Productizing Complex Visual AI Systems for Autonomous Flight

Carlo Dal Mutto, Director of Engineering
Project Wayfinder @ Acubed, Airbus' Innovation Center in Silicon Valley

May 25, 2021

2021 embedded VISION summit VIRTUAL | MAY 25-28

AN AIRBUS INNOVATION CENTER

2021 embedded vision summit VIRTUAL | MAY 25-28

Project Wayfinder:

Developing autonomous flight and machine learning solutions for the next generation of aircraft

AN AIRBUS INNOVATION CENTER


Project Wayfinder
Flightpath toward autonomy
WAYFINDER
WAYFINDER
WAYFINDER

Problem Statement

- Develop a multi-sensor system to deliver on autonomous flight (focus on landing)
- Development streams
  - Sensing development (cameras, radars, INSs)
  - SW development
  - Perception development (including ML)
- Target functions
  - Large-scale data collection
  - Development of perception functions

Challenges

- Perception development is complex and spans multiple years
- Multi-disciplinary interdependent development streams
- “Clean” data availability is a blocker for ML development
- Visual AI deployment is a bottleneck in the product development process, even though it is often considered an “automated process”

Productizing Visual AI System

Which Systems?

- Real-time perception systems
  - Inference pipelines with visual AI and classical computer vision (CV) modules
- Examples:
  - Autonomous driving/flying systems
  - Visual inspection tasks (manufacturing, logistics, etc.)

AI Model Design

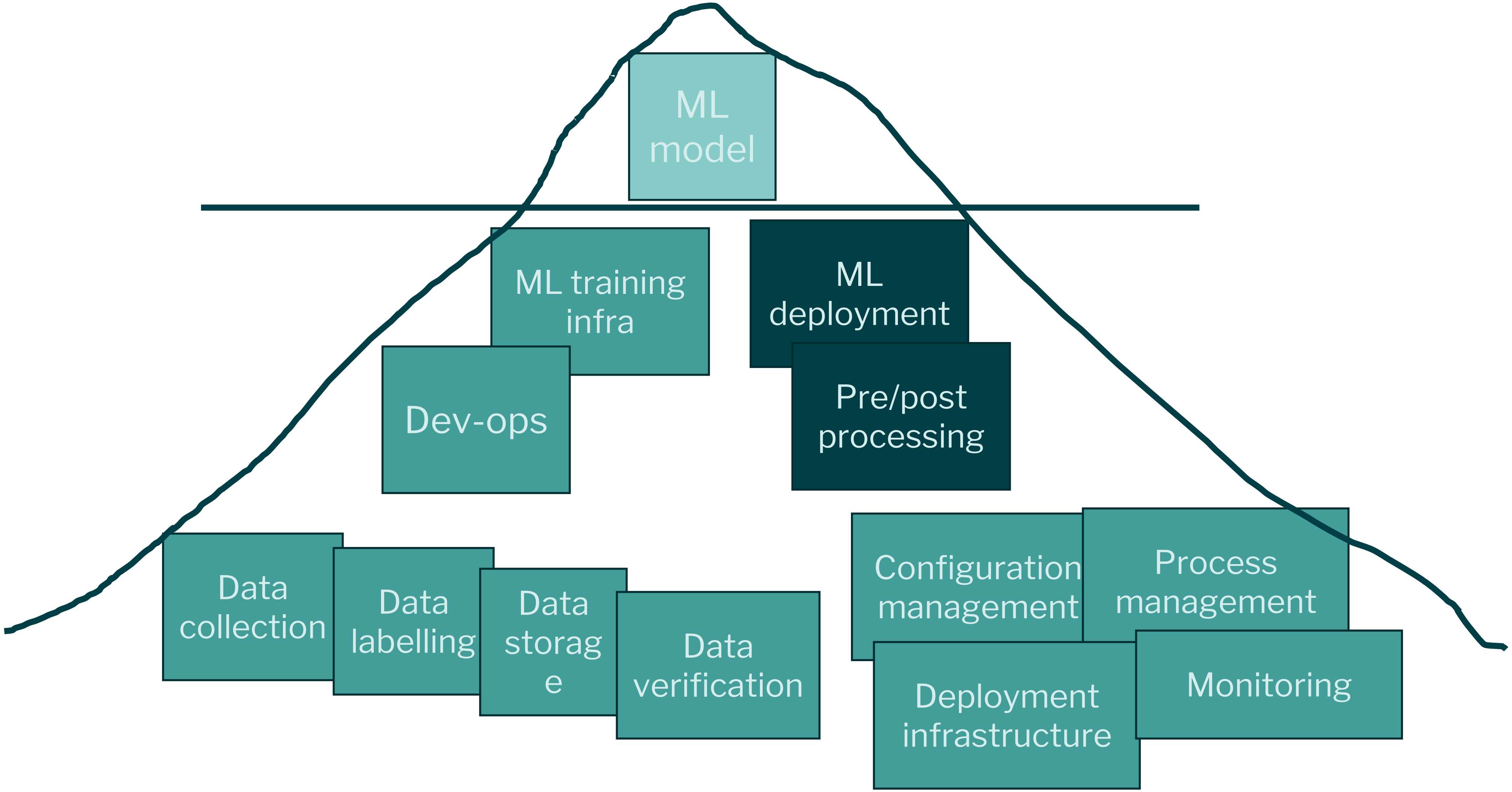
- AI model designs receive most of the attention (academic papers, media coverage)
- Network architecture has progressed in the last 8 years

Going deeper with convolutions
C Szegedy, W Liu, Y Jia, P Sermanet… - Proceedings of the …, 2015 - cv-foundation.org
We propose a deep convolutional neural network architecture codenamed Inception that achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC2014). The main hallmark of this architecture is …
Cited by 29478 Related articles All 50 versions

[PDF] **Hidden technical debt** in **machine learning** systems
D Sculley, G Holt, D Golovin, E Davydov… - Advances in neural …, 2015 - kaust.edu.sa
Abstract **Machine learning** offers a fantastically powerful toolkit for building useful complex prediction systems quickly. This paper argues it is dangerous to think of these quick wins as coming for free. Using the software engineering framework of **technical debt**, we find it is …
Cited by 483 Related articles All 8 versions

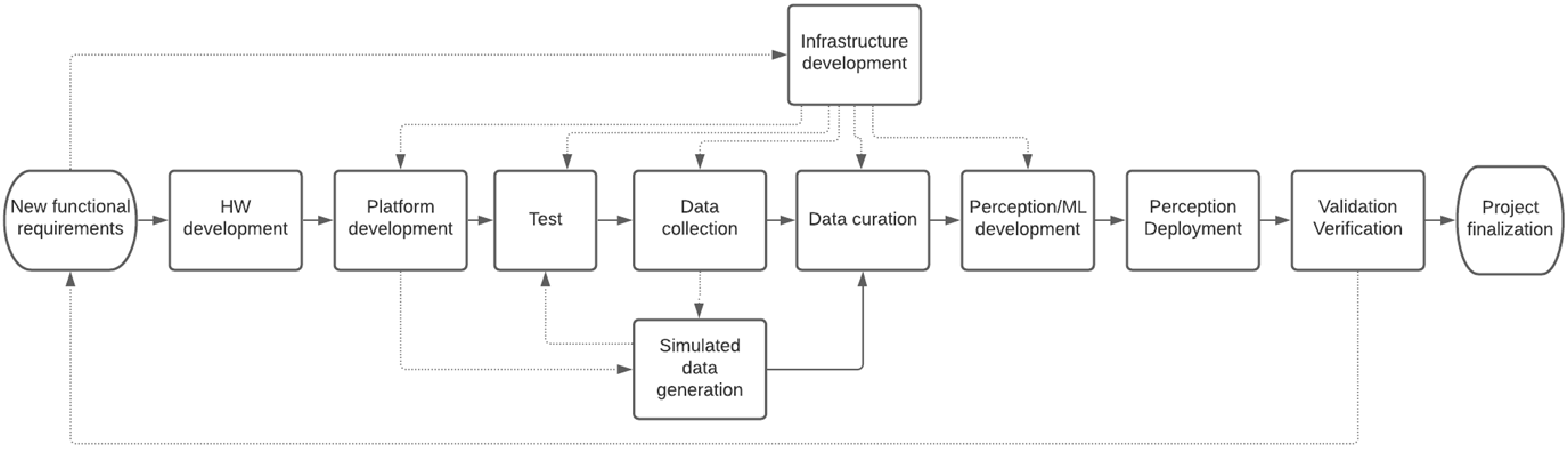
- AI model design is the tip of the iceberg
- ML-OPS trend - the rest of the iceberg - is gaining momentum

System Productization (MLOPS)

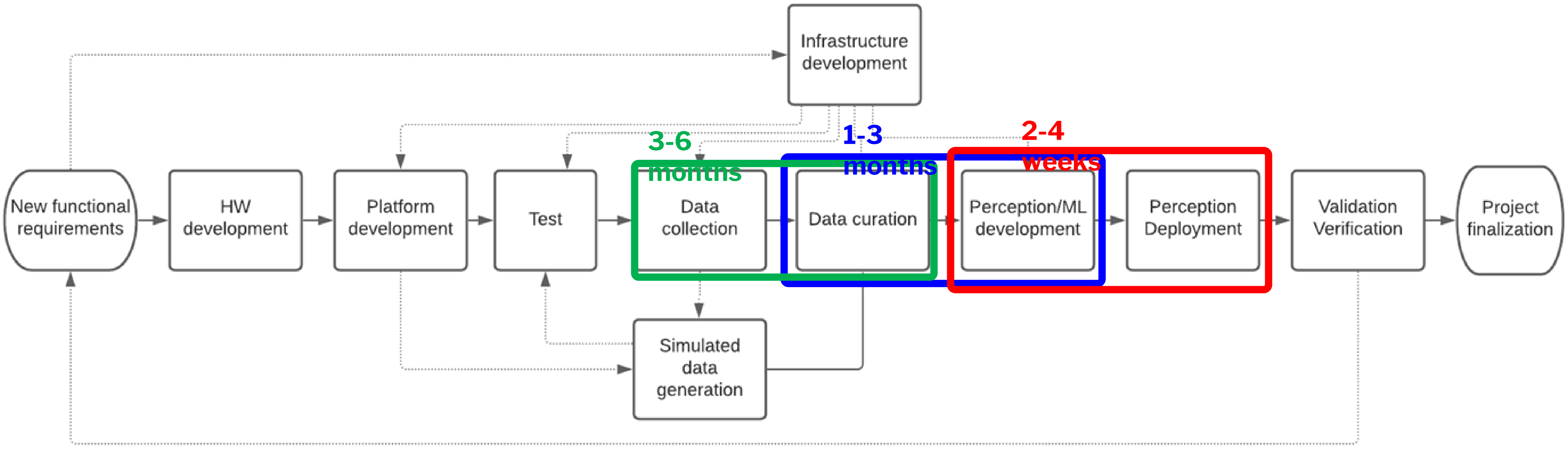


AN AIRBUS INNOVATION CENTER

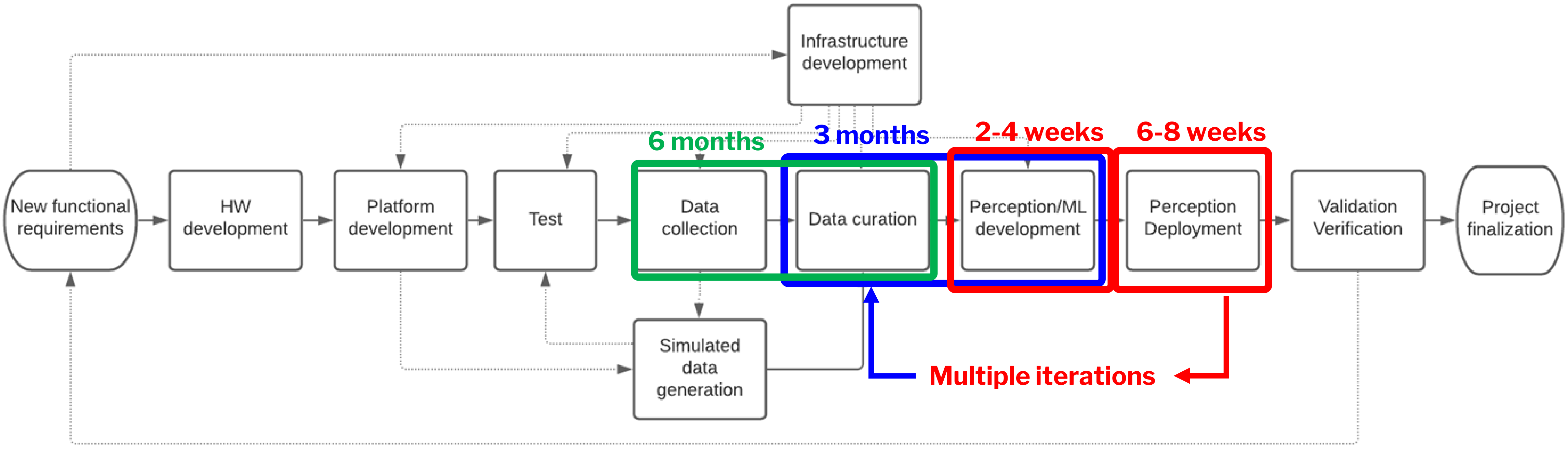
System Productization Process



Expensive Cycles



Expensive Cycles

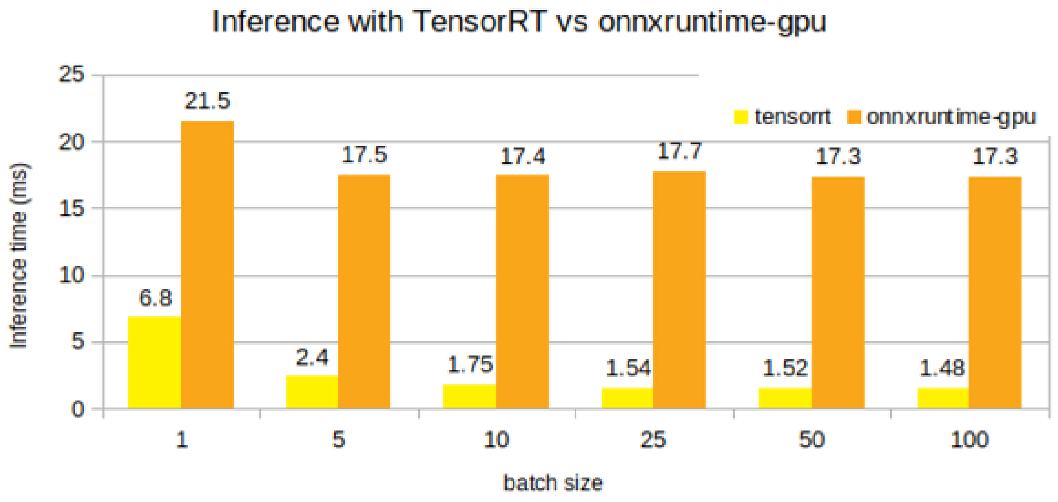


AN AIRBUS INNOVATION CENTER

Model Deployment

Model Deployment

- Model deployment (or model serving) is the process of preparing a trained AI model (architecture and weights) for deployment
  - Language conversion (Python -> C++ …)
  - Model optimization: Pytorch -> TensorRT (10x speedup on REsNet50)
- Model deployment is agile, as most operations can be automated



Plot source::
Accelerate PyTorch Model With TensorRT via ONNX

Hidden Complexity of Model Deployment

- Real world
  - 1 model → inference graph
    - Multiple ML models
    - Pre/post processing, including multiple traditional CV and ad-hoc components.
- Deployment complexity drivers
  - **Memory** footprint complexity dominated by ML models
  - Code and **software** complexity dominated by traditional CV components
  - Porting **time** severely dominated by CV/ ad-hoc components (minutes vs weeks)

Model Deployment Approaches

A) Develop on a C++ codebase [scarcity of developers]

B) Hire an army of CV/SW experts to deploy in days [cost/complexity]

C) Minimize amount of deployed solutions [2-stage dev. process]

- Continuous development of ML models, release-staggered deployment
- Partial porting
  - Test partially-ported solutions with parallel Software In the Loop environments
  - Use message passing technologies to support cross-language communications (e.g., ZeroMQ, RabbitMQ, DDS, ROS2)

Conclusions

- Development complexity
- Overlooked process bottlenecks
- Model deployment
- Partial porting
- Software-In-the-Loop testing

Thank you

References

1. Sculley, David, et al. "Hidden technical debt in machine learning systems" (2015).
2. Doshi-Velez, Finale, and Been Kim. "Towards a rigorous science of interpretable machine learning" (2017).
3. Tfx: A tensorflow-based production-scale machine learning platform.
4. TensorRT: NVIDIA SDK for high-performance deep learning inference.
5. Seldon Machine learning deployment for enterprise.
6. Kubeflow Machine learning toolkit for Kubernetes.
7. MLflow: Open source platform for the machine learning lifecycle.
8. Code to production-ready machine learning in 4 steps.
9. A Chat with Andrew on MLOps: From Model-centric to Data-centric AI