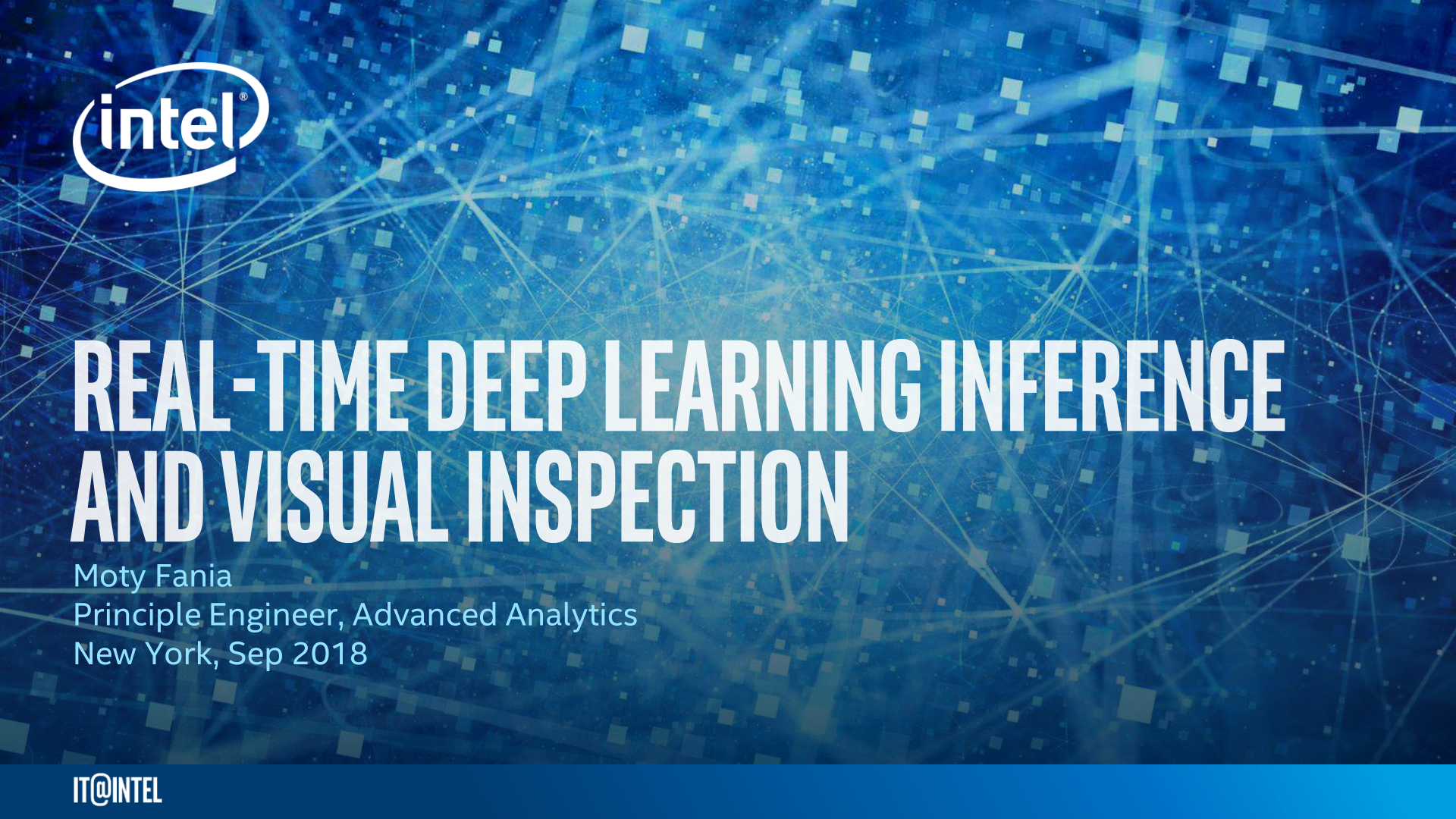

intel
REAL-TIME DEEP LEARNING INFERENCE AND VISUAL INSPECTION
Moty Fania
Principle Engineer, Advanced Analytics
New York, Sep 2018
IT@INTEL

Legal notices

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

Agenda

Advanced Analytics @ Intel

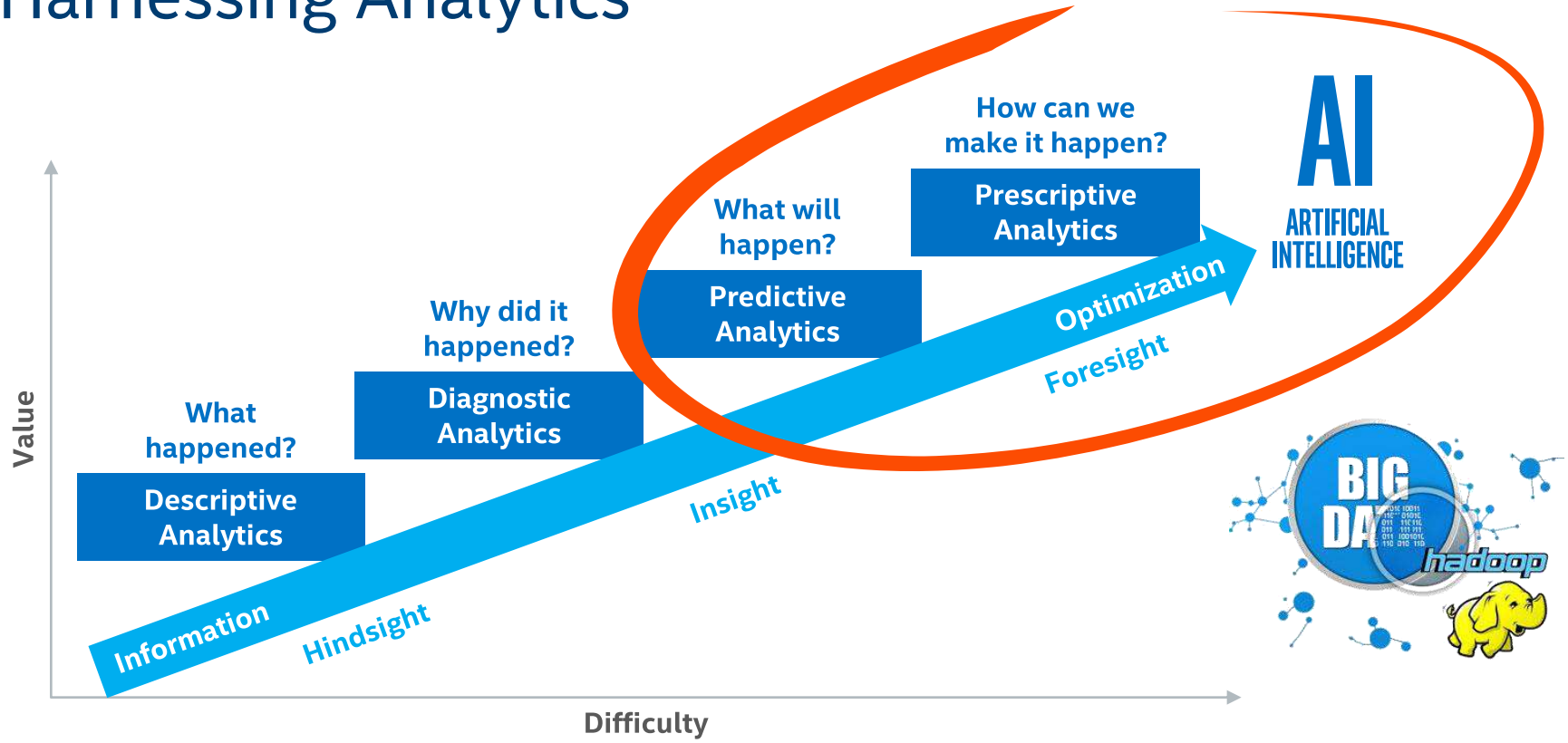
Corporate AI concepts

Deep learning Inference system

Visual inspection

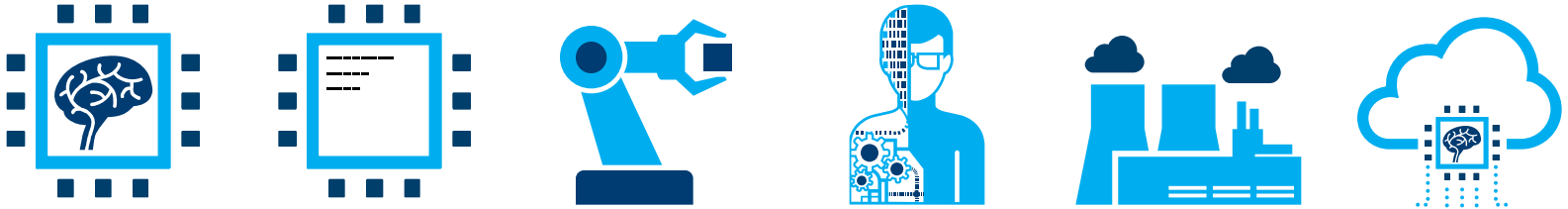
- Use cases
- Architecture

Harnessing Analytics



Source: Gartner

About Us – Advanced Analytics @ Intel



	EMBED LEARNING	HW VALIDATION	PRODUCT DEV	SALES	INDUSTRIAL IOT	HEALTH
VALUE	Improve products power/ performance	Cut product time to market	Reduce test cost and improve quality	Increase revenue	Reduce Manufacturing cost	Improve clinical trails outcome
HOW	Adaptive & personalized HW	Automated validation with Context	Personalized units testing	Autonomous accounts coverage	Proactive actuation to changes	Continuance Monitoring at home

VISION : PUT AI TO WORK FOR HUMAN EXPERTS

AI in a Nutshell

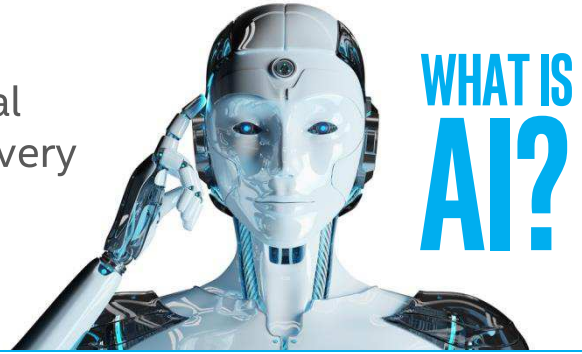
Artificial intelligence is about **replacing human decision making with more sophisticated technologies.**

- These are not repetitive tasks, but rather judgment-based work
- Requires a more complex set of algorithms and machine learning which can use a variety of inputs to recognize patterns, predict future outcomes and make decisions.

Forbes

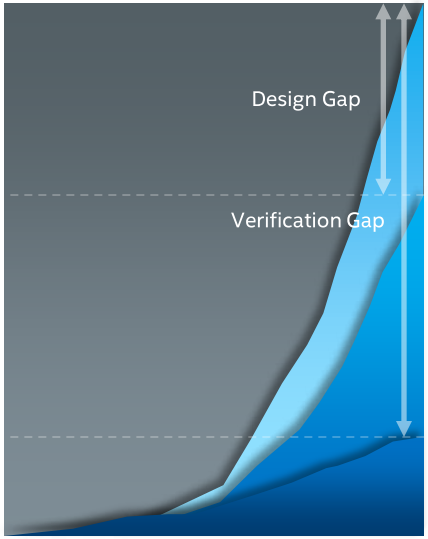
Narrow AI

Also referred to as “weak AI.” It is the only form of Artificial Intelligence achieved so far. This is AI that works within a very limited context, and can’t take on tasks beyond its field.



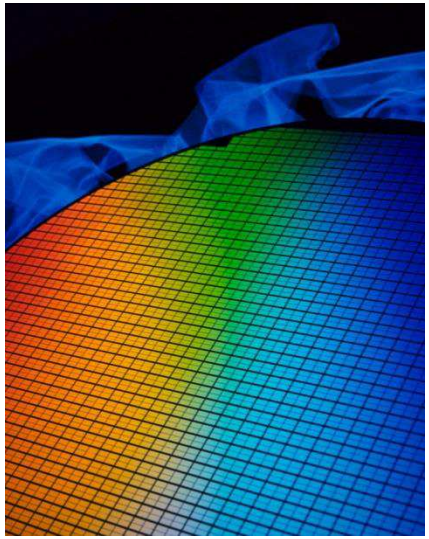
Corporate AI – “Narrow AI“ Use Cases

SALES



DESIGN

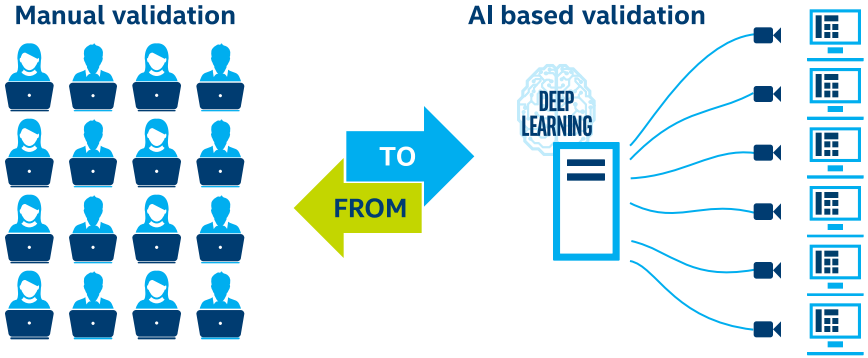
VISUAL INSPECTION



PRODUCT DEVELOPMENT

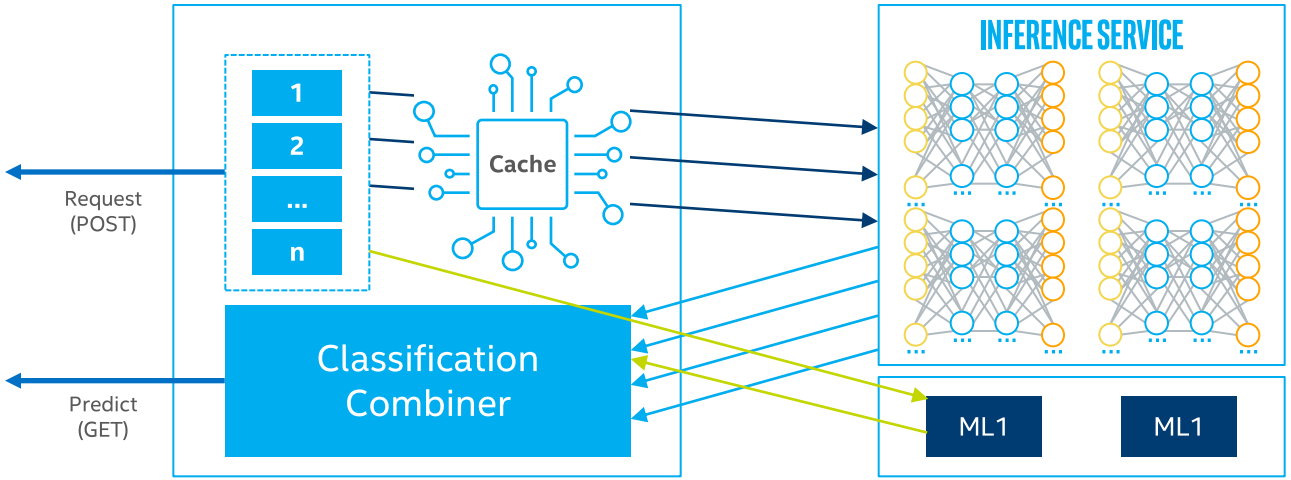
Our Visual Inspection Challenge

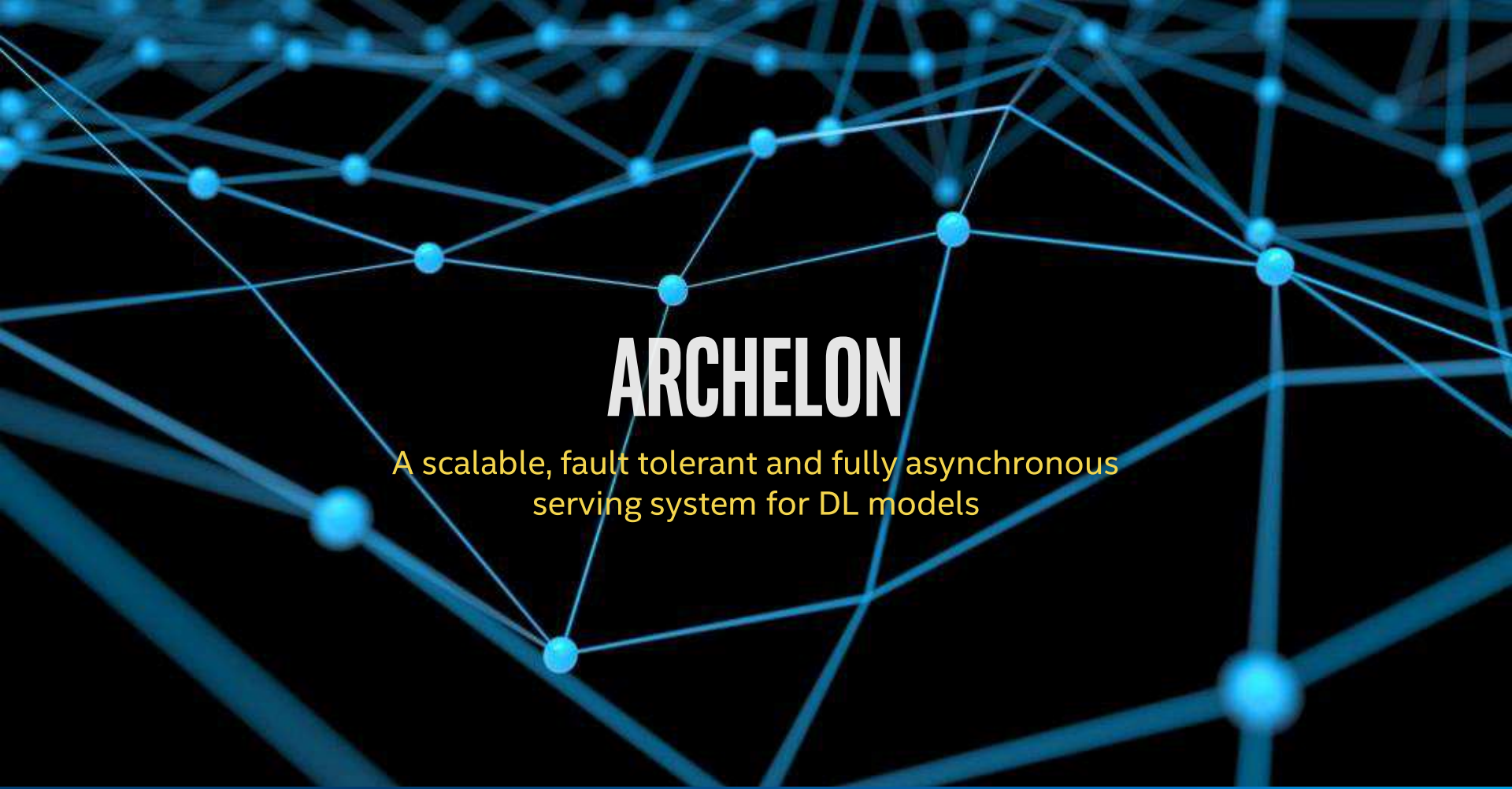
- Source: Full HD video streams from multiple cameras
- Multiclass classification problem – ~15 classes (multiple DL algorithms required)
- Inspection process will be running 24x7 - detect “issues” at frame level **online**
- Low tolerance for mistakes – very high precision & recall required
- Potential to scale out to hundreds of input cameras



Real Time DL Inference – What is Required?

- Production Inference service for DL models
- Smart in-memory cache for data batching , sequencing
- Fast, scalable APIs for data ingestion & real time responses
- Sync – Async calls
- Full Scalability

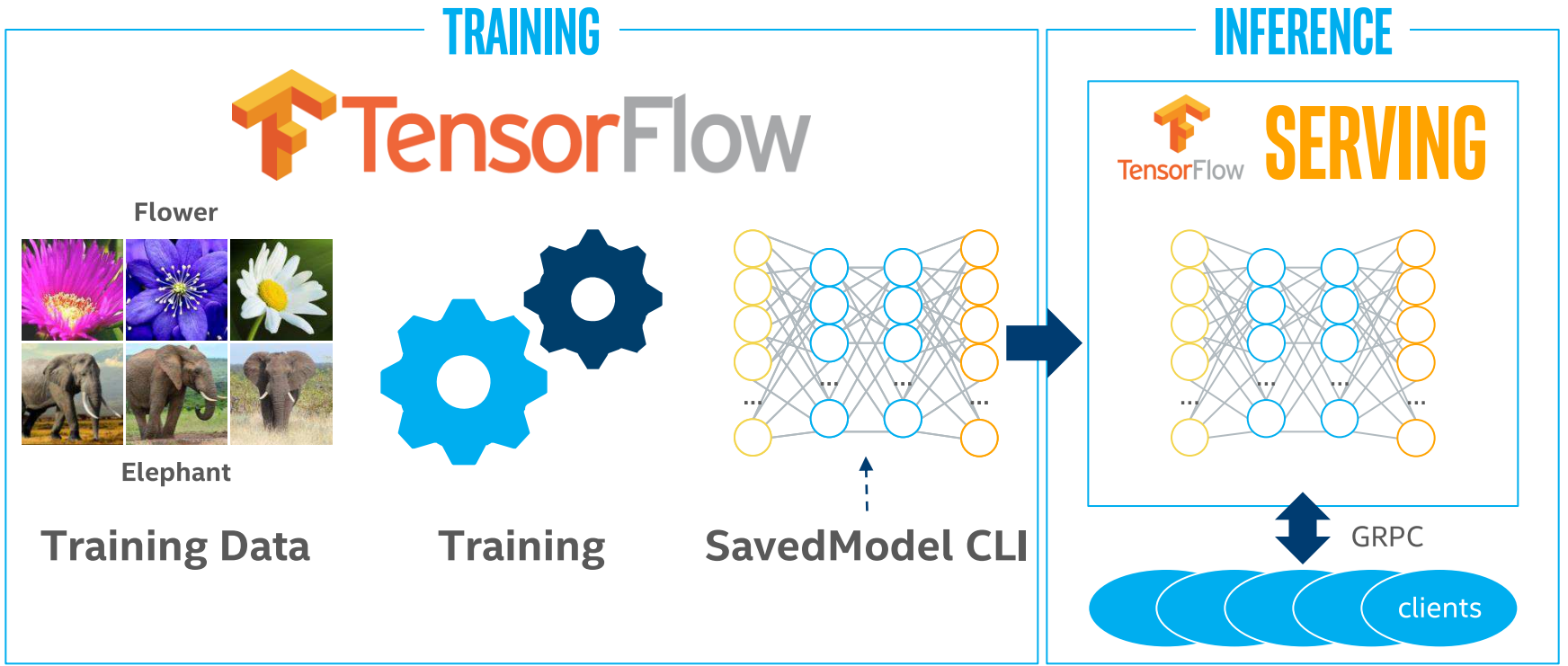



ARCHELON
A scalable, fault tolerant and fully asynchronous
serving system for DL models

MODEL SERVING

IT@INTEL

Deep Learning Lifecycle with TensorFlow



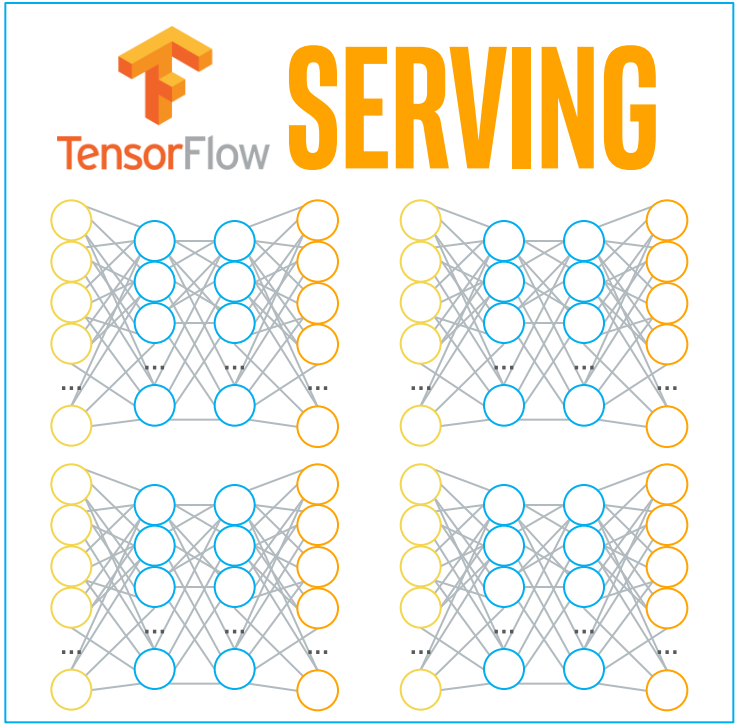
* Other names and brands may be claimed as the property of others.

Tensor Flow Serving

A flexible, high-performance serving system for machine learning models, designed for production environments

We have added the following capabilities:

- Simple APIs to deploy new models
- Generic client to query any model (sync / Async)
- Optimized Docker image for CPU & GPU
- Implementation within Kubernetes with performance optimizations & scaling
- Full automation of deployment



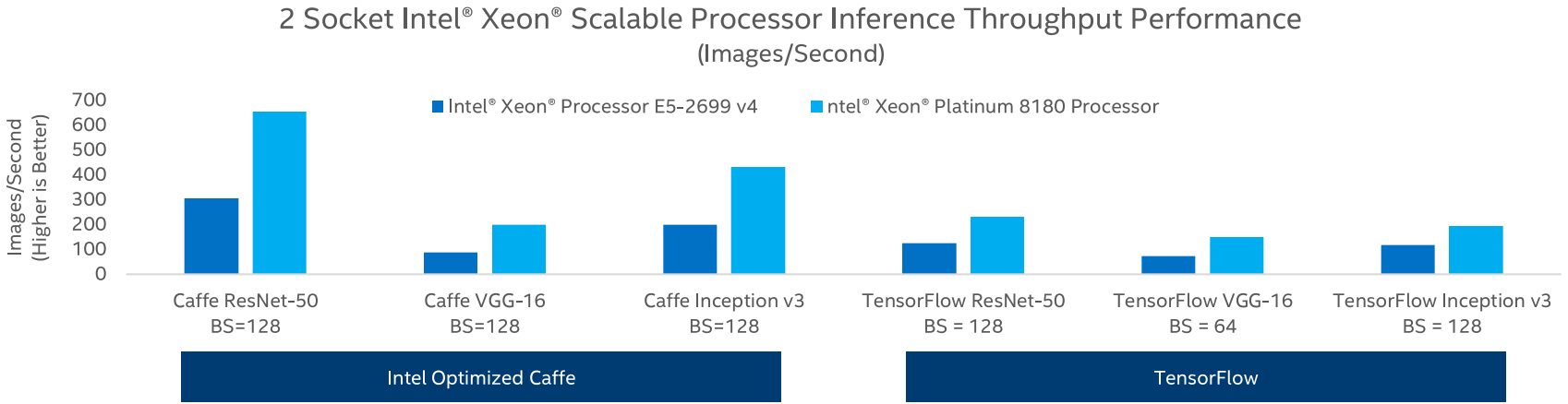
BATCHING & SEQUENCING

(The Data Store)

IT@INTEL

Batch Size for Optimal Inference Throughputs

- To maximize the utilization of inference HW and minimize its cost, we have to apply batches vs. single frame inference
- The goal is achieving the right balance between latency and throughput
- Batch size is determined while keeping latency under the given latency requirements.



Source: https://software.intel.com/en-us/articles/intel-processors-for-deep-learning-training

REDIS – Smart Batching & Sequencing (Data Store)

Redis is an open source **data structure store** that works with **in-memory datasets**

Used as a persistence database, cache and message broker

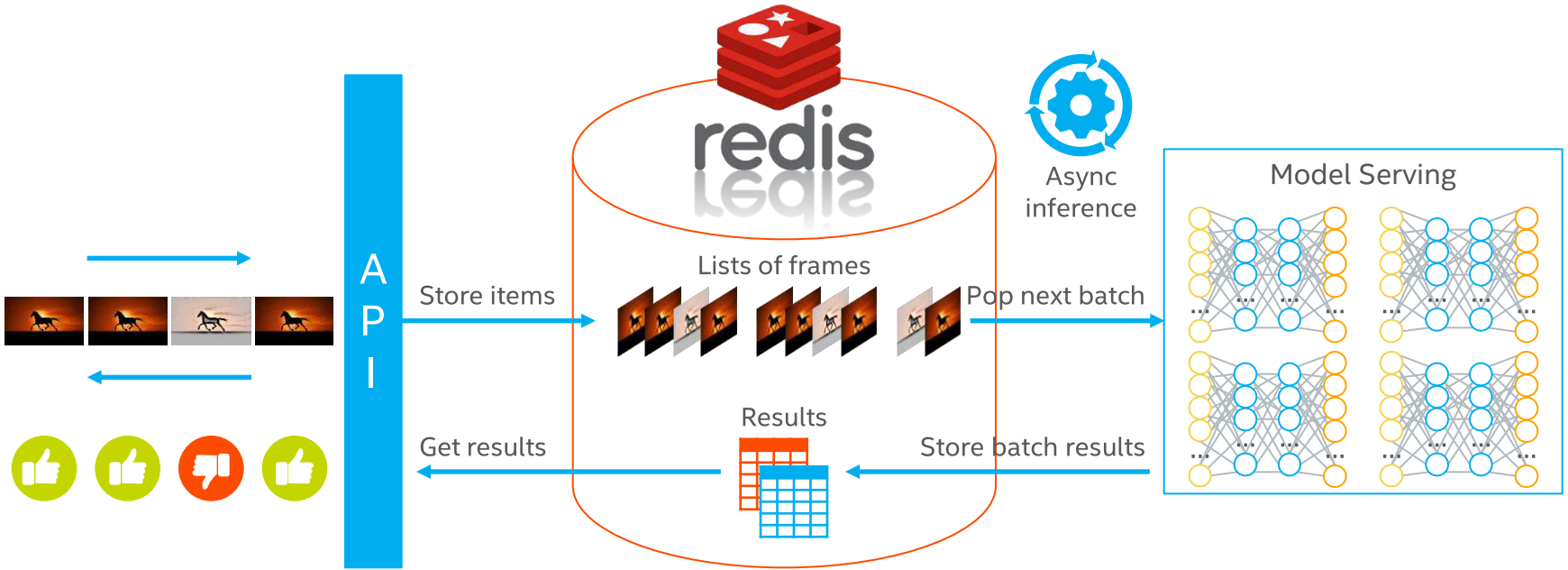
It supports data structures such as strings, hashes, lists, sets, sorted sets

Other features include:

- Transactions
- Pub/Sub
- Keys with a limited time-to-live
- Clustering

* Other names and brands may be claimed as the property of others.

Batch Management with REDIS

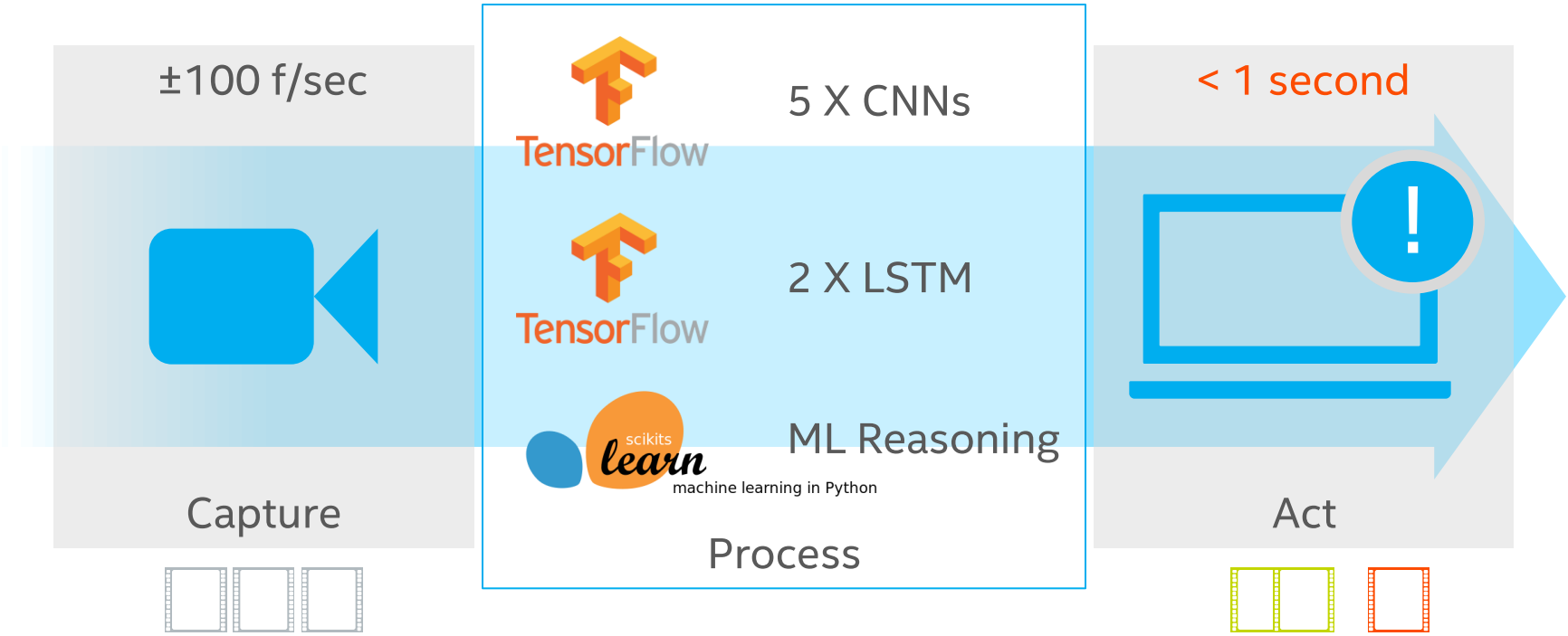


* Other names and brands may be claimed as the property of others.

ASYNCHRONOUS INFERENCE

IT@INTEL

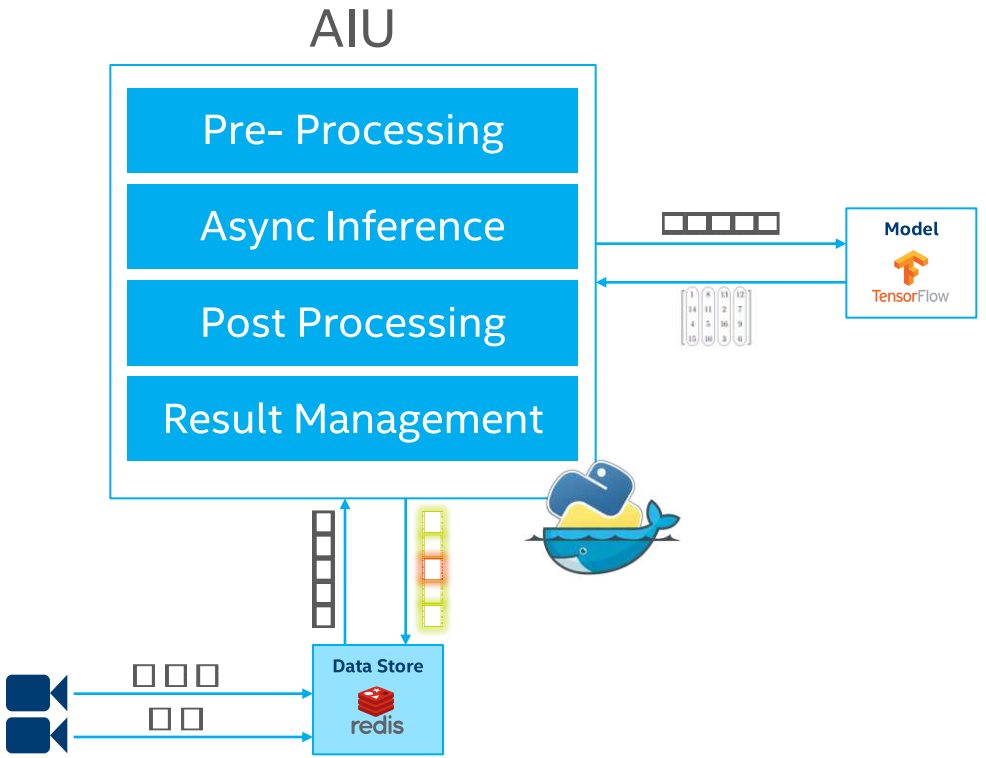
The Latency Challenge



* Other names and brands may be claimed as the property of others.

Asynchronous Inference Unit (AIU)

- Always ON
- Continuous resource utilization
- Stateless
- Dynamic Batching
- Logical grouping of Models – “Units”
- Easily scalable



* Other names and brands may be claimed as the property of others.

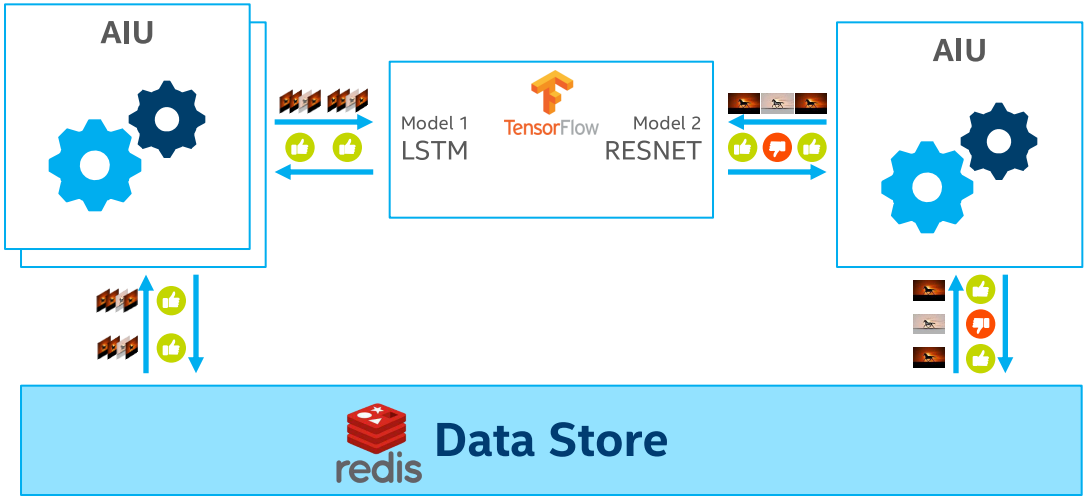
AIU Allocation

AIUs enable flexibility and Separation of concerns

Can be allocated as required

- per model
- per camera / source
- Latency / throughput
- Inference HW

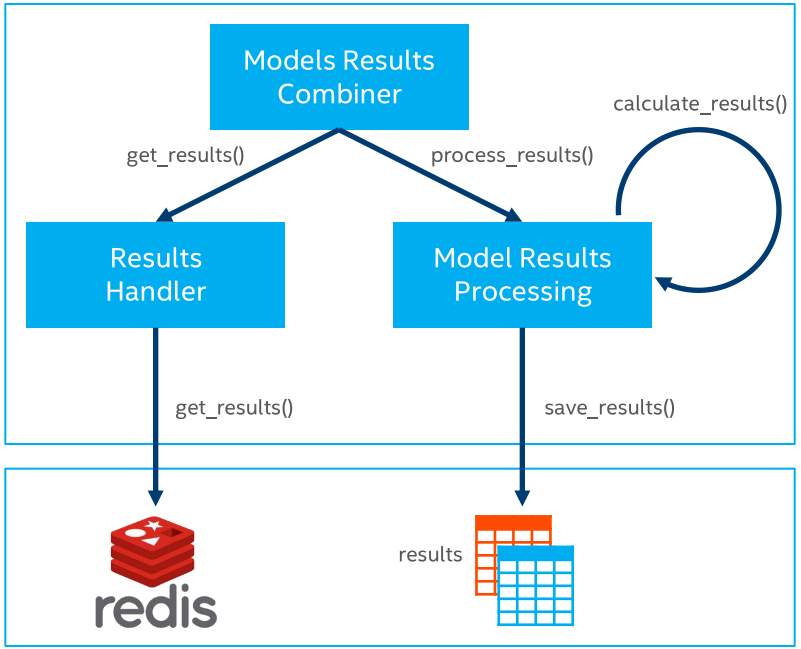
Example: Dedicate AIU per model

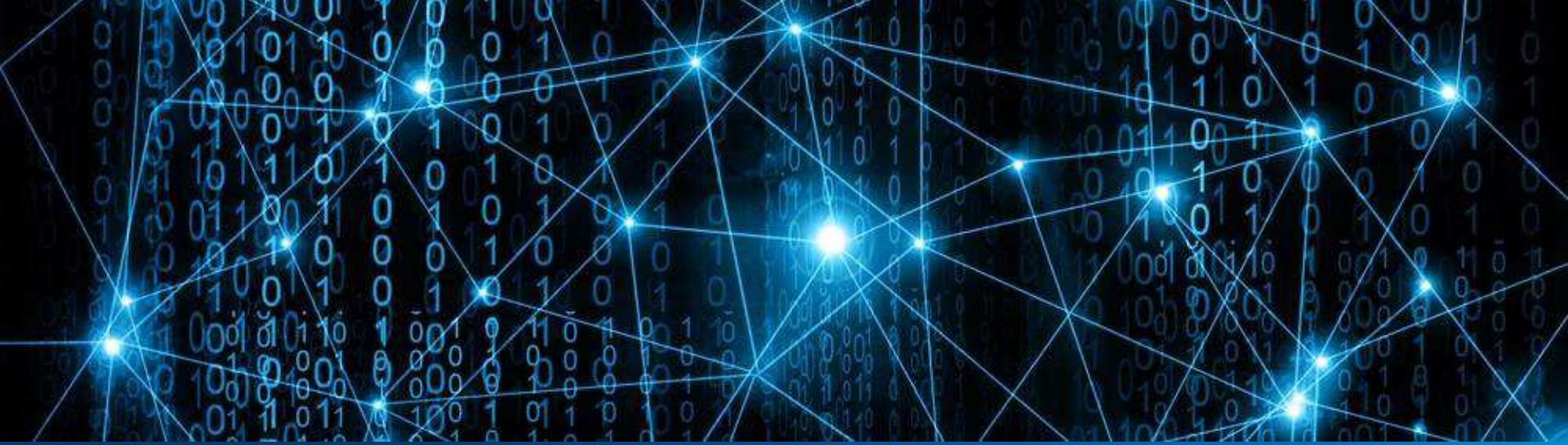


* Other names and brands may be claimed as the property of others.

Handling Compound Result

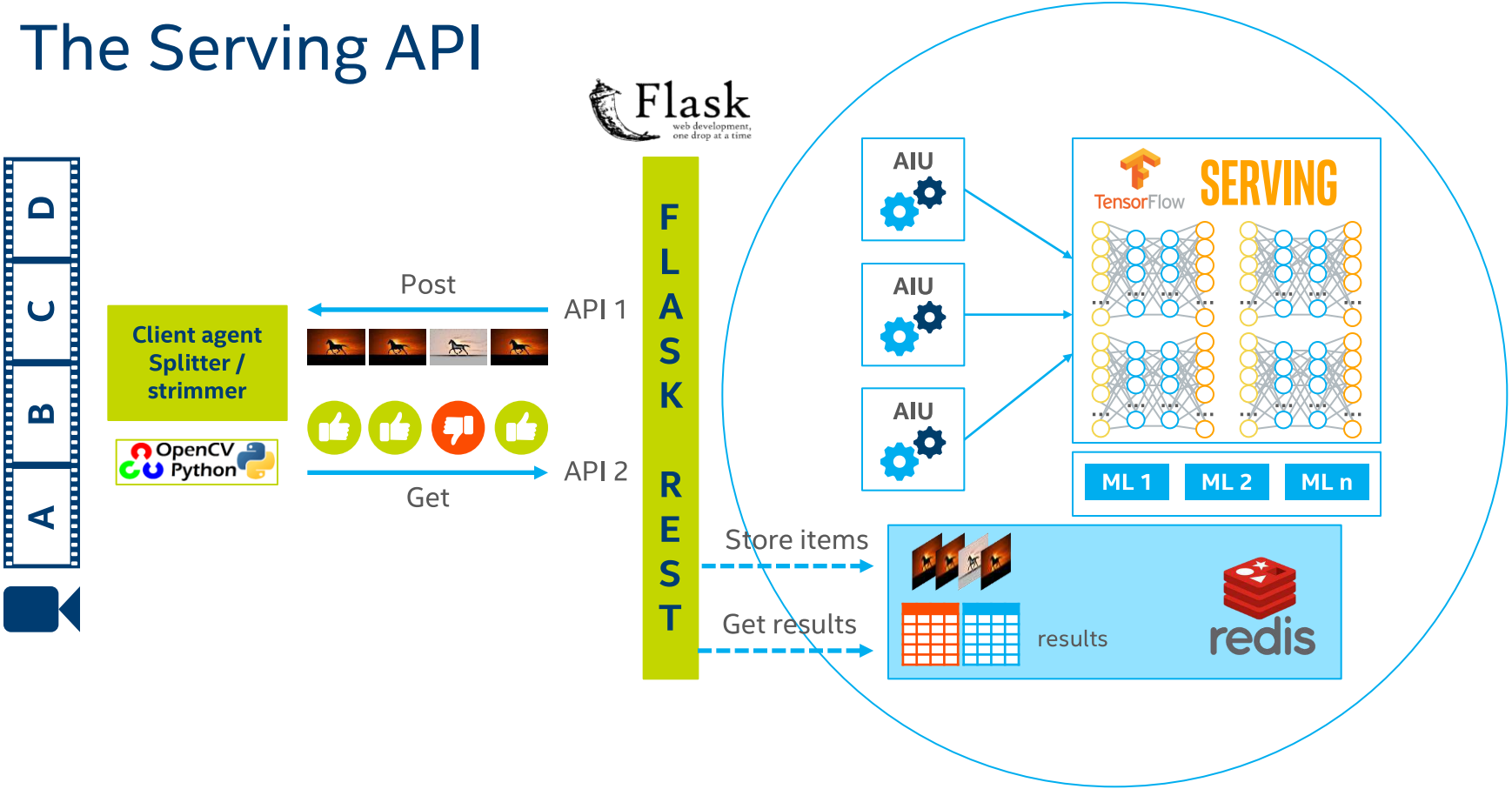
- The System supports compound results based on an ensemble of Ml / DL models
- Interim results are stored into Redis and a combiner process is responsible for applying the final logic





THE SERVING API

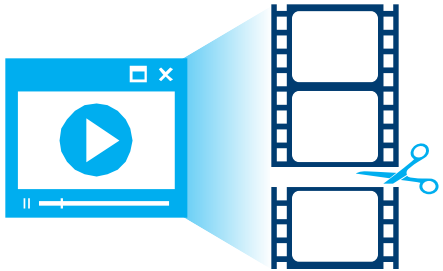
The Serving API



* Other names and brands may be claimed as the property of others.

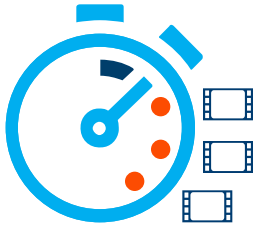
Logic in Client

Splitter / strimmer Agent



- Fetch next Frame
- **Call API 1** to Post Frame with metadata

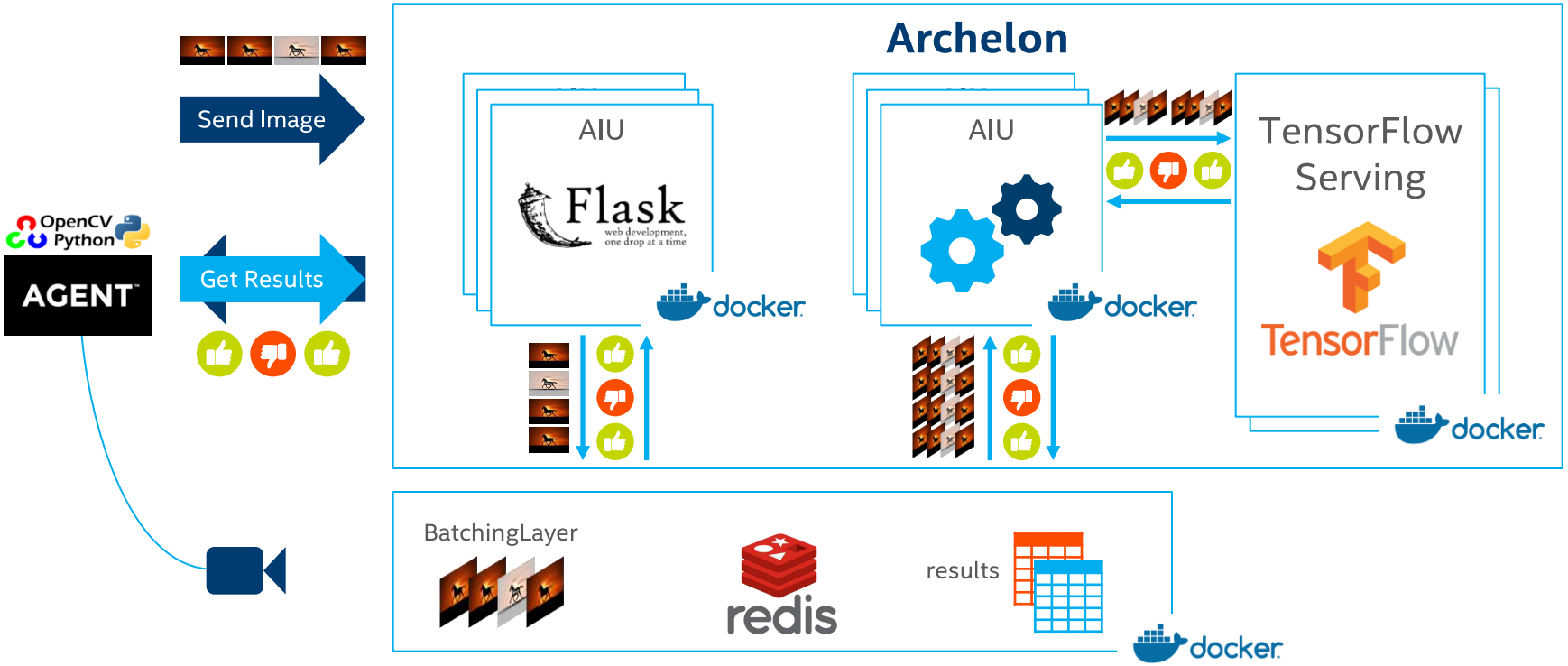
Result collector



- Sleep for X ms
- **Call API 2** – to get inference results for next set of processed frames

IMPLEMENTATION

System Flow Illustration



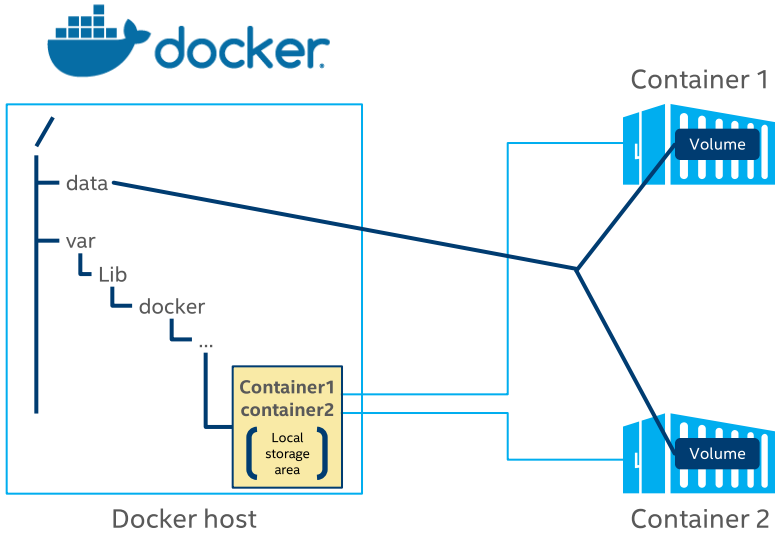
* Other names and brands may be claimed as the property of others.

Sample Implementation Project with AppBuilder

```
from archelon.builder.app_builder import AppBuilder
import SimpleModelUtil
import SimpleModelProcessingResultsLogic
import SimpleResultsLogic
import os

models_config = "%s/models_service.ini" % os.environ["CONF_FOLDER"]

with AppBuilder() as app_builder: app_builder\
      .add_model('DNN1', SimpleModelUtil, 'u1', 'g1') \ .add_model('DNN2', SimpleModelUtil, 'u1', 'g1') \
      .add_models_processing('u1', 'g1', SimpleModelProcessingResultsLogic, models_config) \
      .add_results_logic('u1', SimpleResultsLogic)

app = app_builder.build()
app.start()
```

Why Docker?

- Portability and ease of deployment anywhere while avoiding the dependency hell
- Docker guarantees that the software will always run the same, regardless of its environment.
- A unit of isolation (decoupled)
- Modularity and Scale Out

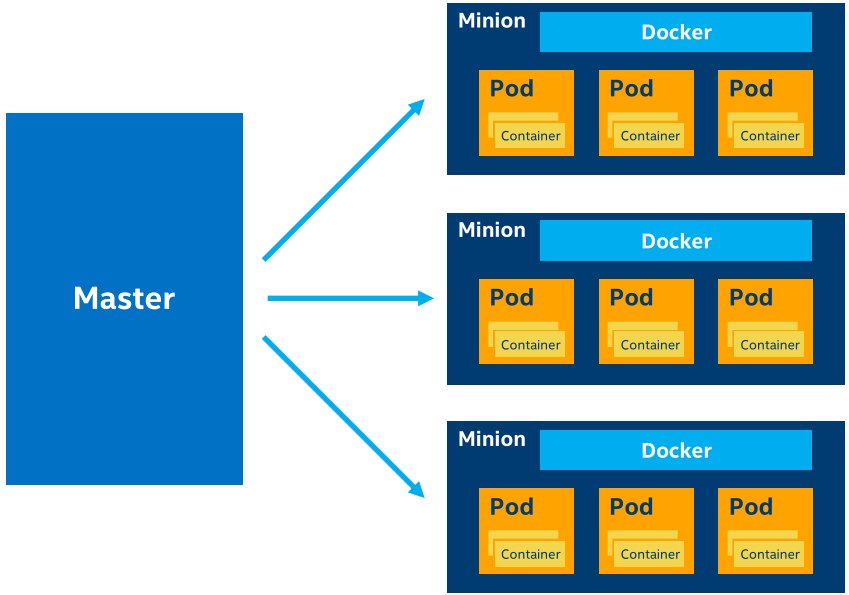


Units of software delivery (ship it!)

* Other names and brands may be claimed as the property of others.

Scale Out with Kubernetes

- An open-source system for automating deployment, scaling, and management of containerized applications.
- Groups containers that make up an application into logical units for easy management and discovery.
- Provides container grouping, load balancing, auto-healing, scaling features
- Progressively rolls out changes and updates without rebuilding images

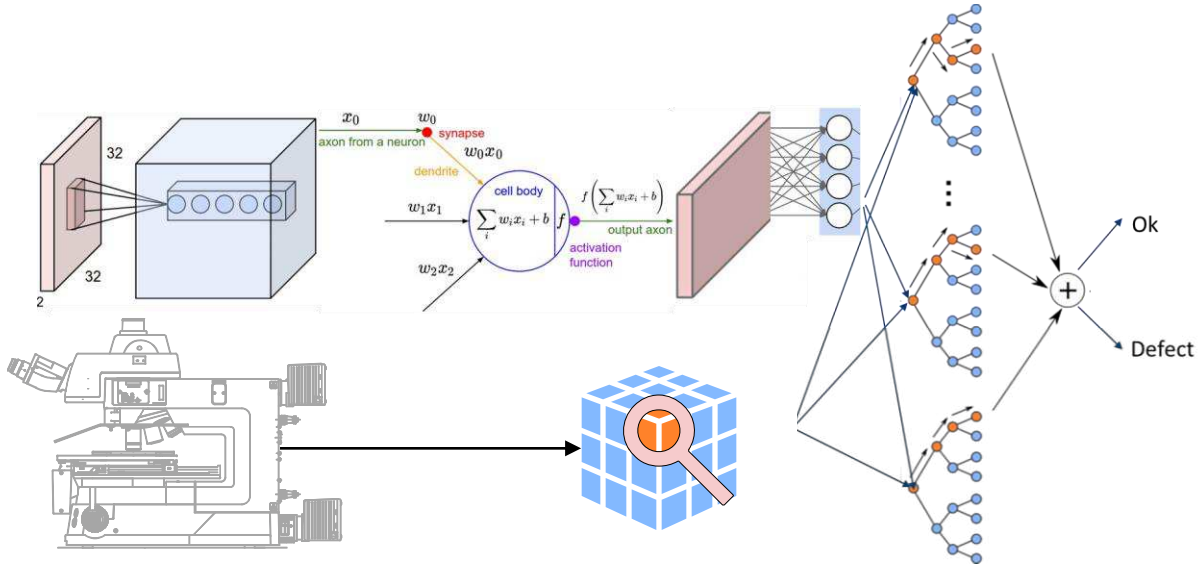


A FLEXIBLE CLUSTERING TECHNOLOGY FOR CONTAINER BASED PLATFORMS

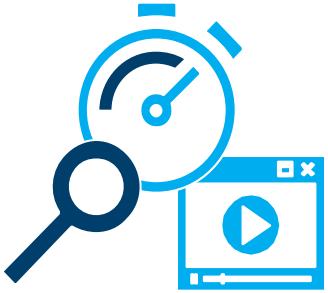
* Other names and brands may be claimed as the property of others.

Use Case 2 : Wafer Image Inspection

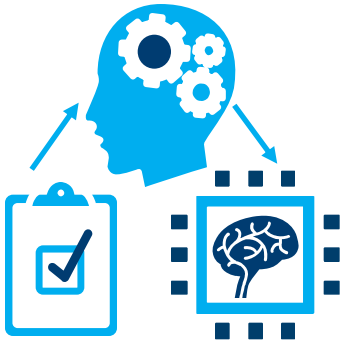
- Detecting wafer defects based on optic microscope images.
- Input consists of images of defects and reference images of wafers, as well as tool metadata.



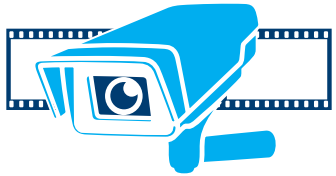
Other Potential Applications



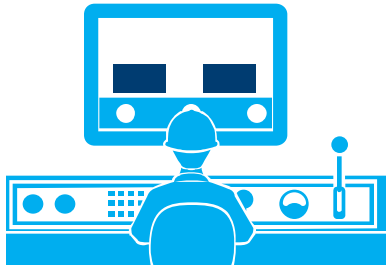
FAST SEARCH IN A VIDEO



REAL TIME TEXT ANALYTICS – SUMMARIZER, CLASSIFIER



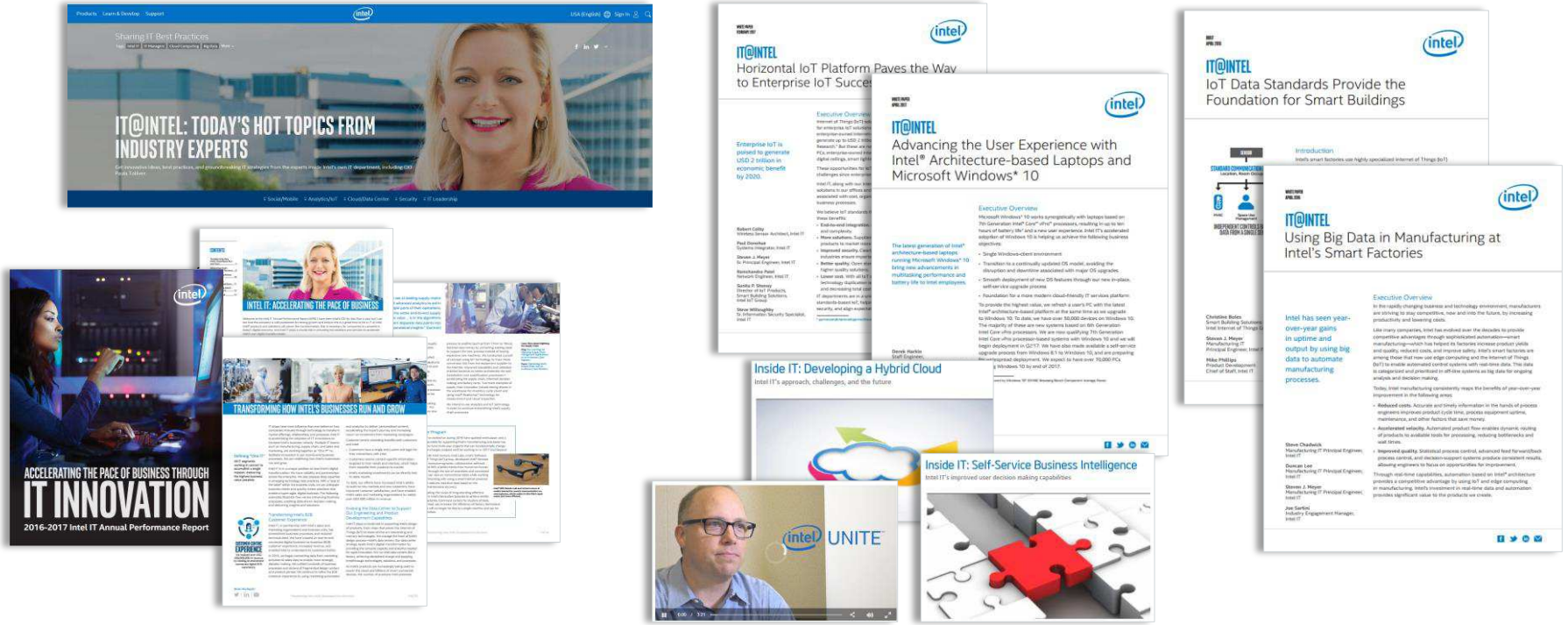
SECURITY AND SURVEILLANCE



INDUSTRIAL IOT USE CASES

DEMO

IT@INTEL: Sharing Intel IT best practices with the world



Learn more about Intel IT's initiatives at: **www.intel.com/IT**

THANK YOU!

ALT