. So complex machine learning algorithms are required to tokenize the texts of those languages.

(iii) Sentence breaking

The next step in the pipeline for preparing the text is to perform sentence breaking. After tokenization, we need to identify the boundaries of texts. For alphabet-based languages, it is easy in contexts where the presence of period sign indicates end of sentences. But sometimes for indicating abbreviations also there can be a period symbol (example, Mr. represents Mister). Similarly, to break sentences in short texts communication in social medias, the technique used are different.

(iv) Part of speech tagging

After breaking down the sentences, the next step used is the process of tagging it. This method is used to identify the parts of sentence if it is verb, noun, or proper noun etc. Because of the wide varieties of ways in which the languages are used, this process is not straight forward often. The system must be trained to understand various usages of each language to perform the right tagging.

(v) Chunking

It is the process of identifying several types of phrases in sentences, if it belongs to noun phrase, verb phrase etc. Hence it assigns the PoS tags to phrases.

(vi) Syntax parsing

HPE official - Not for circulation

It is a method of identifying the structure of text, which is especially important in identifying the sentiment involved in sentences. It is overly complex in terms of computational aspect. This method can understand if the context of the text is positive or negative. It can be tackled with the help of machine learning algorithms.

(vii) Sentence chaining

Sentence chaining is the process of connecting similar sentences. It can link individual sentences to a common topic and context irrespective of the positioning of sentences in a document. This is the last step before doing a deeper analysis on the document text.

Text extraction plays a key role in social media monitoring, business intelligence applications like competition analysis and market research.

3.10 Image classification and object detection

3.10.1 Image Classification

Image classification and object detection techniques help the machine to understand and identify the objects in a digital image. Consider an image of a dog as shown in Figure 1.3. Humans can recognize it has dog just by seeing the image. The technique used to classify or predict the class of an object in an image is known as image classification. The features of an image can be accurately identified.

The main steps of image classification include Image preprocessing, Feature extraction and training and classification of the object.

1)**Loading and Image Preprocessing**: -The dataset which is a folder which contains all the images must be imported. Image preprocessing is used to improvise the quality of an image by pixel calibration and standardization, reducing noise which helps in better analysis. Image preprocessing consists of processes like image resize, reshape, data augmentation techniques to improvise the image and obtain better classification accuracy.

HPE Official - Not for circulation

2)**Feature Extraction and Training**: -Depending upon the type of image, its unique features must be identified and model must be trained to classify accurately.

3)**Classification of the object:** -In this step the image patterns are compared with the target patterns and using suitable classification technique the detected objects are classified into predefined classes.

4)**Estimating the performance of the model**: -The models performance can be analyzed by calculating various metrics like accuracy, precision, recall.

Image classification can be binary or multiclass. If the classification consists of only either 'yes' or 'no,' its binary classification. If it consists of various labels, it is called multiclass classification.

The numerous examples for image classification

i)Classify from an X ray image whether a patient has covid or not. (Binary)

ii)Classify images of fruits like mango, apple, and pear (multiclass).

iii)Classify several types of vehicles in a street (multiclass).

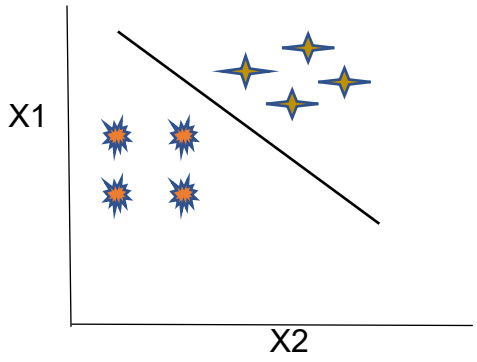


Figure 3.11 Binary Classification

Multiclass Classification

Multiclass classification is the technique where the classification task consists of more than two classes. Given a dataset of 'n' training examples with each having features and

Not for circulation

a label. Each training example can have ‘p’ features. Each label corelate to a class. In case of identification of fruits, shape, color, and size are the features and Mango, Apple and Pear are the different class labels.

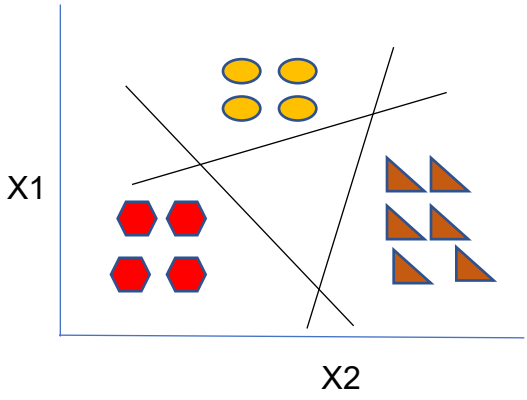


Figure 3.12 Multiclass classification

There are several types of algorithms in machine learning that can classify images. They can be categorized into unsupervised and supervised algorithms.

Unsupervised Classification

As we know, in unsupervised method, there is no training data available. So here, the algorithms cluster the unlabeled data into diverse groups by identifying the patterns or common characteristics from the images. The two most popular algorithms used for the image classification are K-means and ISODATA.

K-Means: It is an algorithm that groups the images/data into K groups based on their patterns. The data points are clustered based the proximity of the value to the average or mean value fixed for a group. This iterative process continues until all the data which are close to the mean of a group falls into the respective group.

HPE Official - N

ISODATA: It stands for “Iterative Self-Organizing Data Analysis Technique. It uses Euclidean distance as a measure to cluster image into classes.

Supervised Classification

In this method, reference samples or training data of images are given to train the classifier. Based on the learning obtained from this stage, it classifies the unknown images. It uses algorithms such as Support Vector Machine, K-nearest Neighbors, Deep learning algorithm such as Convolutional Neural Network etc.

Support Vector Machine: SVM tries to draw a decision boundary among different classes by using the learning obtained during the training phase. This decision boundary is a hyperplane and support vectors are the data points which are closest to the hyperplane.

K-Nearest Neighbors (KNN): Using KNN, the images can be segmented or classified based on the Euclidean distance of the testing data with that of the training data. The data points which are closest neighbors are classified into one group.

Decision Trees (DT): Decision Tree based algorithms can also be used to recognize patterns from the images and perform classification. If we use combination of machine learning techniques are used to create a model, that approach is called Ensemble technique. Ensembles of decisions trees are also used to get better classification accuracy. There are two types of ensembles called bagging and boosting methods. Decision Trees being a weak learner, bagging technique combines the decisions of multiple decision trees on the training samples to arrive at the decision on classification. In Boosting techniques, the difference is that each tree learns from the mistakes of the previous one. So, execution takes place in sequential manner. The error from the previous step is forwarded to the next step and this process keeps repeating till the final stage is reached. So, the decision tree at the last stage is assumed to be learning the most as the tree at each stage improves upon the learning as it learns from the mistakes of the previous stages. Random Forest (RF) is an example for Bagging technique while AdaBoost and Gradient Boosting algorithms are examples for Boosting methods.

HPE Official - Not for circulation

CNN: The Convolutional Neural Network models are the most famous algorithm used in the image data domain. It has the capability to work extremely well on computer vision tasks like object detection, classification of images and image recognition. CNN requires little pre-processing and works best for image data. CNNs have input layer, output layer, and hidden layers. The hidden layers are made up of convolutional layers, ReLU layers, pooling layers, and fully connected layers. Convolutional layers apply a convolution operation to the input. This passes the information on to the RELU layer which works on RELU activation function. Pooling layer merges the outputs of clusters of neurons into a single neuron in the next layer. Fully connected layers connect every neuron in one layer to every neuron in the next layer.

Let us now see how object detection works

3.10.2Object detection

Suppose we have multiple objects in an image. An image usually contains some part which does not carry any useful information. So, a technique called image segmentation is used first to filter out the unwanted areas and focus on the critical areas containing the objects. Based on the masks created by the image segmentation process, we can identify the shapes of objects present in an image. Once these segments are finalized, we can identify and classify the objects using the technique known as Object detection. Object detection is the technique where the location of object is marked by a bounding box and the class of the located object in an image is identified. The model predicts the object's location and labels the object. Object detection helps us to understand and analyze scenes in an image.

Object detections consist of three frameworks:

- Generate region of interest or region proposals using suitable model or algorithm. Large set of bounding boxes covering the entire image is the region proposal.
- For each of the bounding box, visual features are extracted and evaluated to determine the objects present in the image.
- Finally, overlapping boxes are conjoined into a single bounding box.

HPE Official - Not for circulation

The most popular applications of object detection are in crowd counting, self-driving cars, video surveillance, Face detection and Anomaly detection.

The most famous architectures include R-CNN, Mask-RCNN, Fast-RCNN, Faster-RCNN and Yolo.

R-CNN

R-CNN takes the image and identifies the objects in the picture. The input to the system is image and the outputs are bounding boxes and the corresponding labels for each object in the images.

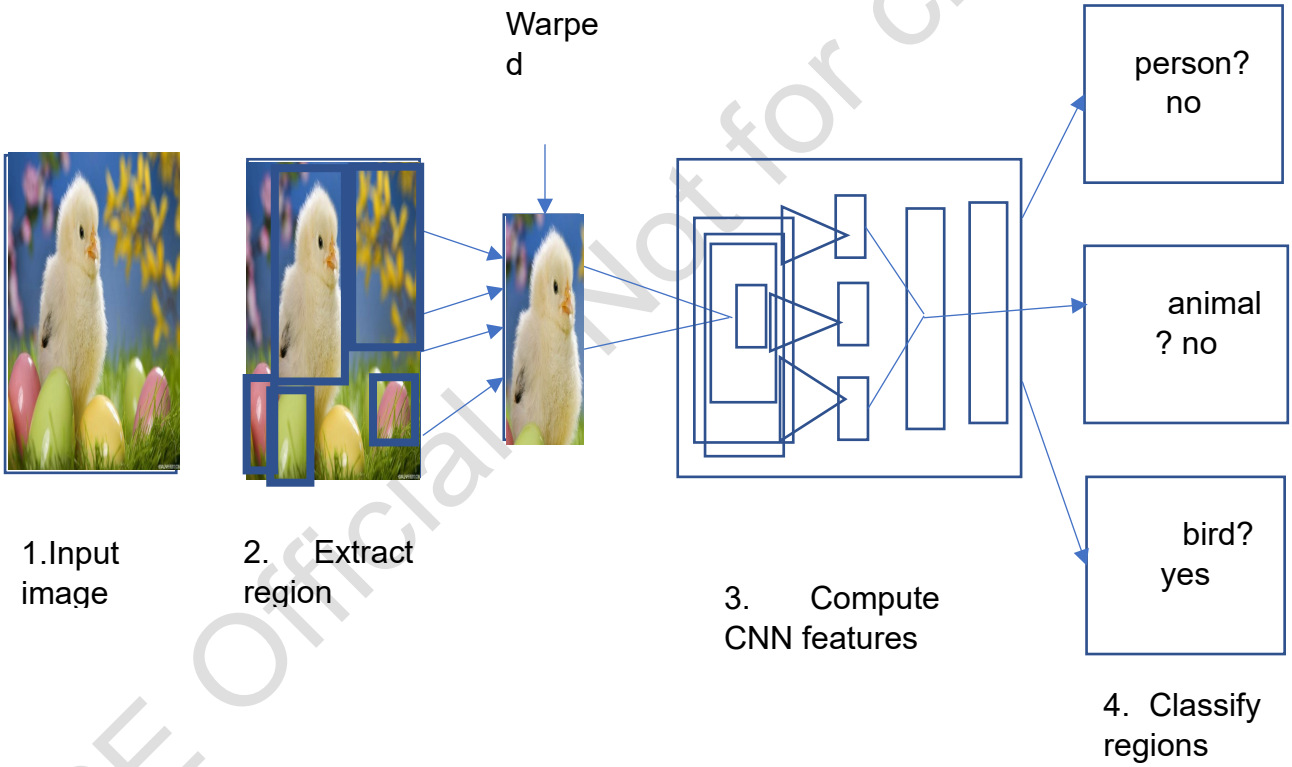


Figure 3.13 R-CNN Architecture

As per the figure above, we can see the modules. The first module inputs the image. In the following step category independent proposals are generated. Here all the image categories are detected. R-CNN performs selective search for creating the bounding boxes for object detection. Windows of many sizes are looked upon and pixels are grouped. Next, deep CNN step is introduced, which extracts features from each region

HPE Official - Not for circulation

which are identified by the previous step. The output of CNN is given to linear SVM classifier. It can classify the images based on the features identified. During the test time, R-CNN takes 47s/image.

Mask R-CNN

In this method, a segmentation mask of the regions is also included on top of the Faster-R-CNN architecture. So, segmentation of objects is performed using these masks. The training time of Mask R-CNN is extremely high.

Fast -RCNN

In R-CNN, all stages of the network demand separate training, it consumes more time, requires more resources and computation. Training CNN on various objects and making SVM to classify the features of each region proposal, make R-CNN less efficient. It also demands more testing time. Because of these shortcomings, Fast R-CNN was proposed. The fast R-CNN has CNN as its backbone usually pre trained on ImageNet data. But the final pooling layer of CNN is replaced by an ROI pooling layer. Final Fully Connected layer of the CNN is being replaced by two different branches, which are of SoftMax branch and bounding box regression branch as shown below.

HPE Official - Not for circulation

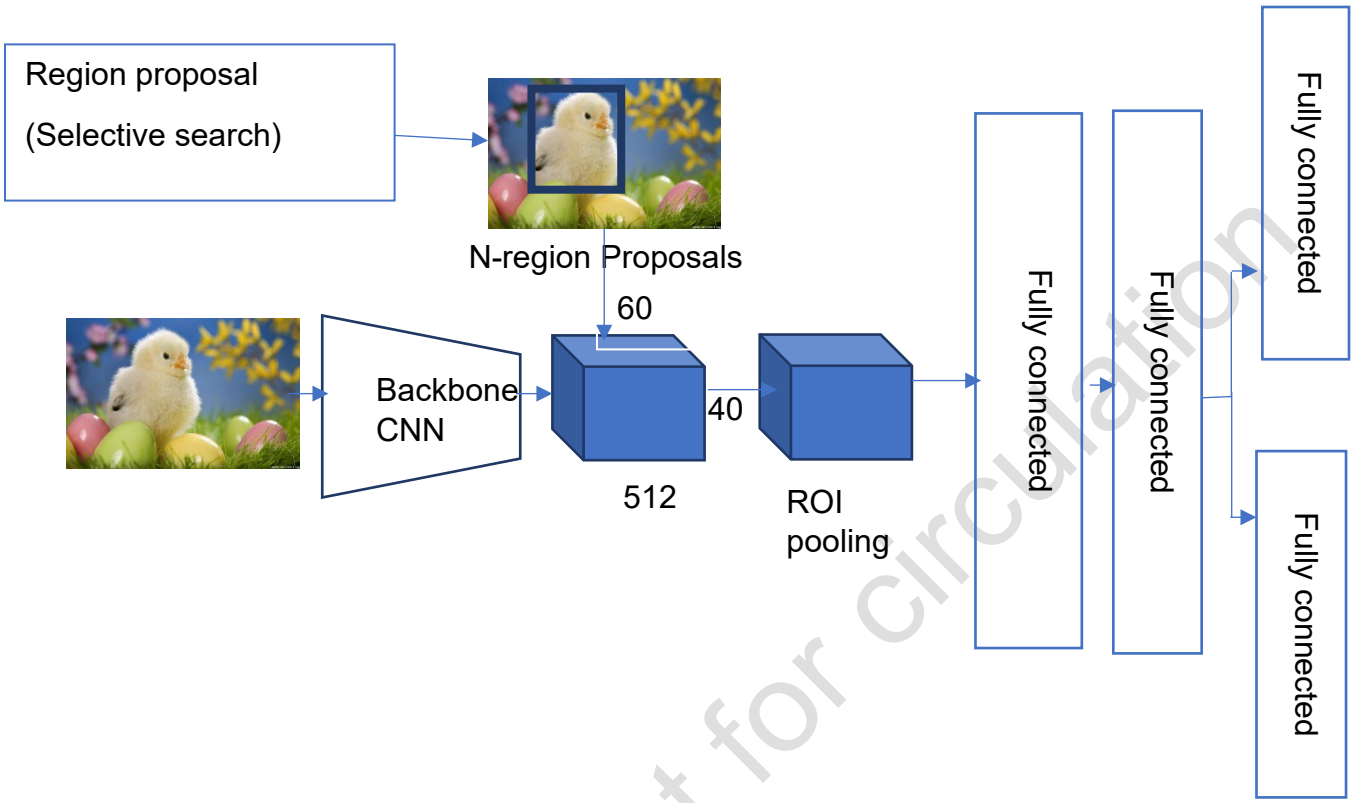


Figure 3.14 Fast R-CNN Architecture

The image is input to the backbone CNN and the feature from its last sage is obtained. At the same time proposal windows are obtained using selective search technique. These rectangular windows indicate the presence of object. The obtained features belonging to the windows are fed to the ROI pooling layer in the next stage. The pooling layer divides the windows with the features into sub windows. Then it performs pooling on this sub windows to give a standard size feature set. The output size of this layer will be in line with the size required by the fully connected layer present in the next stage. The output of the FC layers is then passed onto the SoftMax and bounding box (BB) regression modules. SoftMax branch will produce probability values for the Regions of Interest while bounding box branch will make the bounding boxes from the proposals more accurate. These final layers attempt to minimize the loss in obtaining the precise bounding boxes.

HSE Official - Not for circulation

Fast R-CNN performs forty-five times faster than R-CNN at test time and it gives 9 times faster results during training time.

Faster R-CNN

In Fast R-CNN, selective search method was used for region proposal where which consumes around 2 seconds per image. Faster R-CNN was proposed to reduce this delay further. As shown in the figure below, the faster R-CNN consists of Fast R-CNN and Region Proposal Network. The region proposal network is a convolution layer which helps in reducing the delay of region proposal to 10ms. Here extraction of features is performed using CNN and then passes it on to RPN. RPN returns the object proposals which is passed onto ROI pooling layer that creates uniform size for all proposals. This output is given to a fully connected layer which can classify the objects. Here, Region Proposal Network shares the layers with the Fast R-CNN block which helps in improved way of representing the features. The Fast R-CNN block acts as the detector for Faster R-CNN. Here, RPN and Fast R-CNN are trained separately and fine-tuned for their respective tasks of Region Proposal and Object detection. Then based on this training, weights are shared, and each block is fine-tuned again for Region Proposal and Object detection, respectively.

HPE Official - Not for Circulation

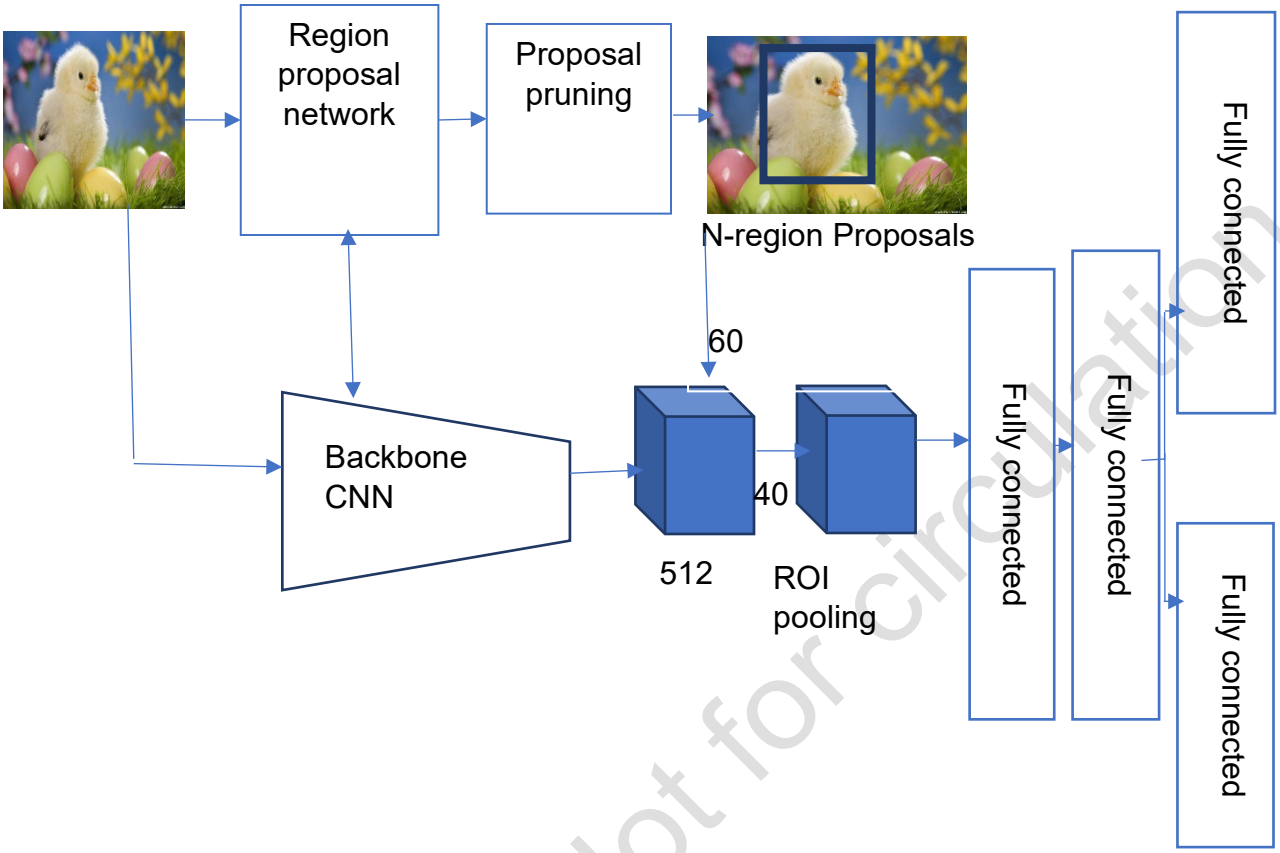


Figure 3.15 Faster R-CNN Architecture

The object detection speed in faster R-CNN is reduced to milliseconds range compared to the time in seconds required in Fast R-CNN.

Yolo

This algorithm utilizes a single Convolutional Neural Network. It stands for you look only once. It performs classification and bounding box regression in one step. It is much faster than Fast R-CNN and Faster R-CNN. It has multi class prediction capabilities. It performs well with small object classification while it performs bad for large or medium objects. Yolo V3 and Yolo V4 are some of the versions of this type of classification. The release AI v2.1.0 supports training a YOLO model to detect any type of objects. YOLO sees the entire image during training and test time, because of which it is capable of encoding context of the classes. It makes much lesser errors in comparison with Fast R-CNN.

HCL Confidential - Not for circulation

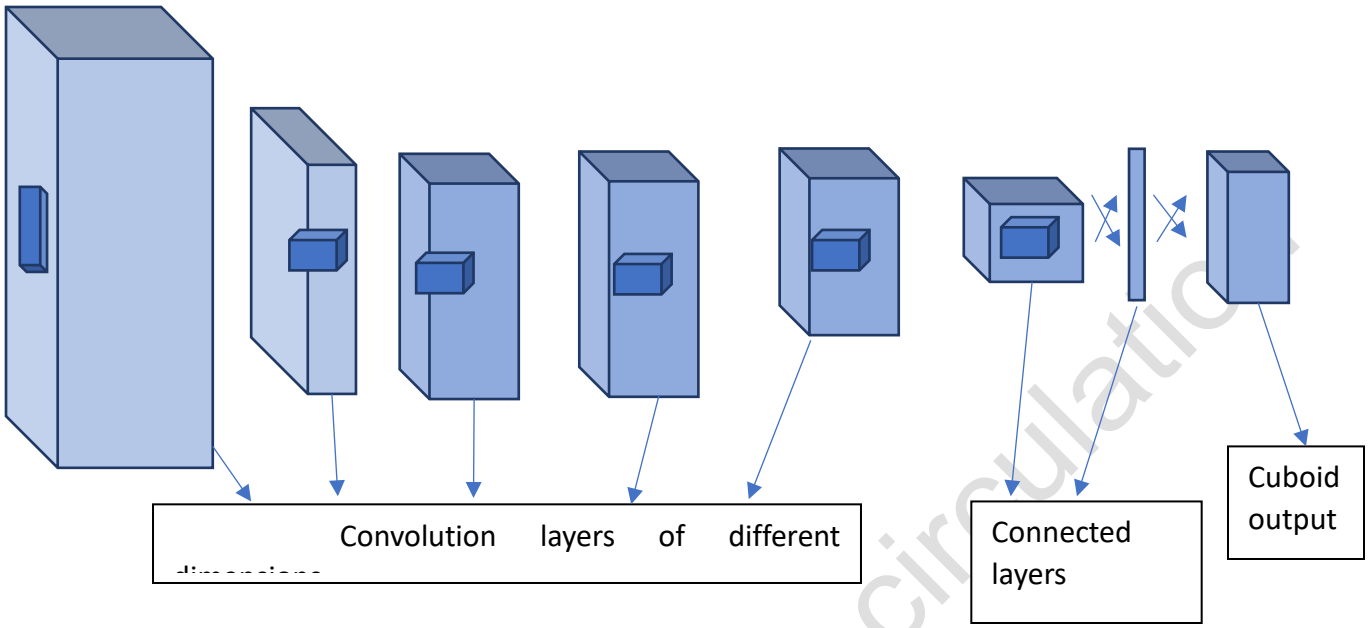


Figure 3.16 YOLO Architecture

Yolo takes an image as input and resizes it. It is passed onto CNN network. The version 1 has twenty-four convolution layers, 4 max-pooling layer layers and 2 fully connected layers. For the reduction of layers, 1*1 convolution is used first and 3*3 convolution next. The last layer predicts cuboid output. It uses Leaky RELU activation function and linear activation function in the architecture. Regularization is performed using batch normalization.

3.11Form Recognition

Consider an organization which has thousands of bills and invoices which must be documented digitally. These data are very precious. It is impossible for an employee to scan all the documents and create a database as its tedious and time consuming. Data entry operators spend hours to complete this task. Form Recognizers extract data from handwritten or digital invoices, forms, and receipts with the help of artificial intelligence within a span of few seconds. Form Recognition helps to provide a well structure database which can be searched easily for information. Also, data science and machine learning algorithms could be applied on this data to make predictions and reduce paper usage.

HPE Official - Not for circulation

Optical Character Recognition (OCR) algorithms can extract the information and automatically digitize these documents and store it in a database. OCR has two parts: In the image, the text must be detected which is called as text detection. Later the text must be extracted known as text recognition. For unstructured documents, OCR gives poor results. To avoid disadvantages of OCR, Intelligent Data Processing (IDP) can be used. IDP uses preprocess and extract the data using multiple AI technologies.AI Capabilities integrated with IDP can handle variations and understand that is read from documents and images. Machine learning and deep learning algorithms learn from existing data which increases the accuracy of extraction.

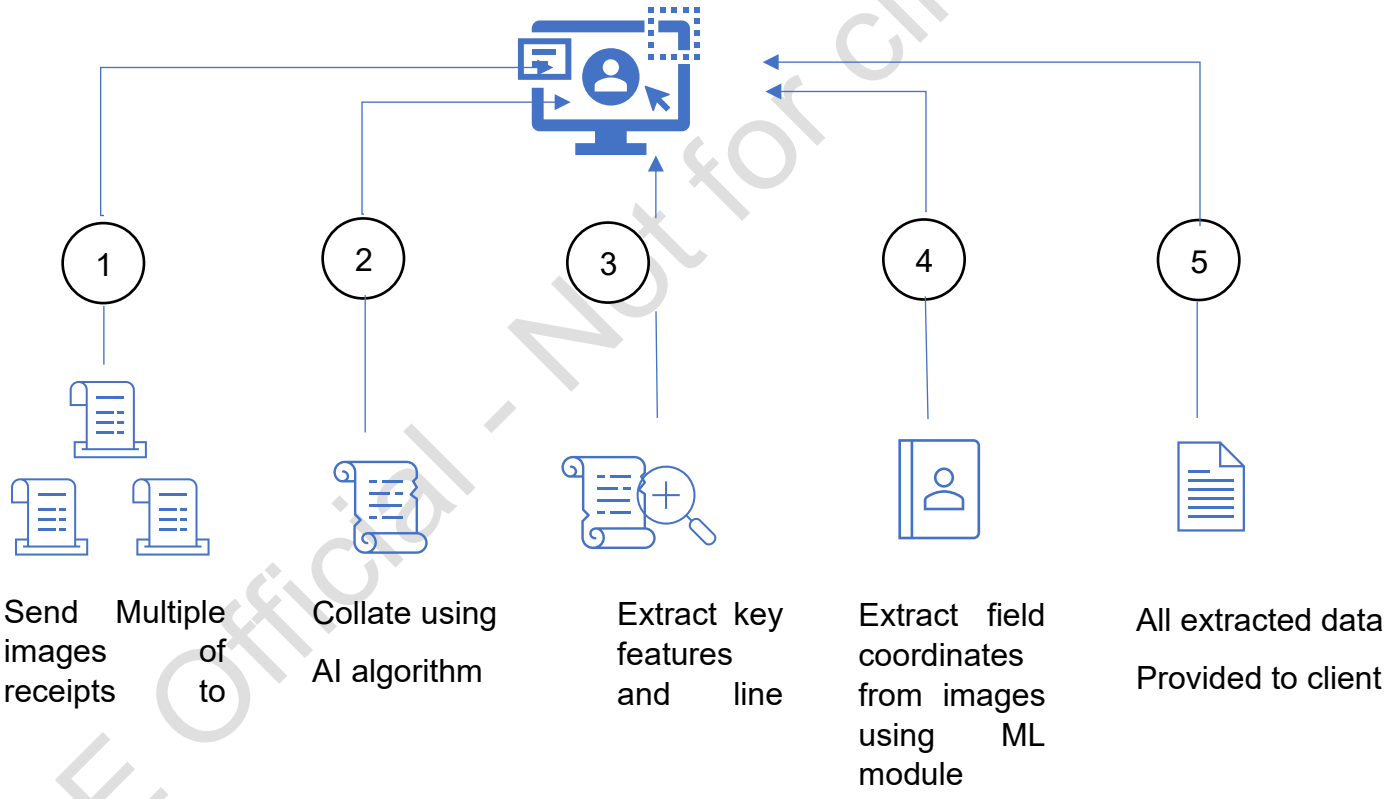


Figure 3.17. Process of form recognition

Multiple images of the receipts will be sent to the API. OCR engines are used to extract text from the documents. Using machine learning algorithms all the images will be collated. The key features and items of line are extracted by using extraction module. Also

HPE Official - Not for circulation

features and key values are extracted from handwritten images. Finally, all the extracted data are available for client use.

Prebuilt Model

Form recognizer can support several types of documents. Prebuilt model allows to upload the document, select the type of file and by running the model all the data in the form can be obtained in a digital format. Form can be of several types like receipts, invoices, and ID cards.

3.12 Anomaly Detection

A variation from the normal is known as Anomaly. In nature, certain processes or behaviors exist that follow certain broad principles which is observed as data that results in a state of the system. Systems also exhibit abnormal states leading to different data values when no such process/state variations occur. Such variations must be discovered with the use of anomaly detection. Three cases need to be considered when anomaly detection is applied:

1.Correct Detection: Abnormalities detected in data correspond exactly to abnormalities in the process.

2.False Positives: Unexpected data values are observed, but process continues to be normal. For example, due to intrinsic system noise.

3.False Negatives: When compared to noise in the system, signal of the abnormality becomes insufficiently strong, the consequences do not get registered in the abnormal data, but the process becomes abnormal.

Anomaly detection is the identification of patterns that deviate from normal behavior. These abnormal instances are known as outliers(anomalies) and normal instances are known as inliers. The reasons for anomalies can be data errors, noise, and hidden patterns in the dataset like fraud requests. There are two directions to search anomalies: They are

i)Outlier detection: If an observation differs from other data points, its known as outlier.

HPE Official - Not for circulation

ii)Novelty detection: If the training data does not have any outlier and detecting anomaly in the new observation.

Anomaly detection can be univariate or Multivariate. If a single indicator is measured, its univariate. If a host of factors are considered, then its Multivariate. In Multivariate, it consists of a combined score of at least two variables.

3.12.1Algorithms for Anomaly detection

Algorithms for Anomaly detection can be classified into supervised and unsupervised. Usually, data for supervised anomaly detection are highly imbalanced. Data augmentation procedures like SMOTE, k-nearest neighbors' algorithm should be used before applying supervised classification methods.

When no information about anomalies and related patterns are available, unsupervised anomaly detection can be used. The various methods are Isolation forests, k-means, and Neural Networks.

3.12.2 Isolation Forest

Random Forest can be used to detect anomaly in high-dimensional datasets. Randomly observations are isolated by selecting a feature and a split value is selected between maximum and minimum values of the selected feature. This method takes advantage of two properties of anomalies, i) They are minorities consisting of fewer instances ii) compared to normal instances, their attribute values are quite different. As anomalies are few and different, every single instance can be isolated with a tree structure. Anomalies fall closer to the tree root and normal points will be at the deeper end of the tree. Detection of anomalies are based on these isolation characteristics of tree known as Isolation Tree or iTree.

Isolation Forest or iForest is an ensemble anomaly detection algorithm useful when dataset is high dimensional. At each node, a random forest is created where decision trees are grown randomly. A random threshold value is picked to split the dataset. The dataset is chopped away until all instances are isolated from each other. An anomaly will

HPE Official - Not for circulation

be far away from normal instances, and it becomes isolated from other instances. On the iTree, instances that have short average path lengths are anomalies.

3.12.3 Use Cases

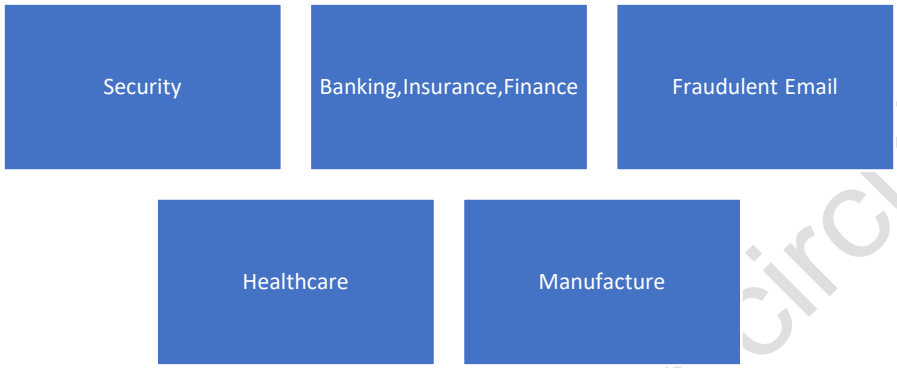


Figure 3.18. Applications of Anomaly Detection

Suspicious Activities and frauds can be prevented with the help of anomaly detection including

Security: Anomaly detection plays a key role in cybersecurity. Consider a criminal hacking access to the account and learning the pattern of the user and pretends to be the user and later launches malicious attacks. Each single attack may look normal, but when all linked together it may be possible to detect abnormal pattern. If an administer file is accessed too many times or too many times of system logins after office hours may indicate anomalous behavior. Thus, suspicious activities and other unauthorized access attempts can be identified faster providing overall cybersecurity.

Banking, Finance, and Insurance: Many cases like credit card fraud and unauthorized debit card usage leads to a loss to the customers every year. In 2013, almost 40 million credit cards were compromised and 70 million suffered confidential data breach. As years go by, the number of cases is increasing and even with all security measures many times some transactions go undetected if the amount is small. Algorithms based on recent purchase data could help to prevent such issues. Bankruptcy prediction and creditworthiness of a customer based on credit history can help to avoid frauds.

HPE Official - Not for circulation

Fraudulent Email: Emails play a key role in everyday transactions. Many times, individuals are lulled to click on harmful websites and provide personal data which could later be sold on the darknet. Anomalies can be detected based on message sizes, unknown senders, arriving at an unusual time, contains unusual typographical errors or asks the user to click on a link.

HealthCare: Anomaly detection can be used for diagnosis and monitoring of patients. Machine learning and classification algorithms are applied to detect whether a tumor is malignant or not. Also, Covid is detected using machine learning algorithms from X Ray or CT scan. This can help in early detection of cancer and provide effective treatment.

Manufacture: Various hardware maintenance can be done using sensors and periodic monitoring of working state and notifying any unusual behavior in a timely manner. This help to avoid precedented hardware shutdown and industrial damages.

3.13. Introduction to AI Reader

Consider if a person is visually impaired or old-aged who find it difficult to read the text. Artificial intelligence can play a key role to help them to read documents. AI reader is the software that converts a digital or printed file to a suitable output in the form of speech. The user can select the voice, pitch, volume, and pace of speech. It gives the user the experience of reading without having to read the document. Text is converted to speech and supports different languages.

If it is a printed matter, it must be scanned. An optical character recognition and machine learning is used to convert the text to ASCII file. The ASCII file is converted to audible words and sentences using speech synthesizer. The speech synthesis technology (TTS) synthesizes digital format text into human voice and will be played through an audio system. The sentences will be converted into natural language just the same way as any human would speak in natural language. Over a decade, technology has improved in such a way that natural voice is replicated using synthetic speech.

HPE Official - Not for circulation

Techniques of Natural Language processing are used for text synthesis. The Deep Learning approaches are used to synthesis speech. Deep Neural Networks are used to model the input texts and their acoustic realizations relationship. Convolutional neural networks and Recurrent Neural networks are used to process audio. A voice codec which encrypts and compresses audio signal is known as vocoder. In case of neural vocoder, encoding/decoding is achieved by neural networks.

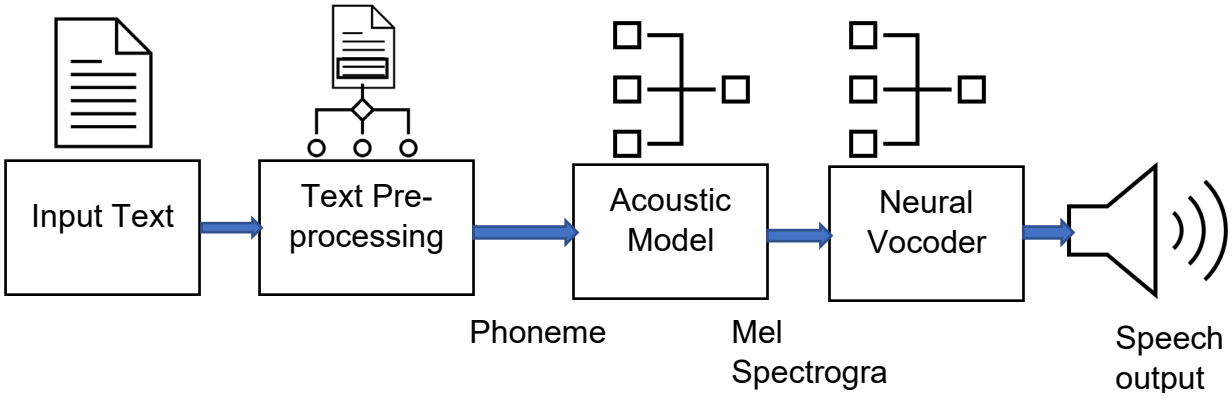


Figure 3.19. Text to Speech Synthesizer

The input text will be converted into the linguistic features of the target language. Normalization (Jan to January) is done. The phoneme is generated and this vector output is fed to acoustic model. Mel Spectrograms can be considered as image of audio and can be labelled with text which allows for classification of audio. The acoustic model preprocess text and generates Mel spectrogram which accounts for all audio features. The neural vocoder converts the Mel spectrogram into a speech waveform. Currently neural vocoders are implemented using generative adversarial networks (GAN).

3.14 Introduction to SmartSim

SmartSim is an open-source framework that enables the use of Artificial Intelligence (AI) with existing traditional high-performance computing (HPC) simulations just by adding couple of lines of code. SmartSim arms researchers and engineers with a completely new communication paradigm between simulations and AI methodologies offering them many

HPE Official - No

additional functionalities, which help them run their computational science projects more easily and in less time.

SmartSim provides this capability by

1. Automating the deployment of HPC workloads and distributed, in-memory storage (Redis).
2. Making TensorFlow, Pytorch, and ONNX callable from Fortran, C, and C++ simulations.
3. Providing flexible data communication and formats for hierarchical data, enabling online analysis, visualization, and processing of simulation data.

Figure 3.20. Context of SmartSim

The main goal of SmartSim is to provide scientists a flexible, easy to use method for interacting at runtime with the data generated by simulation. The type of interaction is completely up to the user.

- Embed calls to machine learning models inside a simulation
- Create hooks to steer a simulation manually or programmatically
- Visualize the progression of a simulation integration from a Jupyter notebook

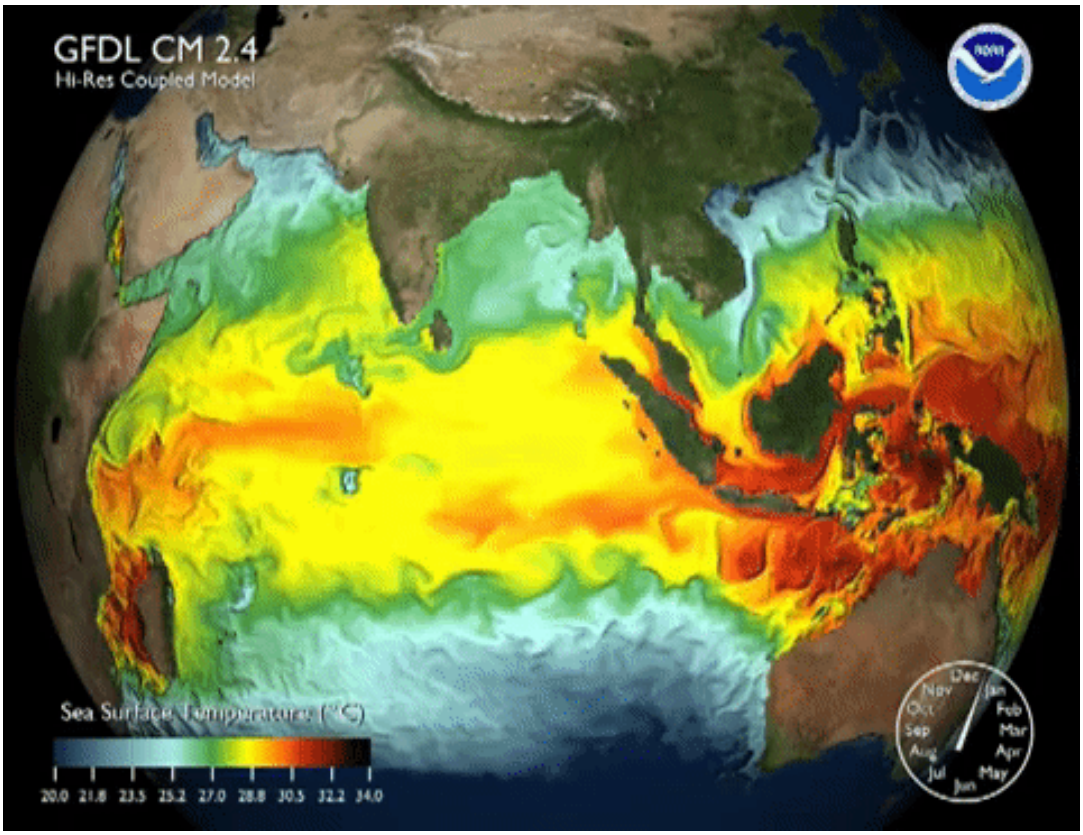


Figure 3.21. Simulation

The figure below shows the architecture of SmartSim for a given use case. SmartSim can create, configure, and launch workloads (called a Model), as well as groups of workloads (Ensembles). The data communication between a workload and in-memory storage is handled by the SmartRedis clients, available in Fortran, C, C++, and Python.

HPE

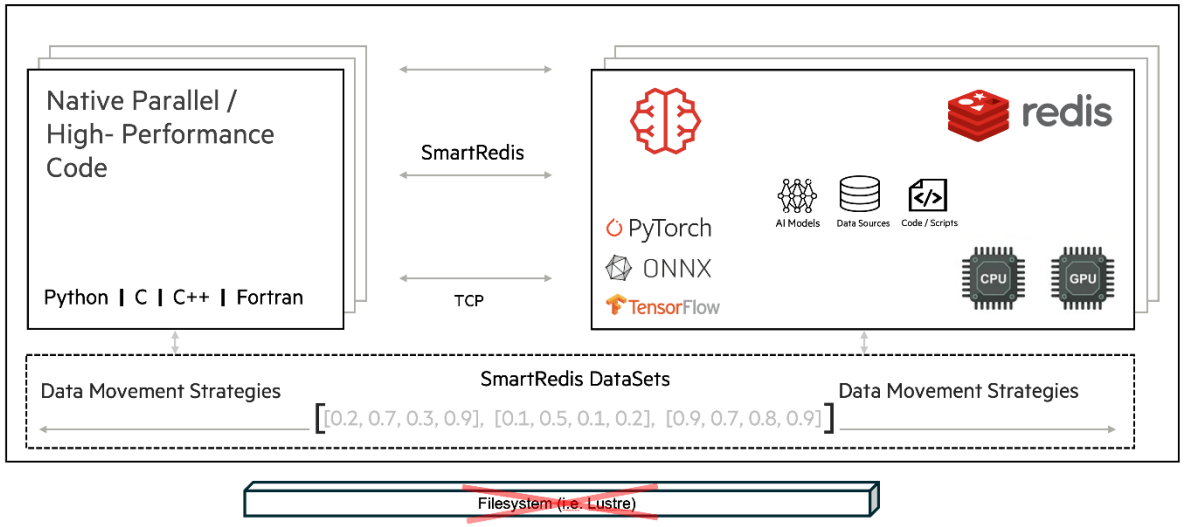


Figure 3.22. Library Design

There are two core components of SmartSim:

1. SmartSim (infrastructure library)
2. SmartRedis (client library)

The two libraries can either be used in conjunction or separately, depending on the needs of the user.

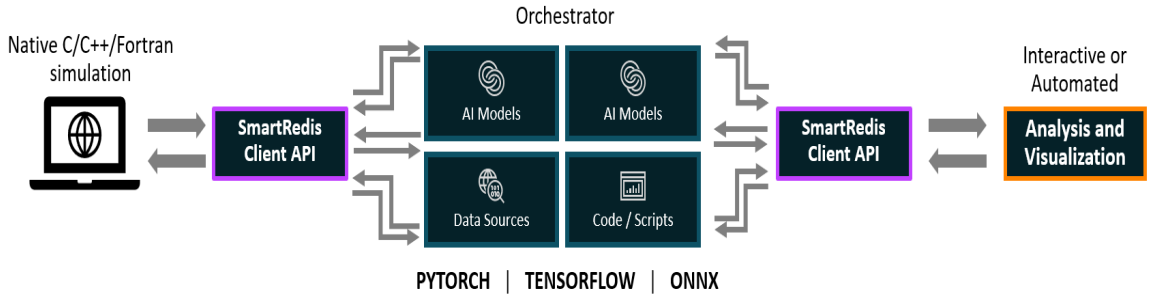


Figure 3.23. SmartSim (infrastructure library)

The infrastructure library (IL) provides an API to automate the process of deploying HPC workloads alongside an in-memory database: Redis.

The key features of the IL are:

- An API to start, monitor, and stop HPC jobs from Python or from a Jupyter notebook.
- Automated deployment of in-memory data staging (Redis) and computational storage (RedisAI).
- Programmatic launches of batch and in-allocation jobs on PBS, Slurm, LSF, and Cobalt systems.
- Creating and configuring ensembles of workloads with isolated communication channels.

The IL can configure and launch batch jobs as well as jobs within interactive allocations. The IL integrates with workload managers, (like Slurm and PBS), if it is running on a supercomputer or cluster system.

The IL can deploy a distributed, shared-nothing, in-memory cluster of Redis instances across multiple compute nodes of a supercomputer, cluster, or laptop. In SmartSim, this clustered Redis deployment is called the Orchestrator.

By coupling the Orchestrator with HPC applications, users can connect their workloads to other applications, like trained machine learning models, with the SmartRedis clients.

HPE Official - Not for circulation

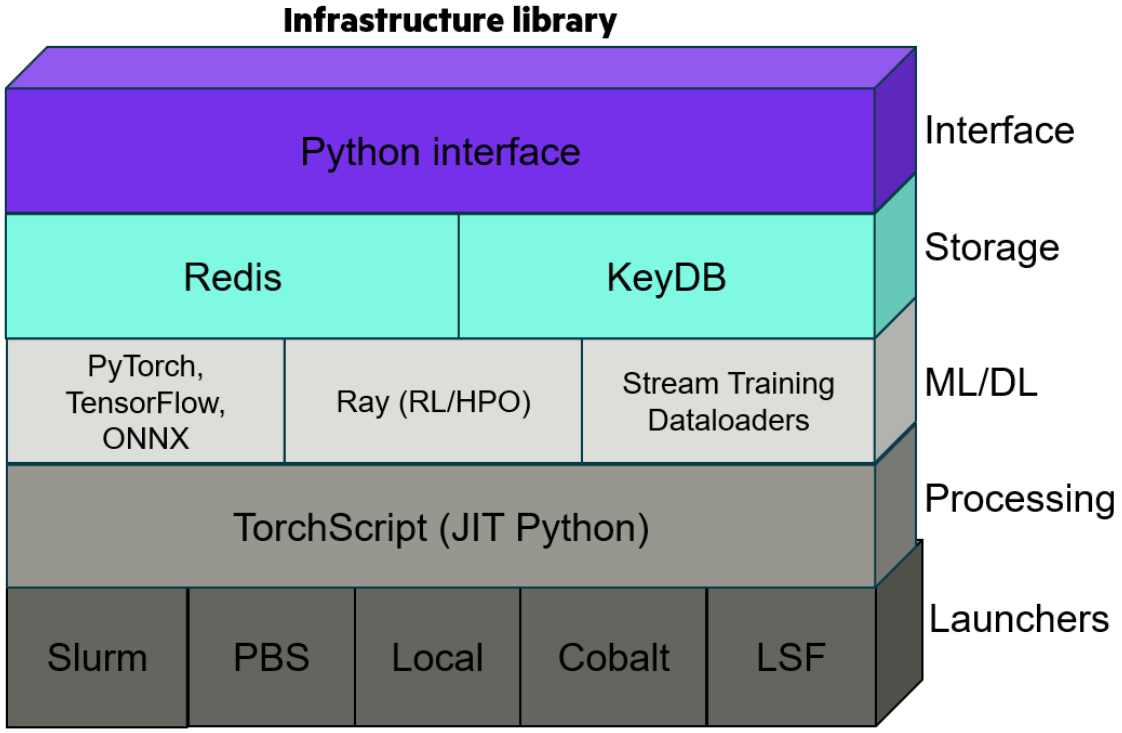


Figure 3.24. SmartRedis (Client Library)

SmartRedis is a collection of Redis clients that support RedisAI capabilities and include additional features for high performance computing (HPC) applications. Key features of RedisAI that are supported by SmartRedis include:

- A tensor data type in Redis
- TensorFlow, TensorFlow Lite, Torch, and ONNXRuntime backends for model evaluations
- TorchScript storage and evaluation

In addition to the RedisAI capabilities above, SmartRedis includes the following features developed for large, distributed HPC architectures:

- Redis cluster support for RedisAI data types (tensors, models, and scripts).
- Distributed model and script placement for parallel evaluation that maximizes hardware utilization and throughput

HPE Official - Not

- A Dataset storage format to aggregate multiple tensors and metadata into a single Redis cluster hash slot to prevent data scatter on Redis clusters and maintain contextual relationships between tensors. This is useful when clients produce tensors and metadata that are referenced or utilized together.
- An API for efficiently aggregating Dataset objects distributed on one or more database nodes.
- Compatibility with SmartSim ensemble capabilities to prevent key collisions with tensors, Dataset, models, and scripts when clients are part of an ensemble of applications.

SmartRedis provides clients in Python, C++, C, and Fortran. These clients have been written to provide a consistent API experience across languages, within the constraints of language capabilities. The table below summarizes the language standards for each client.

HPE Official - Not for circulation

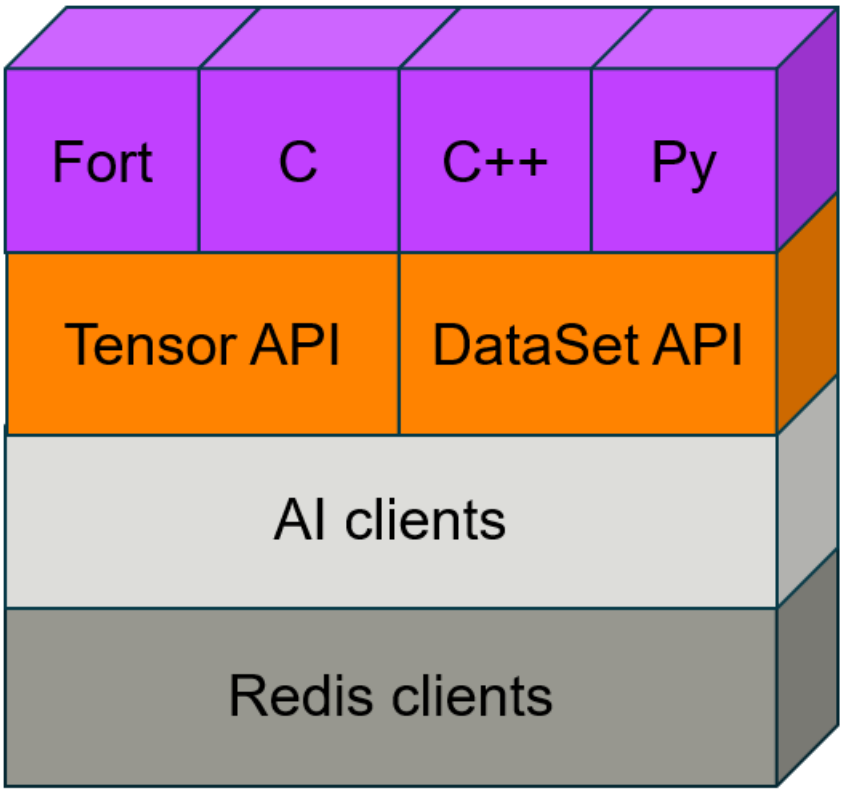


Figure 3.25. SmartRedis (Client Library)

HPE Official - ation

Client data structure	API	Description
Tensor	Client.put_tensor()	Set a tensor
	Client.get_tensor()	Retrieve a tensor
	Client.unpack_tensor()	Fill user-provided memory space with retrieved tensor
Model	Client.set_model()	Store and distribute ML/DL model
	Client.get_model()	Retrieve model
	Client.run_model()	Execute a ML/DL model on some stored data
Script	Client.set_script()	Store and distribute TorchScript script
	Client.get_script()	Retrieve a TorchScript script
	Client.run_script()	Run a TorchScript function within a script on some data

HPE Official - Not for distribution

DataSet actions	API	Description
Construct	DataSet.add_tensor()	Add a tensor to the DataSet
	DataSet.add_meta_scalar()	Add a metadata value to a scalar field
	DataSet.add_string_scalar()	Add a metadata value to a string field
Store	Client.put_dataset()	Set a DataSet
	Client.get_dataset()	Retrieve a DataSet
Inspect	DataSet.get_tensor()	Get a tensor from the DataSet
	DataSet.unpack_tensor()	Fills user-provided memory space with tensor

Experiments

The Experiment acts as both a factory class for constructing the stages of an experiment (Model, Ensemble, Orchestrator, etc.) as well as an interface to interact with the entities created by the experiment.

Users can initialize an Experiment at the beginning of a Jupyter notebook, interactive python session, or Python file and use the Experiment to iteratively create, configure and launch computational kernels on the system through the specified launcher.

Hyperion - Not for circulation

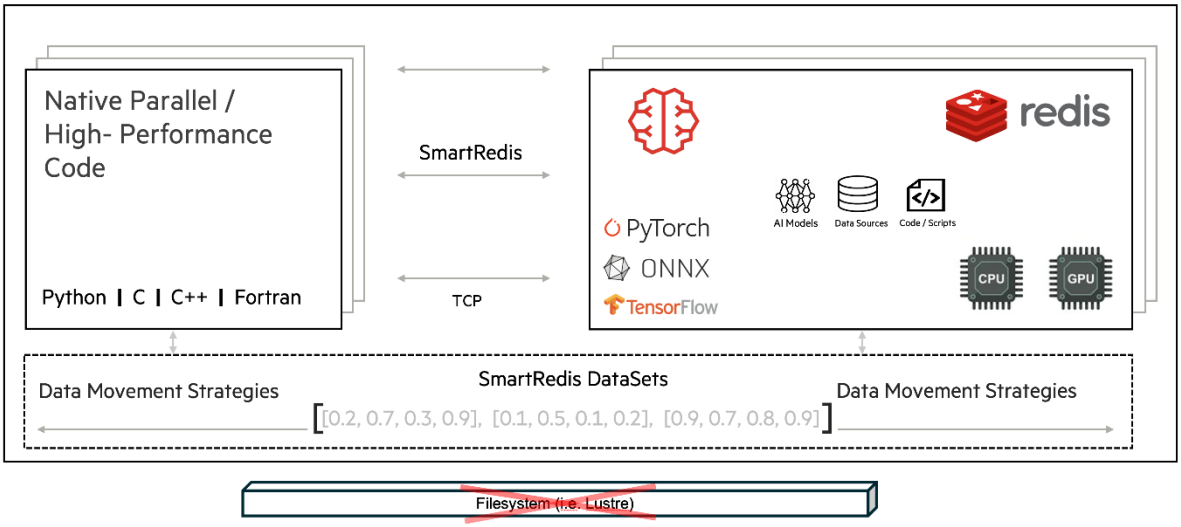


Figure 3.26. Smartsim Architecture

The interface was designed to be simple, with as little complexity as possible, and agnostic to the backend launching mechanism (local, Slurm, PBSPro, etc.).

Model

Models are subclasses of SmartSimEntity(s) and are created through the Experiment API. Models represent any computational kernel. Models are flexible enough to support many different applications, however, to be used with our clients (SmartRedis) the application will have to be written in Python, C, C++, or Fortran.

Models are given RunSettings objects that specify how a kernel should be executed with regard to the workload manager (e.g. Slurm) and the available compute resources on the system.

Each launcher supports specific types of RunSettings.

- SrunSettings for Slurm
- AprunSettings for PBSPro and Cobalt
- MpirunSettings for OpenMPI with *mpirun* on PBSPro, Cobalt, LSF, and Slurm
- JsrunSettings for LSF

HPE Official - Not

These settings can be manually specified by the user, or auto-detected by the SmartSim Experiment through the Experiment.create_run_settings method.

A simple example of using the Experiment API to create a model and run it locally:

```
from smartsim import Experiment

exp = Experiment("simple", launcher="local")

settings = exp.create_run_settings("echo", exe_args="Hello World")
model = exp.create_model("hello_world", settings)

exp.start(model, block=True)
print(exp.get_status(model))
```

Ensemble

In addition to a single model, SmartSim can launch an Ensemble of Model applications simultaneously.

An Ensemble can be constructed in three ways:

1. Parameter expansion
2. Replica creation (by specifying replicas argument)
3. Manually (by adding created Model objects) if launching as a batch job

Ensembles can be given parameters and permutation strategies that define how the Ensemble will create the underlying model objects.

Three strategies are built in:

1. all_perm: for generating all permutations of model parameters
2. step: for creating one set of parameters for each element in *n* arrays
3. random: for random selection from predefined parameter spaces

HPE Official - Not for circulation

Here is an example that uses the random strategy to initialize four models with random parameters within a set range. We use the params_as_args field to specify that the randomly selected learning rate parameter should be passed to the created models as a executable argument.

```
import numpy as np
from smartsim import Experiment

exp = Experiment("Training-Run", launcher="auto")

# setup ensemble parameter space
learning_rate = list(np.linspace(.01, .5))
train_params = {"LR": learning_rate}

# define how each member should run
run = exp.create_run_settings(exe="python",
                              exe_args="./train-model.py")

ensemble = exp.create_ensemble("Training-Ensemble",
                               params=train_params,
                               params_as_args=["LR"],
                               run_settings=run,
                               perm_strategy="random",
                               n_models=4)
exp.start(ensemble, summary=True)
```

A callable function can also be supplied for custom permutation strategies. The function should take two arguments: a list of parameter names, and a list of lists of potential parameter values. The function should return a list of dictionaries that will be supplied as

HPE Official - Not for circulation

model parameters. The length of the list returned will determine how many Model instances are created.

For example, the following is the built-in strategy all_perm:

```
from itertools import product

def create_all_permutations(param_names, param_values):
    perms = list(product(*param_values))
    all_permutations = []
    for p in perms:
        temp_model = dict(zip(param_names, p))
        all_permutations.append(temp_model)
    return all_permutations
```

After Ensemble initialization, Ensemble instances can be passed as arguments to Experiment.generate() to write assigned parameter values into attached and tagged configuration files.

Orchestrator

The Orchestrator is an in-memory database that is launched prior to all other entities within an Experiment. The Orchestrator can be used to store and retrieve data during an experiment and across multiple entities. To stream data into or receive data from the Orchestrator, one of the SmartSim clients (SmartRedis) must be used within a Model.

For Official Use Only - Not for circulation

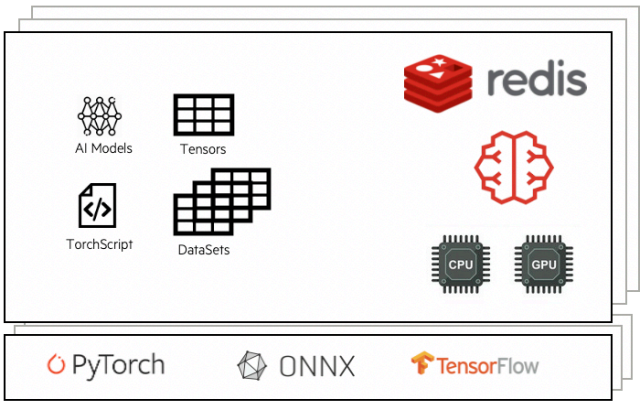


Figure 3.27. SmartSim Objects

Combined with the SmartRedis clients, the Orchestrator is capable of hosting and executing AI models written in Python on CPU or GPU. The Orchestrator supports models written with TensorFlow, Pytorch, TensorFlow-Lite, or models saved in an ONNX format (e.g. sci-kit learn).

Cluster Orchestrator

The Orchestrator supports single node and distributed memory settings. This means that a single compute host can be used for the database or multiple by specifying db_nodes to be greater than 1.

HPE Official - No

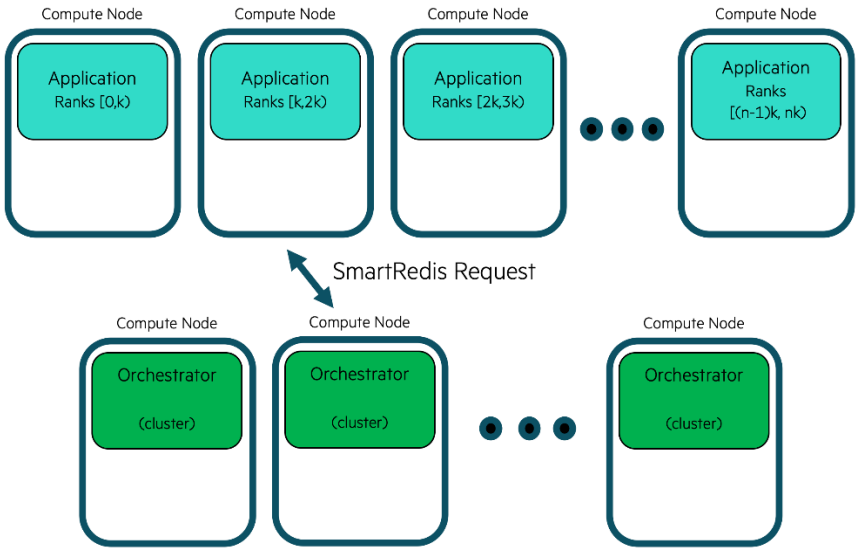


Figure 3.28. Orchestrator

With a clustered Orchestrator, multiple compute hosts memory can be used together to store data. As well, the CPU or GPU(s) where the Orchestrator is running can be used to execute the AI models, and Torchscript code on data stored within it.

Users do not need to know how the data is stored in a clustered configuration and can address the cluster with the SmartRedis clients like a single block of memory using simply put/get semantics in SmartRedis. SmartRedis will ensure that data is evenly distributed amongst all nodes in the cluster.

The cluster deployment is optimal for high data throughput scenarios such as online analysis, training, and processing.

Redis

The Orchestrator is built on Redis. The job of the Orchestrator is to create a Python reference to a Redis deployment so that users can launch, monitor, and stop a Redis deployment on workstations and HPC systems.

HPE Official - Not for C tion

Redis was chosen for the Orchestrator because it resides in-memory, can be distributed on-node as well as across nodes, and provides low latency data access to many clients in parallel. The Redis ecosystem was a primary driver as the Redis module system provides APIs for languages, libraries, and techniques used in Data Science. In particular, the Orchestrator relies on RedisAI to provide access to Machine Learning runtimes.

At its core, Redis is a key-value store. This means that put/get semantics are used to send messages to and from the database. SmartRedis clients use a specific hashing algorithm, CRC16, to ensure that data is evenly distributed amongst all database nodes. Notably, a user is not required to know where (which database node) data or Datasets (see Dataset API) are stored as the SmartRedis clients will infer their location for the user.

SmartRedis

SmartRedis is a collection of Redis clients that support RedisAI capabilities and include additional features for high performance computing (HPC) applications. Key features of RedisAI that are supported by SmartRedis include:

- A tensor data type in Redis
- TensorFlow, TensorFlow Lite, Torch, and ONNXRuntime backends for model evaluations
- TorchScript storage and evaluation

In addition to the RedisAI capabilities above, SmartRedis includes the following features developed for large, distributed HPC architectures:

- Redis cluster support for RedisAI data types (tensors, models, and scripts).
- Distributed model and script placement for parallel evaluation that maximizes hardware utilization and throughput
- A Dataset storage format to aggregate multiple tensors and metadata into a single Redis cluster hash slot to prevent data scatter on Redis clusters and maintain contextual relationships between tensors. This is useful when clients produce tensors and metadata that are referenced or utilized together.

- An API for efficiently aggregating Dataset objects distributed on one or more database nodes.
- Compatibility with SmartSim ensemble capabilities to prevent key collisions with tensors, Dataset, models, and scripts when clients are part of an ensemble of applications.

SmartRedis provides clients in Python, C++, C, and Fortran. These clients have been written to provide a consistent API experience across languages, within the constraints of language capabilities. The table below summarizes the language standards for each client.

3.15 Usecases Of Smartsim

1. Online Training

Simulating a complex phenomenon can be computationally intensive and expensive. In some cases, the computational model is too expensive or too slow to be used in production, for example, when results are needed in real time. In other cases, a correct computational model, capable of predicting the behavior of a complex system, might not even exist.

In this notebook, a neural network is trained to act like a surrogate model and to solve a well-known physical problem, i.e., computing the steady state of heat diffusion. The training dataset is constructed by running simulations *while* the model is being trained.

HPE Official - Not for

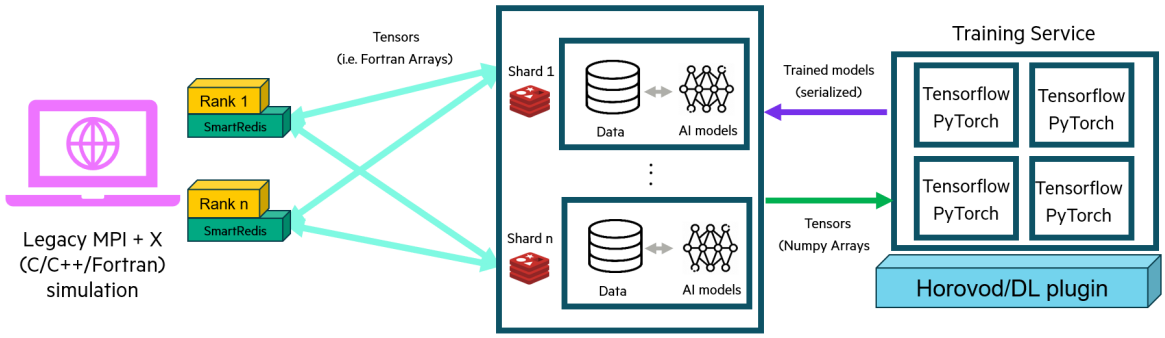
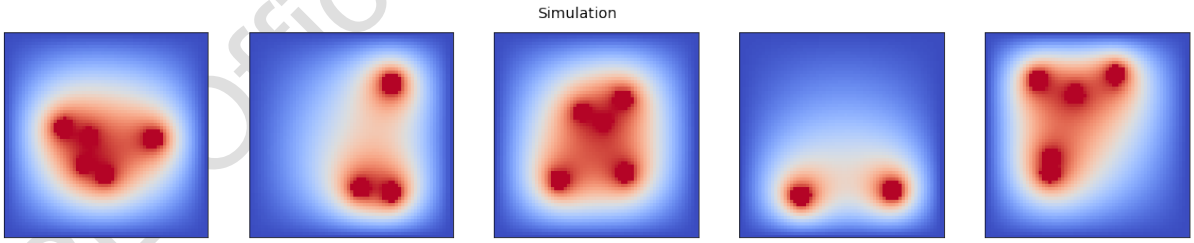


Figure 3.29 Online Training

2D Heat Diffusion and Steady State

The heat equation solved on the two-dimensional domain, setting the initial temperature to 0 K, except for some specific spots (of radius 0.05), where heat sources at a constant temperature of 1 K are placed. Since all boundaries are kept at 0 K, after enough time, the domain will reach a steady state, which is the outcome the convolutional Neural Network surrogate model will learn to compute. The figure shows the actual simulation and the outcome of the surrogate model created using SmartSim.



HPE Official - Not for circ

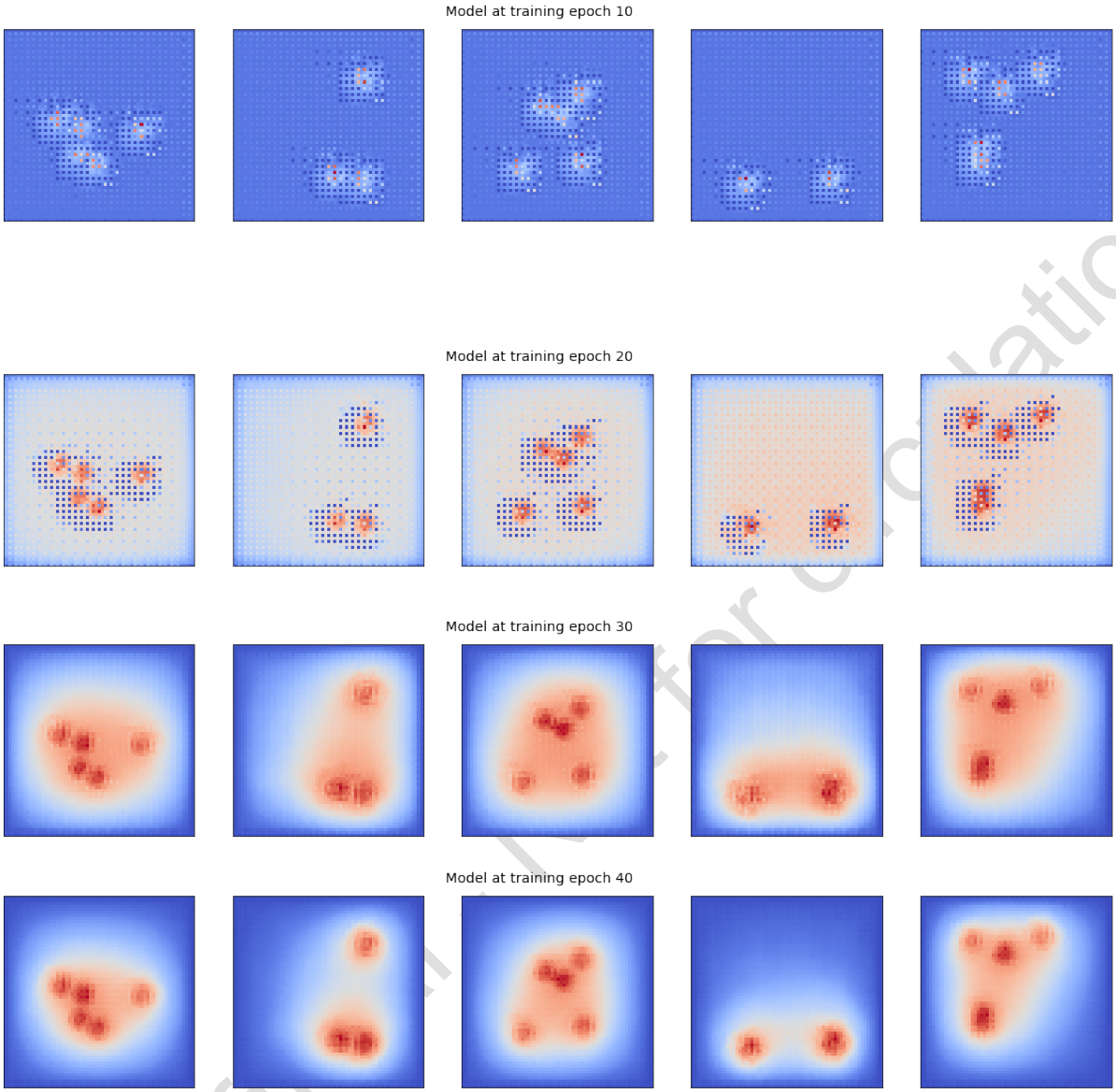


Figure 3.30 Heat Simulation by Surrogate Model

2. Online Inference

We can use ML/DL models to infer using SmartSim. How to use trained PyTorch, TensorFlow, and ONNX (format) models, written in Python, directly in HPC workloads written in Fortran, C, C++, and Python.

HPE Offi... Not for circulation

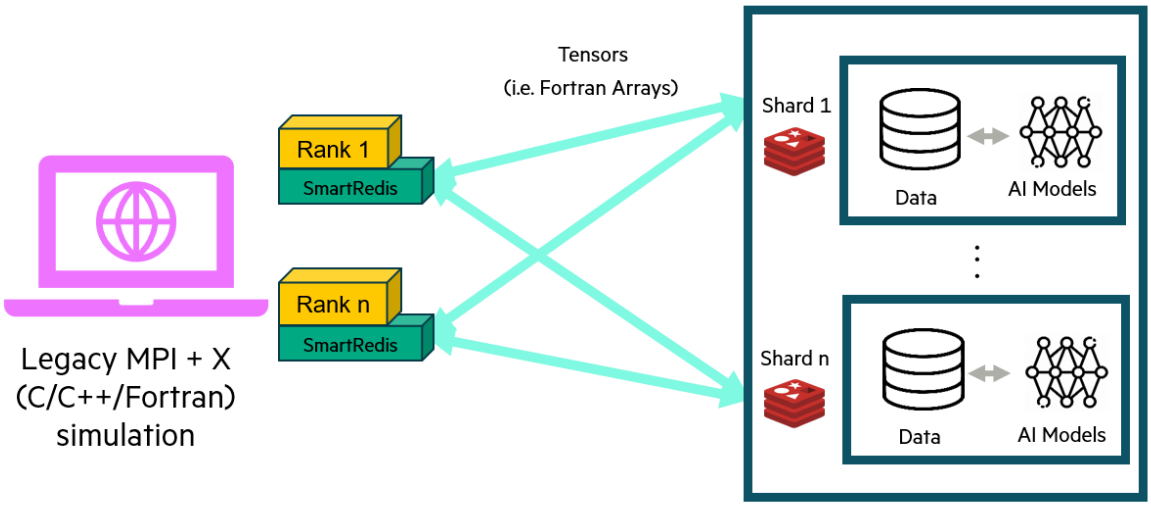


Figure 3.31 Online Inference

The example simulation here is written in Python for brevity, however, the inference API in SmartRedis is the same (besides extra parameters for compiled languages) across all clients.

Starting the Database for Inference

SmartSim performs online inference by using the SmartRedis clients to call into the Machine Learning (ML) runtimes linked into the Orchestrator database. The Orchestrator is the name in SmartSim for a Redis or KeyDB database with a RedisAI module built into it with the ML runtimes.

Therefore, to perform inference, you must first create an Orchestrator database and launch it. There are two methods to couple the database to your application to add inference capability to your application. - standard (not co-located) - co-located

standard mode launches an optionally clustered (across many compute hosts) database instance that can be treated as a single storage device for many clients (the many ranks of an MPI program) where there is a single address space for keys across all hosts.

HPE Official - Not for

co-located mode launches an orchestrator instance on each compute host used by a, distributed, application. each instance contains their own address space for keys. In SmartSim, Model instances can be launched with a co-located orchestrator through Model.colocate_db. Co-located Models are used for highly scalable inference where global aggregations are not necessary for inference.

3. Online Analysis

Being able to visualize and interpret simulation data in real time is invaluable for understanding the behavior of a physical system. SmartSim can be used to stream data from Fortran, C, and C++ simulations to Python where visualization is significantly easier and more interactive.

SmartSim can be used to analyze the vorticity field during a simple, Python based Philip Mocz's implementation Lattice Boltzmann fluid flow simulation. This is a cool and simple technique to simulate fluid flow: instead of evolving the fluid (Navier-Stokes) equations directly, microscopic particles on a lattice are simulated with streaming and collision processes. The power of the method comes from reducing the high dimensionality of the microscopic physics onto the lattice, which has limited degrees of freedom.

Figure 3.32 Simulation

HPE Confidential - Not for circulation

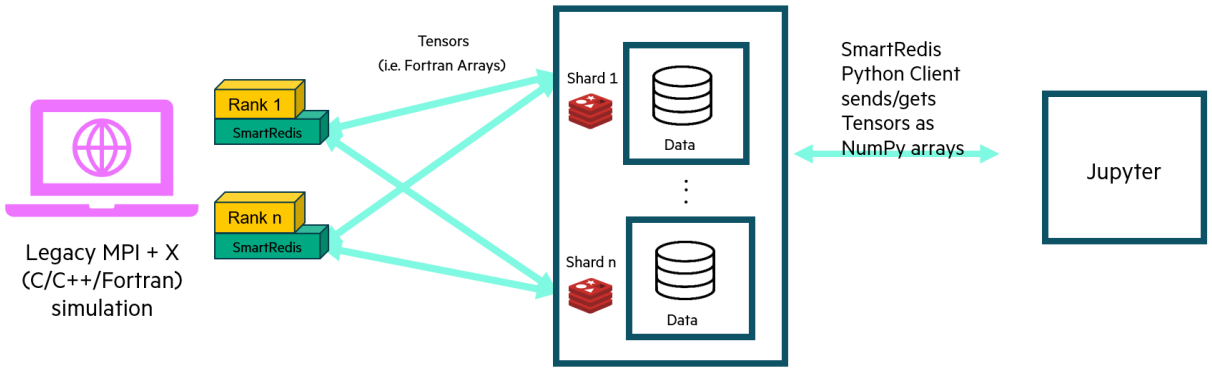
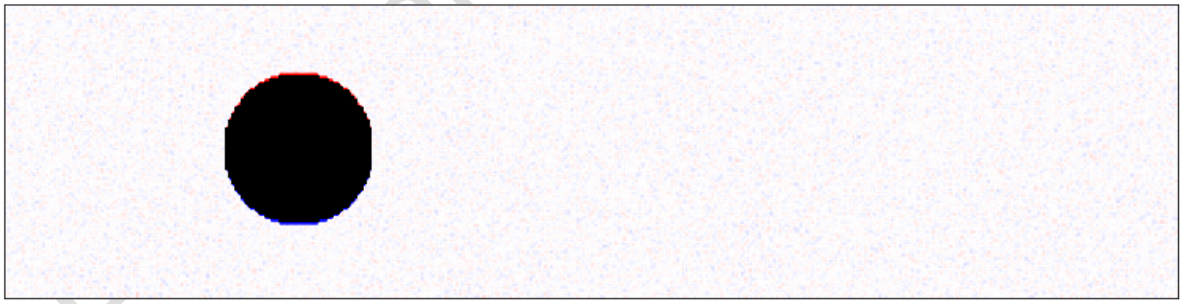


Figure 3.33 Online Analysis

Typically, HPC simulations are written in C, C++, Fortran, or other high-performance languages. Embedding the SmartRedis client usually involves compiling in the SmartRedis library into the simulation.

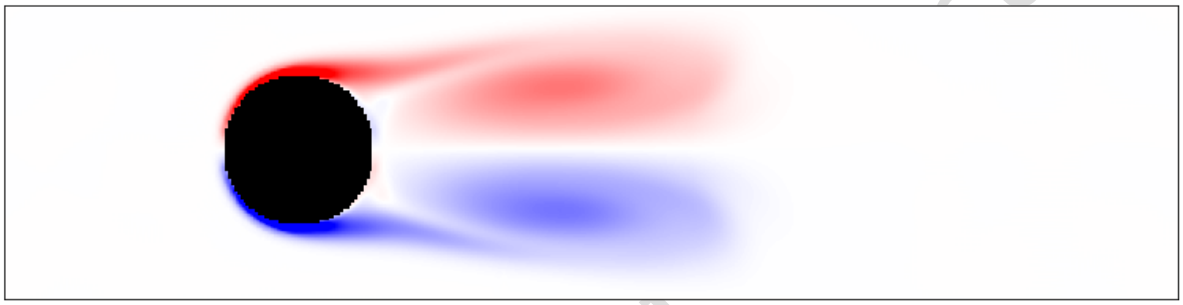
Because this simulation is written in Python, we can use the SmartRedis Python client to stream data to the database. To make the visualization easier, we use the SmartRedis Dataset object to hold two 2D NumPy arrays. A convenience function is provided to convert the fields into a dataset object. The outcome of the online analysis using SmartSim is listed below.



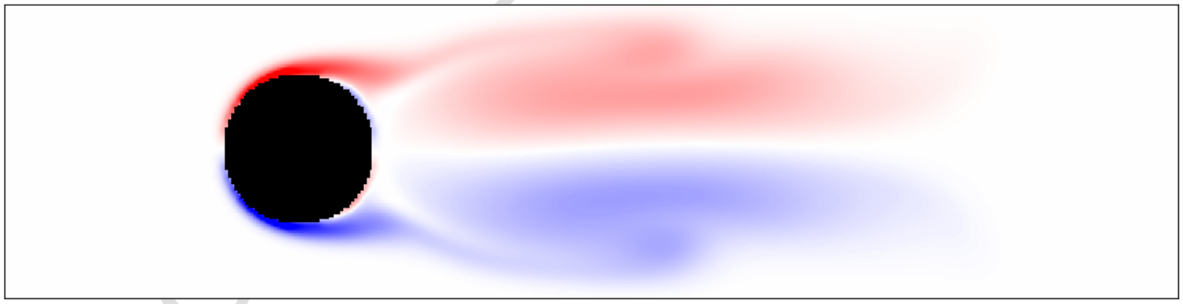
Vorticity plot at timestep: 0

HPE Confidential - Not for circulation

Vorticity plot at timestep: 700



Vorticity plot at timestep: 1400



Vorticity plot at timestep: 2100

HPE Confidential - Not for Circulation

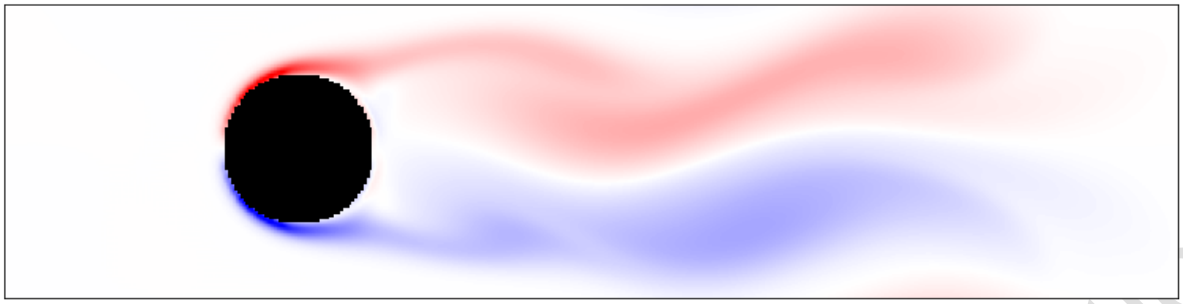


Figure 3.34 Online Analysis output

HPE Official - Not for circulation

MODULE 4

AUGMENTED AI & NATURAL LANGUAGE PROCESSING

HPE Official - Not for circulation

4.1 Augmented AI

Virtual assistants do not make decisions for you. Instead, they provide the data you need when you need it. If you have ever used Siri, Alexa, or another virtual assistant, you have used augmented intelligence. Augmented AI enhances AI functionality by combining the power of machine intelligence with human intelligence. Augmented AI is also called Intelligence Amplification (IA) due to this. The AI tries to replace humans with the machine. Instead, Augmented Intelligence tries to coexist with humans. Machines have an excellent ability to ingest large volumes of data, and their ability to carry out repetitive work is very high. Whereas the human's ability to generalize the knowledge he gathered is irreplaceable. Also, humans, creativity, and emotional intelligence are other attributes that we cannot replace using machines. One of the significant areas where the machine fails is exception handling. For example, a self-driving car cannot be operated well under severe weather or in a traffic emergency; instead, the human operator needs to take over. So, it is intuitive that combining virtues of both can make better technology. Augmented AI is an attempt in this direction

Augment itself means adding more value or size to anything. So here, adding value to AI is necessary as AI behaves like humans but performs mistakes as humans do. So, to improve the accuracy of AI systems where it is lacking, we go for Augmented AI. We know AI finds variation in massive data or anomalies in the data, but its decision-making ability is not comparable with humans. The primary thing missing in AI's decision is empathy. This can be well explained in the case of autonomous cars. Let us consider a hypothetical situation where a baby is on the road. As humans, we do not make rational decisions but think with empathy, morality, and ethics to stop the car and reach out for help. AI cannot inherit such property of empathy. Hence as humans, we trust the AI system when we know humans supervise them.

Another reason for moving towards IA is the Return on Investments (ROI). Many models developed are not implemented because Machines can never replace humans. However, with Augmented Intelligence, human capability can be enhanced by leveraging ML decisions. Thus, by combining human intelligence and artificial intelligence, it becomes

HPE Official - Not for circulation

superintelligence. For example, humans can use machine intelligence to perform the enormous task of analysing big data.

The changes needed to implement Augmented AI are minimal as we need to bring a few changes to the workflow, which aids in immediate ROI. This can be well visualized in the case of autonomous cars. Companies have started looking at augmented cars that enable few features such as self-parking and lane control than releasing fully autonomous cars.

4.1.2 How Augmented AI is working

Augmented AI applications will empower human abilities. Consider the same example of a self-driving car. The augmented AI system will update the traffic congestion data in all viable alternative routes to the user, and the user will decide which route to choose. Natural Language Processing (NLP) is inevitable for every AAI system. NLP is beneficial for the interaction between human intelligence and machine intelligence.

Contrary to the traditional view of AI as an autonomous system that operates without human interaction, augmented intelligence provides humans with actionable data based on machine learning and deep learning. The ability of an AI system to learn and improve from experience without programming is described as machine learning. An example of machine learning is natural language processing, allowing computers to recognize human speech. Deep learning, also known as deep neural learning, is an AI method of processing data and identifying patterns that mimic the brain's ability to see patterns, much to the delight of data scientists staring at vast datasets.

4.1.3 Solution to Big Data problem

Datasets with immense, complex, and fast velocity are big data. For AI's decision-making to evolve, big data is essential. By collecting and analysing big data, information and insights can be gained. Big data analytics aims to identify patterns and develop actionable insights through processes and technologies, such as AI and machine learning. Data-driven decisions enable you to take faster, better decisions, resulting in greater efficiency, revenue, and profit. I can increase the amount of data in the dataset, increasing the prediction accuracy. So, scientists have already started applying augmentation

HPE Official - Not for circulation

techniques to generate new data with the existing dataset. For example, consider a situation where you have images of 100 tigers. However, for good training, you need more images. So, a simple data augmentation technique such as rotation, cropping, zooming, etc., can be applied to increase the images in the dataset. So, we overcome the overfitting scenario of a Machine Learning model. Data augmentation can also be done by changing the noise level in the image. It is applied to images and any form of data such as Text, Speech.

Augmented has the upper hand when it comes to Big Data Analytics. They play a vital role in getting the insights of data where it reduces the time taken for cleaning data and allows more time in analysing, visualizing, and recommending suggestions in the Extract, Transform and Load (ETL) process.

4.1.4 Difference with AI

Even though artificial intelligence and augmented intelligence are often used interchangeably, the procedural difference between these two is distinct. The main difference between AI and IA is that AI replaces humans, but IA is to aid humans. AI works efficiently when dealing with Natural Language Processing, Speech Translators. However, when it comes to handling big data analytics and ethics-related decision-making, IA will perform well. For example, consider a health care system. A physician is assisted by augmented intelligence rather than having the assignment performed for them, as with artificial intelligence. Despite augmented intelligence allowing for collaborative approaches between doctors and machines, it does not lighten physicians' load as much as artificial intelligence. With the help of IA, it will be easier for physicians to deliver quality care to their patients rather than dealing with large data sets or electronic health records.

4.2 Basics of Natural Language Processing

Natural Language Processing is a subfield of Artificial Intelligence. Natural language processing is a field of study that deals with how computers and humans interact in their natural language. It combines computer science, machine learning, and linguistics. Just like how human beings interact, computers should understand the text and spoken words.

HPE Official - Not for circulation

Various algorithms are used to manipulate and interpret human language. Some NLP examples are sentiment analysis, entity extraction translation, and speech recognition. NLP enables the extraction of structured data from natural language. Machine learning algorithms use this raw data to make predictions.

In any communication (whether verbal or written), vast amounts of information are conveyed. Everything we do adds information that can be interpreted and value extracted. Our topics, tone, and choice of words all contribute to this. We can use this information to understand human behavior and even predict it. These unstructured data types are generated by conversations, declarations, and even tweets. Most of the data available in the real world are unstructured data since it does not fit neatly into relational databases' traditional rows and columns. It is messy and challenging to work with. It is scaling to many industries, a discipline that emphasizes the intersection of data science and human language. With today's advancements in data access and computational power, natural language processing (NLP) is experiencing a boom, allowing practitioners in fields such as healthcare, media, finance, and human resources to create meaningful outcomes.

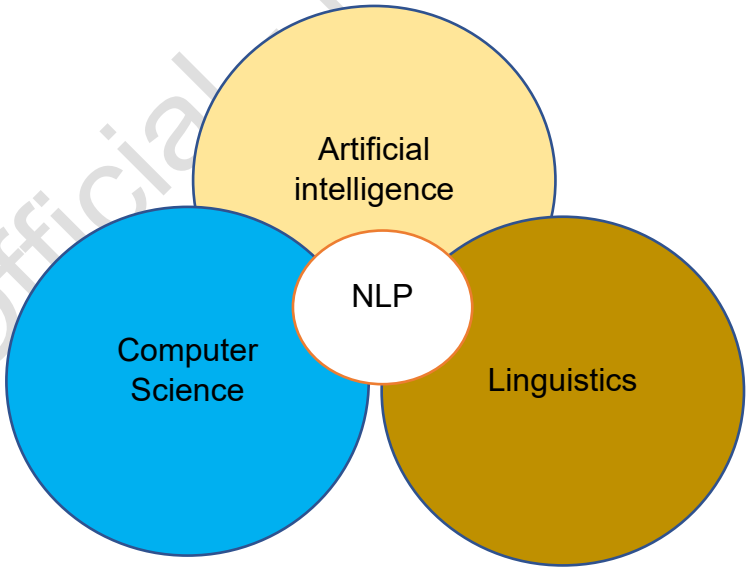


Figure 4.1 Natural Language Processing

HPE Official - Not for circulation

Natural Language Processing consists of two steps, Natural Language Understanding (NLU) and Natural Language Generation (NLG).

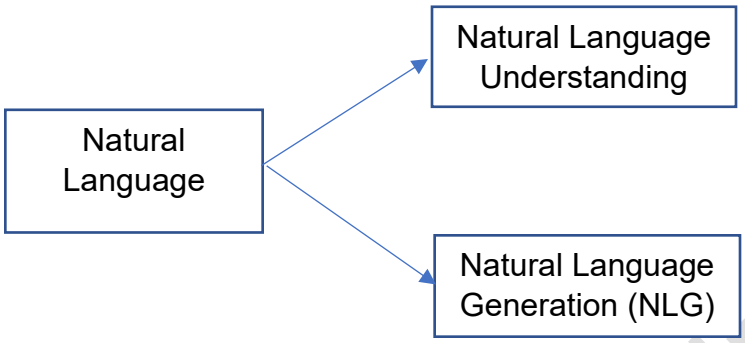


Figure 4.2 Steps of Natural Language Processing

Human language is understood and analyzed using NLU. The two tasks of NLU are generating a proper representation from the given input, and analysis is done on the language aspects. Natural Language Representation (NLG) translates the computerized representations into natural language. It involves text planning, sentence planning, and text realization.

1. Text Planning: -Relevant content is retrieved from the knowledge base.
2. Sentence Planning: -In order to form a proper sentence, the right words and phrases are chosen.
3. Text Realization: -The sentence plan is mapped to a sentence structure.

4.3 Steps Involved in Natural Language Processing

Currently, the main drawback of NLP is that languages are incredibly tricky. The process of understanding and manipulating language is overly complex, so it is common to use different techniques depending on the task before deciding which technique to use. The process of NLP is depicted in the figure below. The basic steps of Natural Language Processing are listed below.

- Morphological and Lexical Analysis

HPE Official - Not for circulation

- Syntactic Analysis
- Semantic Analysis
- Discourse Integration
- Pragmatic Analysis

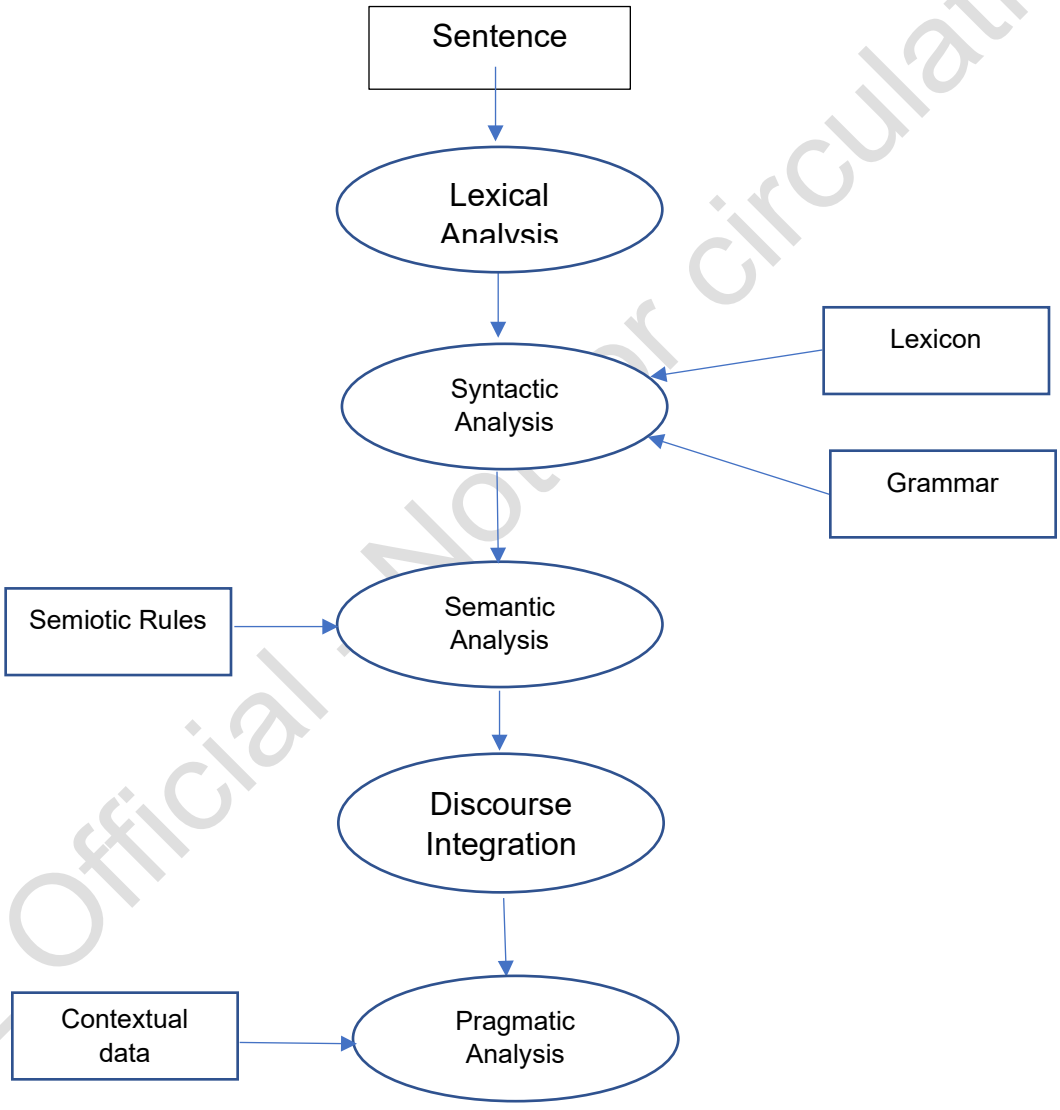


Figure 4.3. Components of NLP

HPE Official - Not for circulation

4.4 Morphological and Lexical Analysis

Morphological segmentation is the process of dividing words into individual units. In this phase, the data could be paragraphs, sentences, and words are broken into smaller tokens. For example, the exchange could be broken as 'ex-change.' In lexical analysis, the structure of words is identified and analyzed. The two standard methods are Lemmatization and stemming.

4.4.1 Tokenization

Tokenization involves splitting a text into parts called tokens and dropping certain characters, like punctuation marks, at the same time. It might seem like this in this case and in languages like English, where words are separated by a blank line (segmented languages), but it is different in many other languages.

4.4.2 Stop Words Removal

Common language articles, pronouns, and prepositions are removed from the text in the NLP process. This includes words such as "and" "the," or "to" in English. A potential approach is to adopt pre-defined stop words and later add words to the list if necessary. There is no universal list of stop words. They can be constructed from scratch or pre-selected. While the use of sizeable standard stop words lists has been declining over the past few years, no lists have been used. It is essential to know that removing stop words in each sentence can remove relevant information and modify the context in that sentence. For instance, removing a stop word such as "not" might throw our algorithm off if we perform sentiment analysis. You might select a minimal stop-word list and supplement it with additional terms depending on your objective.

HPE Official - Not for circulation

4.4.3 Stemming

Refers to the process of slicing the end of the beginning of words to remove affixes (lexical additions to the root of the word). Reducing the word to its base form is known as stemming. When used in English, prefixes must always be derivational (a prefix will create an unfamiliar word, such as "eco" in "ecosystem"). As in the case of the suffix "ist" in the word "guitarist" or the suffix "er" in the word "faster," suffixes can be derivational (create an unfamiliar word as in the case of the suffix "er" in the word "faster").

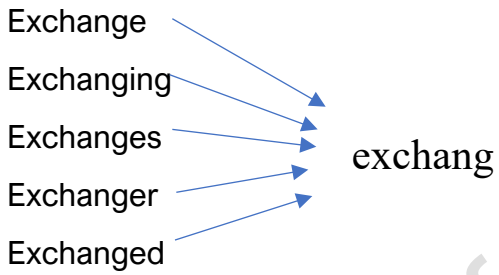


Figure 4.4. Stemming

4.4.4 Lemmatization:

Lemma means the base dictionary form of a word. Lemmatization is removing intonational endings to produce an accurate word from a dictionary. Words with the same meaning as their root are standardized by changing their past tense into the present (e.g., "went" is changed to "go"), and suitable synonyms (e.g., "best" is changed to "good"). While Lemmatization may seem like stemming, it uses a different methodology to uncover words' roots.

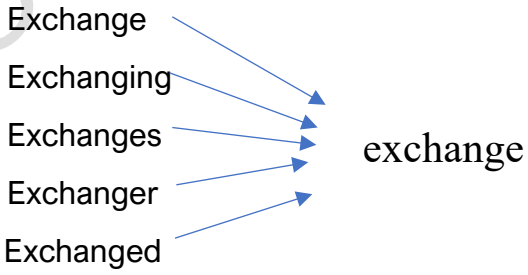


Figure 4.5. Lemmatization

HPE Official - Not for circulation

4.5 Syntactic Analysis

The arrangement of sentences properly to make a grammatical sense is known as syntax. The alignment of natural language with grammatical rules is assessed using syntactic analysis. The various syntax techniques are

- Word Segmentation: It creates distinct units from large pieces of continuous text.
- Part-of-speech tagging: For every word, parts-of-speech is identified.
- Parsing: Analysis of Grammatical correctness for the provided sentence.
- Sentence breaking: If it is a large piece of text, place sentence boundaries.

4.6 Semantic Analysis

This step checks for the meaningfulness of the given text. It checks for a semantically correct sentence and finds the sentence's meaning. Meaningful insights are retrieved from the text.

4.6.1Discourse Integration

There are many sentences. It would be meaningful only if there is no discontinuity between the preceding and succeeding sentences. Here the aim is to analyze whether these sentences are meaningful.

4.7 Pragmatic Analysis

Pragmatic analysis checks the references obtained in the previous phase(semantic) and correlates to the actual object/events in the given context. For example, "Place the ball on the floor in the basket" can have two semantic analyses, and one is chosen by a pragmatic analyzer depending upon the context.

The most common applications that use NLP are

- Translators like Google translate
- Conversion personal assistants like Siri, Cortana, Alexa, and Ok Google.
- I am responding to users' requests using Interactive Voice Response (IVR) applications.

HPE Official - Not for circulation

- NLP is used to check the grammatical correctness of sentences such as Microsoft Word and Grammarly.

4.8 Language Understanding Intelligence

Intelligence in language understanding means that the machine should understand the utterance and convert it into meaningful representation in machine-readable form. It should indicate the intent and any entities present in the utterance. Consider an intelligent language understanding dialog system embedded in the smart watch. For example, consider that the dialog system in a smart watch should check the body mass index and other parameters while exercising. The system recognizes intents like StartActivity and StopActivity related to ActivityType. However, if the person utters "Start yoga," the system should recognize the intent as Start Activity and entity as Activity Type='yoga'. So, an Intelligent system should be trained to understand all utterances, which are possible by proper training and deployment. Thus, using AI, an intelligent language understanding model can be built.

HPE Official - Not for circulation

4.9 Virtual customer service assistance based on natural language processing

It is essential to keep their customers happy in any business, which could help in business growth. Customers expect a quality product and high satisfaction. Every customer expects an answer to their questions and a solution to problems. A 24x7 support must be provided to their customers to handle feedback and reviews. This requires many employees to overwork which could affect their productivity.

Furthermore, small business models cannot afford to divert too many customer care employees. Virtual customer service assistants can handle routine customer care issues, which could help employees improve productivity. Virtual assistants are intelligent agents that can perform tasks or services based on commands for an individual. This helps customers to make correct use of the product cost-effectively. It includes installing, maintenance, troubleshooting, upgrading, and disposal of a product. The virtual assistant can handle customer support services like email queries, Live chat, and Tech support.

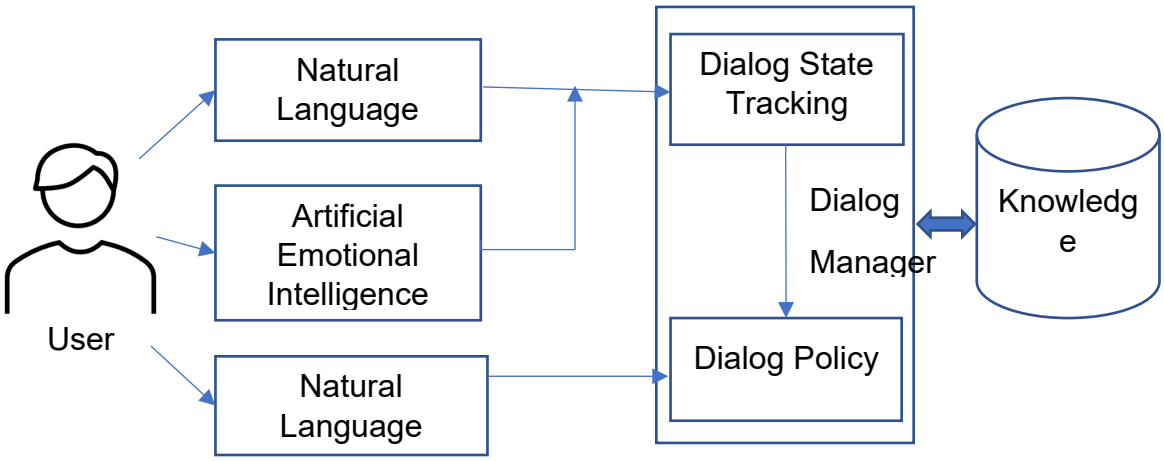


Figure 4.6. Virtual assistant Architecture

4.10 Speech Recognition and Synthesis

Converting human sound signals to words or instructions is known as speech recognition.

Not for circulation

The speech recognition algorithms consist of feature extraction, acoustic, language, and unique algorithms. The speech signals that are collected are sent to feature extraction. The speech features that are obtained are sent to the model library module. According to the model library, the speech pattern matching module identifies speech segments and obtains the result. Speech Recognition allows to convert speech signals to text or commands through identification and understanding and makes the purpose of natural voice communication.

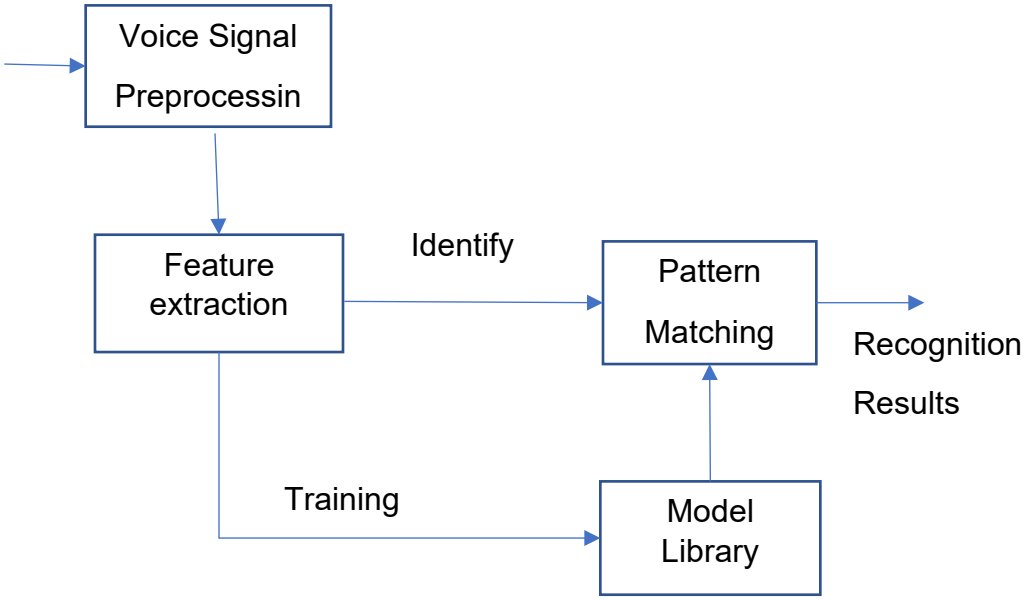


Figure 4.7: Basic principle of Speech Recognition system

The various speech recognition methods include the Hidden Markov model (HMM), vector quantization (VQ), support vector machine (SVM), and artificial neural networks (ANN). Let us investigate a few crucial definitions of speech processing.

4.10.1 Phoneme

An instance of a phoneme makes a word's pronunciation and meaning distinctive from another word. For instance, the /s/ in 'soar' separates it from the /r/ in 'roar,' as it differs in both pronunciation and meaning from 'soar.' The smallest unit of sound that distinguishes one word's pronunciation and meaning from another is called a phoneme.

4.10.2 Prosody

In addition to providing meaning beyond words, prosody also provides semantic information. For example, when describing an upward motion, speakers naturally raise the pitch of their voice.

4.10.3 Mel-spectrogram

The dimensionality of audio's short-time Fourier transform (STFT) is reduced by applying a non-linear transformation to the frequency axis. This method emphasizes low-frequency details that are crucial for distinguishing speech from noise de-emphasizes high-frequency details.

4.11 Speech Synthesis

Speech synthesis is the production of human speech artificially. It is a text-to-speech converter, whereas speech recognition is a speech-to-text converter. It is a technology used for intelligent speech interaction systems. The most prominent approaches are Concatenation synthesis and parametric synthesis.

As the name indicates, concatenation synthesis is constructed on prerecorded speech segment concatenation. In this, segments are stored in the form of waveforms or spectrograms. Speech Recognitions are used to acquire segments and are labeled using acoustic properties (fundamental frequency). The best chain of candidate units is selected from the database to create the desired sequence. Recorded human voices are used in parametric synthesis. The voice is modified using a set of functions and parameters. The two parts of statistical parametric synthesis are training and synthesis. The audio sample characteristics such as fundamental frequency (voice source), frequency spectrum (vocal tract), and duration(prosody)of speech are extracted during training. A statistical model like Hidden Markov Model (HMM) is later used to estimate those parameters. A set of parameters is generated using HMM's from the target text sequence. The final speech is synthesized from these parameters.

Deep learning-based methods can improvise the quality of speech generated. The input text is passed to an acoustic feature generator in these models. This generator generates

HPE Official - Not for circulation

acoustic features like spectrogram or fundamental frequency. A neural vocoder is used to generate the final speech segment. In neural vocoder, the encoding and decoding of audio signals is done by a neural network

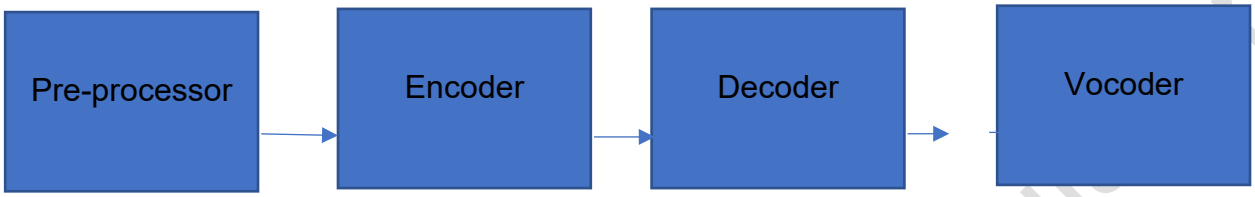


Figure 4.8 Speech Synthesis

4.11.1 Preprocessor

The preprocessing steps involved are following

1. Tokenize: Tokenize a sentence into words
2. Phonemes/Pronunciation: Text input is subdivided into phonemes based on pronounced. For example, "Hello, have a good day" is converted into HH AH0 L OW1, HH AE1 V AH0 G UH1 D D EY1.
3. Phoneme duration: Each phoneme in the audio is represented by the total time it takes.
4. Pitch: Speech prosody is primarily affected emotions are conveyed.
5. Energy: Mel-spectrogram magnitude indicates the frame-level volume of speech and directly affects its prosody.

4.11.2 Encoder

Latent features are crucial because other features such as speaker embedding can be used without a latent feature. Linguistic features (Phonemes) are input to the encoder, which outputs an n-dimensional embedding. Additionally, the latent features are also used to predict energy, pitch, and duration, which are crucial to maintaining the naturalness of the audio.

HPE Official - Not for circulation

4.11.3 Decoder

To decode Latent processed features to Acoustic features, a decoder is needed. Because audio includes more variance information than Mel-spectrograms (e.g., phase), there is a more significant information gap between input and output for text-to-audio conversion than text-to-spectrogram conversion. Therefore, Mel-spectrogram conversion is preferred.

4.11.4 Vocoder

A mathematical model like Griffin Lim has converted the Acoustic feature (Mel-spectrogram) to waveform output (audio). We could also train a neural network to learn the mapping from model-spectrogram to waveforms. To efficiently perform this complex and sophisticated task, we split it into two stages. First, we predict Mel-spectrograms based on Latent Features, and then we generate audio using Mel-spectrograms.

4.12 ANN For Speech Processing

An artificial neural network (ANN) is analogous to the biological nervous system. It uses a parallelly connected substantial number of processing units to form a complex processing system.

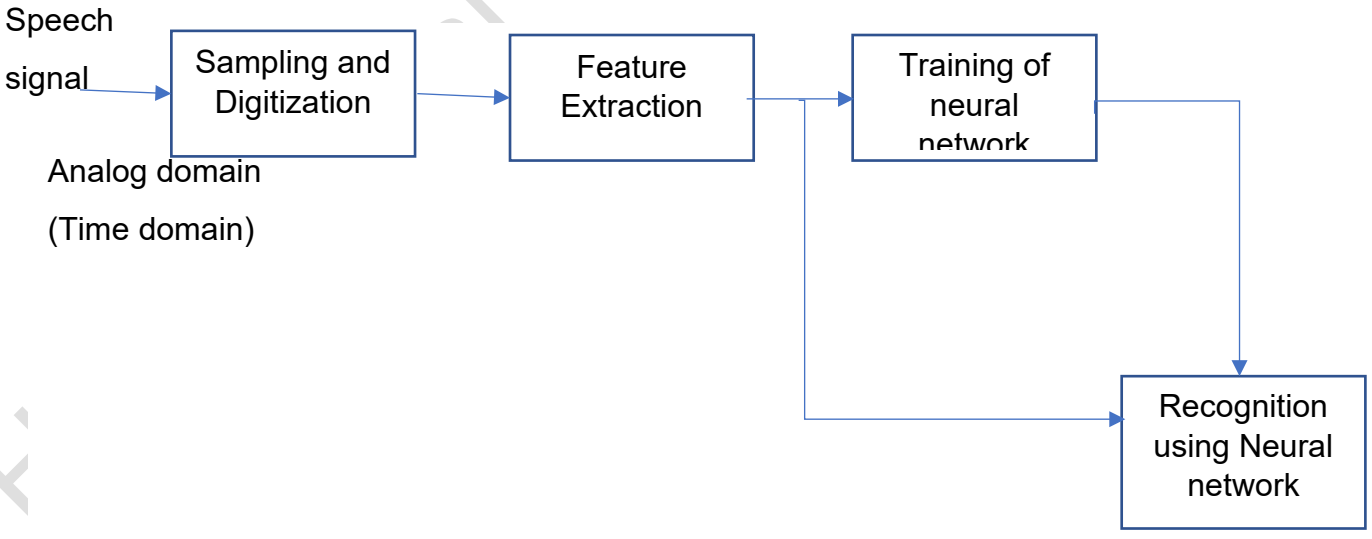


Figure 4.9. Speech Recognition model using neural network

Final - Not for circulation

The analog speech signal must be converted to digital for processing. The feature extractor must process the speech signal such that the classification process should generate output with no errors. Initially, the signal is sampled at frequencies twice greater than human speech signal frequency and converted to a digital signal. The digital signal is passed through a pre-emphasis filter, which is part of the feature extractor used to flatten the signal and make it less susceptible to finite precision effects later. Various windowing functions can be applied to the signal. Windowing minimizes the spectral distortion by tapering the signal to zero at the beginning and end of each frame. The features extracted can be used to train the neural network model. Once the model is trained, it can be used with the target dataset to recognize the speech

4.13 Use cases

4.13.1 Online stores using data analytics to predict customer preferences

This is a simple use case in day-to-day life where e-commerce utilizes ML models to predict user preferences based on previous purchases and searches made on their websites. Also, sentiment analysis can be made to boost their interest in buying. Such recommendations will make the customers get the same satisfaction they get in offline store shopping. The sales team can get insight into the products in high demand. Also, region-based analysis of products sold can be done. It also helps maintain the demand and supply chain in the retail business.

4.13.2 Medical analysis of case files to identify efficient treatment options

Analysing large datasets containing the history of patients helps suggest the best treatment for a particular patient. History can hold all personal details of a patient, the symptoms observed, treatment is given, etc. Holding such large datasets can help investigate a patient from anywhere globally and by different doctors. IA aids doctors in making decision faster and more accurate with the help of analytic data models. Also, it helps in discovering the chances of getting a particular disease in the future. Overuse of medicines can also be avoided using IA.

HPE Official – Not for circulation