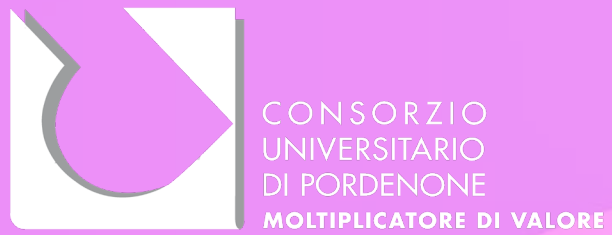
AI at the EDGE con .NET: Cosa potrebbe andare storto?

Marco Dal Pino

Sponsor

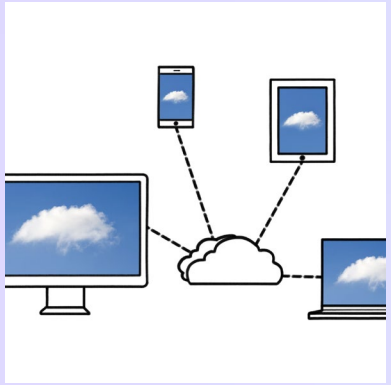

CONSORZIO
UNIVERSITARIO
DI PORDENONE
MOLTIPLICATORE DI VALORE


altitudo


beanTech
IT moves your business

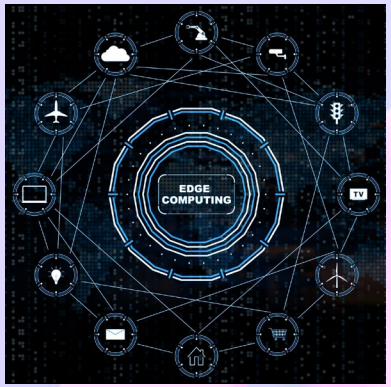

[stesi]
Powered by Innovation

Cos'è l'edge computing?



Definizione di edge computing

L'edge computing si riferisce al posizionamento dell'elaborazione dei dati vicino alla fonte per migliorare l'efficienza e ridurre la latenza.



Vantaggi dell'edge computing

L'edge computing offre diversi vantaggi, come tempi di risposta rapidi e minori requisiti di larghezza di banda, specialmente in ambienti di rete variabili.

Applicazioni dell'edge computing

Questo paradigma è utile per applicazioni come IoT, realtà aumentata e veicoli autonomi, dove le prestazioni in tempo reale sono cruciali.

Differenza tra edge computing e cloud computing

Elaborazione locale vs remota

L'edge computing si concentra sull'elaborazione dei dati localmente, mentre il cloud computing utilizza server remoti. Questa differenza influisce sulle prestazioni e sull'efficienza.

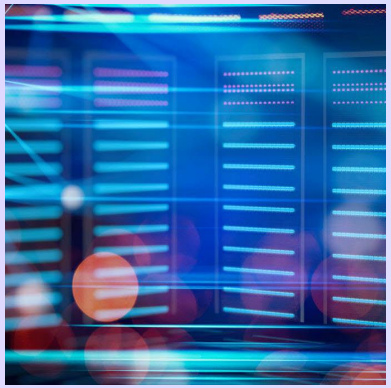
Vantaggi dell'edge computing

L'edge computing offre vantaggi come maggiore velocità, riduzione della latenza e minori costi di larghezza di banda. Questo lo rende ideale per applicazioni in tempo reale.

Applicazioni AI

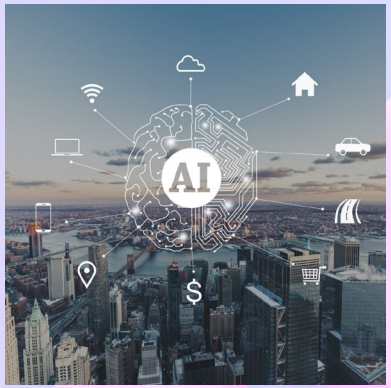
L'edge computing è particolarmente vantaggioso per le applicazioni di intelligenza artificiale, consentendo l'elaborazione dei dati in tempo reale direttamente nei dispositivi.

Vantaggi dell'utilizzo dell'AI all'edge



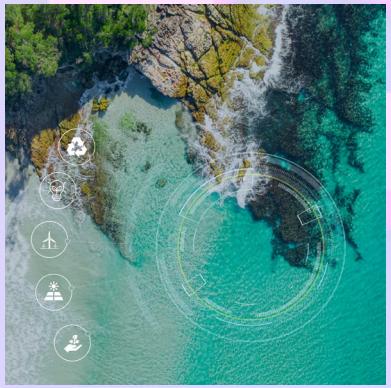
Elaborazione in tempo reale

L'AI all'edge consente l'elaborazione dei dati in tempo reale, riducendo la latenza e migliorando l'efficienza operativa.



Analisi dei dati in loco

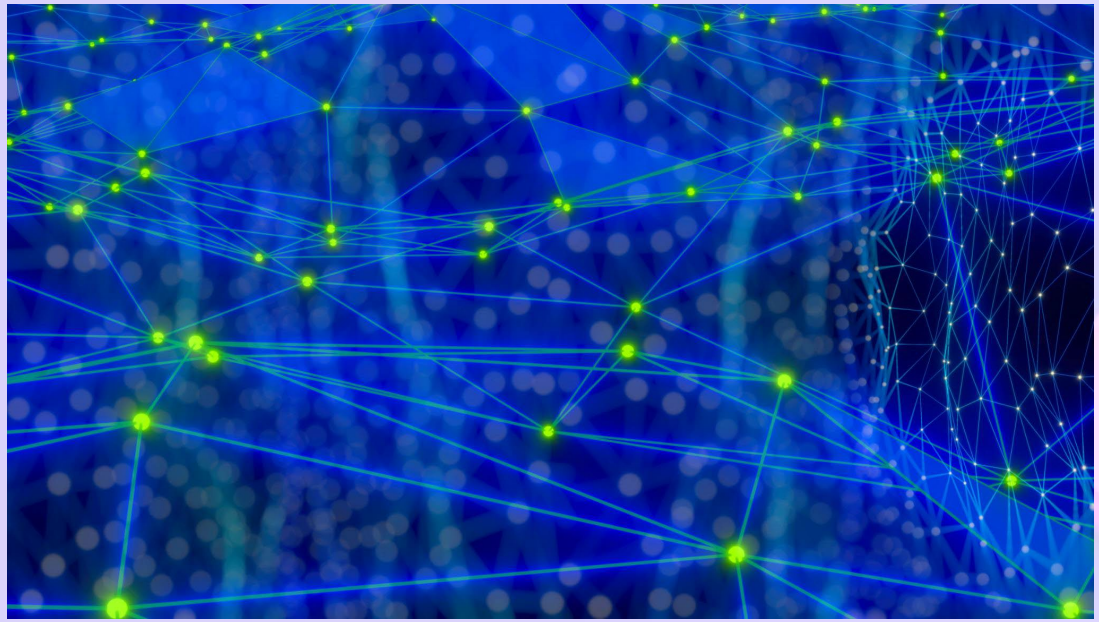
L'analisi dei dati in loco permette di prendere decisioni rapide senza la necessità di trasferire grandi volumi di dati.



Maggiore resilienza

L'AI all'edge offre una maggiore resilienza in ambienti con connettività limitata, garantendo che le operazioni possano continuare senza interruzioni.

Limitazioni di connettività e larghezza di banda



Connettività limitata

La mancanza di connettività può ostacolare le operazioni dell'AI, rendendo difficile l'accesso ai dati necessari.

Sfide per l'AI

L'AI richiede risorse e dati per l'addestramento, e la disconnessione presenta significative sfide operative.

Importanza dei dati locali

Sviluppare strategie per lavorare con dati locali è fondamentale per superare le limitazioni di connettività e garantire l'efficacia dell'AI.

Gestione dei dati e delle risorse locali

Importanza della gestione dei dati

Una gestione efficace dei dati è cruciale per il funzionamento delle applicazioni in ambienti disconnessi e per garantire l'affidabilità dell'AI.

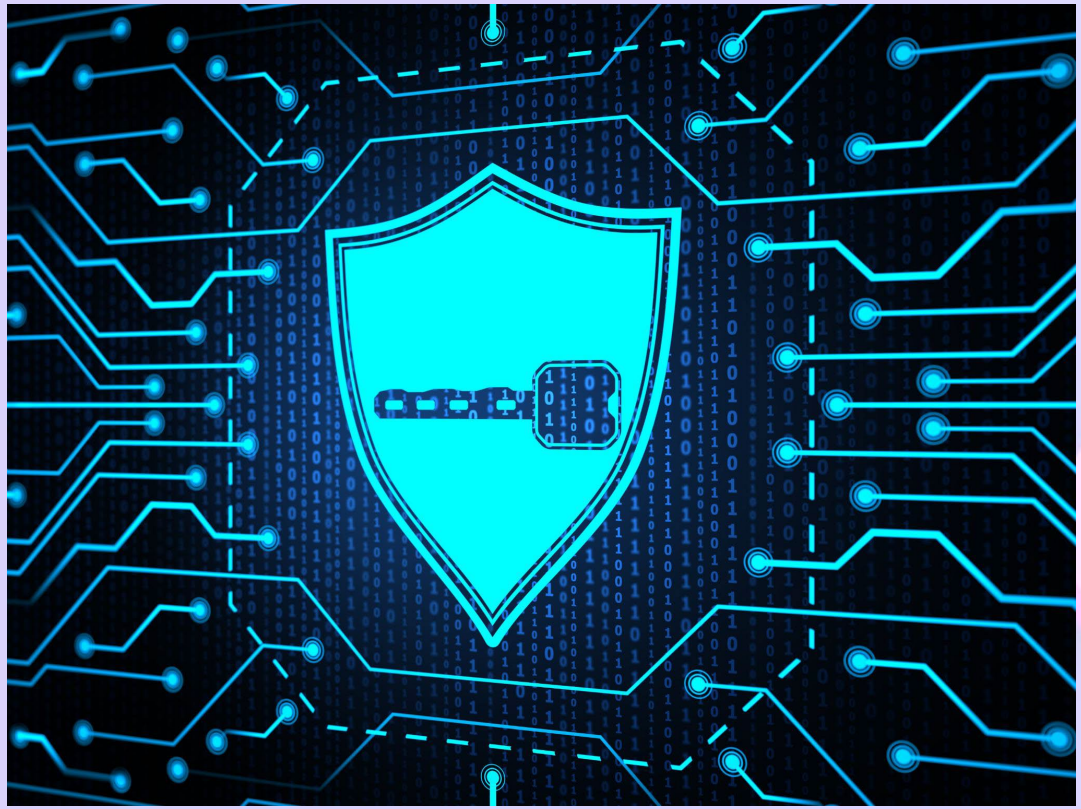
Raccolta e archiviazione locale

Gli sviluppatori devono implementare strategie per raccogliere e archiviare i dati localmente, permettendo un accesso rapido e una gestione efficiente.

Elaborazione senza server centrale

Le applicazioni devono essere progettate per operare senza un server centrale, utilizzando risorse locali per l'elaborazione e l'analisi dei dati.

Sicurezza e privacy dei dati



Importanza della sicurezza dei dati

La sicurezza dei dati è fondamentale per proteggere le informazioni sensibili quando si utilizza l'intelligenza artificiale in ambienti disconnessi.

Protezione dei dati sensibili

È essenziale garantire la protezione dei dati sensibili in scenari di elaborazione e archiviazione locali, per prevenire accessi non autorizzati.

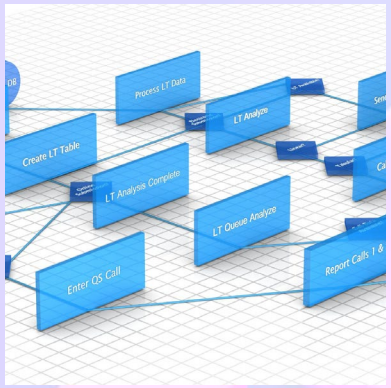
Privacy delle informazioni

La privacy deve essere garantita in ogni fase dell'elaborazione dei dati, per mantenere la fiducia degli utenti e conformarsi alle normative.

Utilizzo di ML.NET per modelli di machine learning

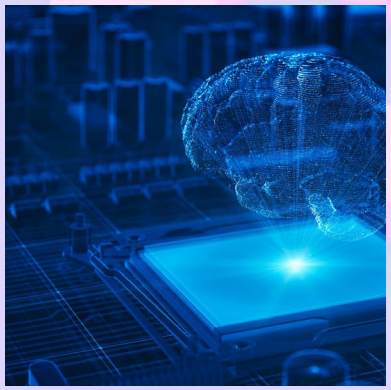
Cos'è ML.NET?

ML.NET è una libreria open-source progettata per facilitare il machine learning con il framework .NET, contribuendo a sviluppare soluzioni intelligenti.



Addestramento di modelli

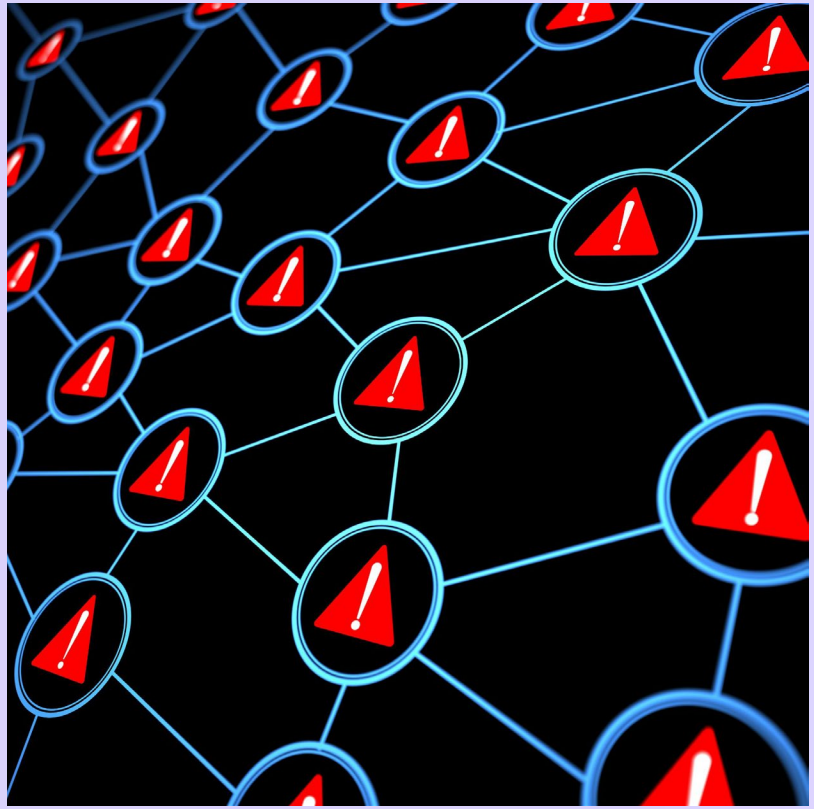
Una delle feature di ML.NET e' quella di permettere di addestrare modelli di machine learning per risolvere problemi specifici in modo efficiente.



Implementazione in ambienti edge

ML.NET consente l'implementazione di modelli di machine learning in ambienti edge, ottimizzando le prestazioni e la scalabilità.

Implementazione di ONNX Runtime per inferenze AI



Cos'è ONNX Runtime?

ONNX Runtime è un motore di inferenza progettato per eseguire modelli AI in modo altamente efficiente e veloce.

Integrazione in .NET

La corretta integrazione di ONNX Runtime nelle applicazioni .NET può migliorare significativamente le prestazioni delle inferenze AI.

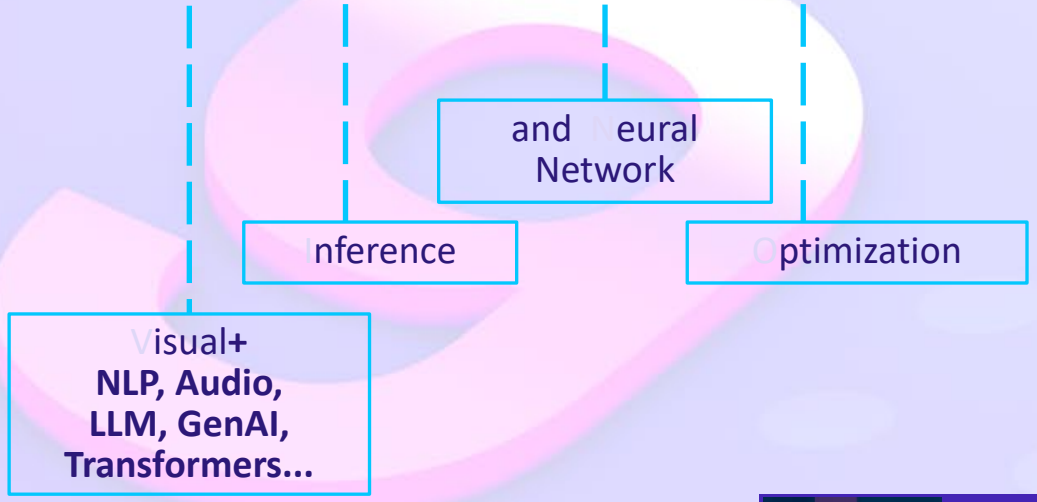
Prestazioni in condizioni limitate

ONNX Runtime è progettato per mantenere alte prestazioni anche in ambienti con risorse limitate, garantendo una veloce inferenza.

Google Summer of Code

open source initiative® Apache 2.0

OpenVINO™



Powered by oneAPI

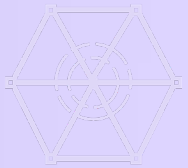
altitudo beanTech IT moves your business [stesi] Powered by Innovation CONSORZIO UNIVERSITARIO DI PORDENONE MOLTIPLICATORE DI VALORE

OpenVINO™

Developer Journey

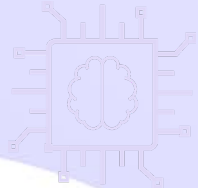
1 Model → 2 Optimize → 3 Deploy

Benefits of Building Applications with OpenVINO™



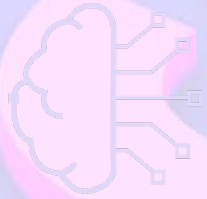
Build and deploy AI applications in simple steps

Faster inference speed

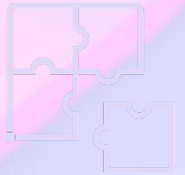


Maximize AI performance across CPU, GPU, NPU

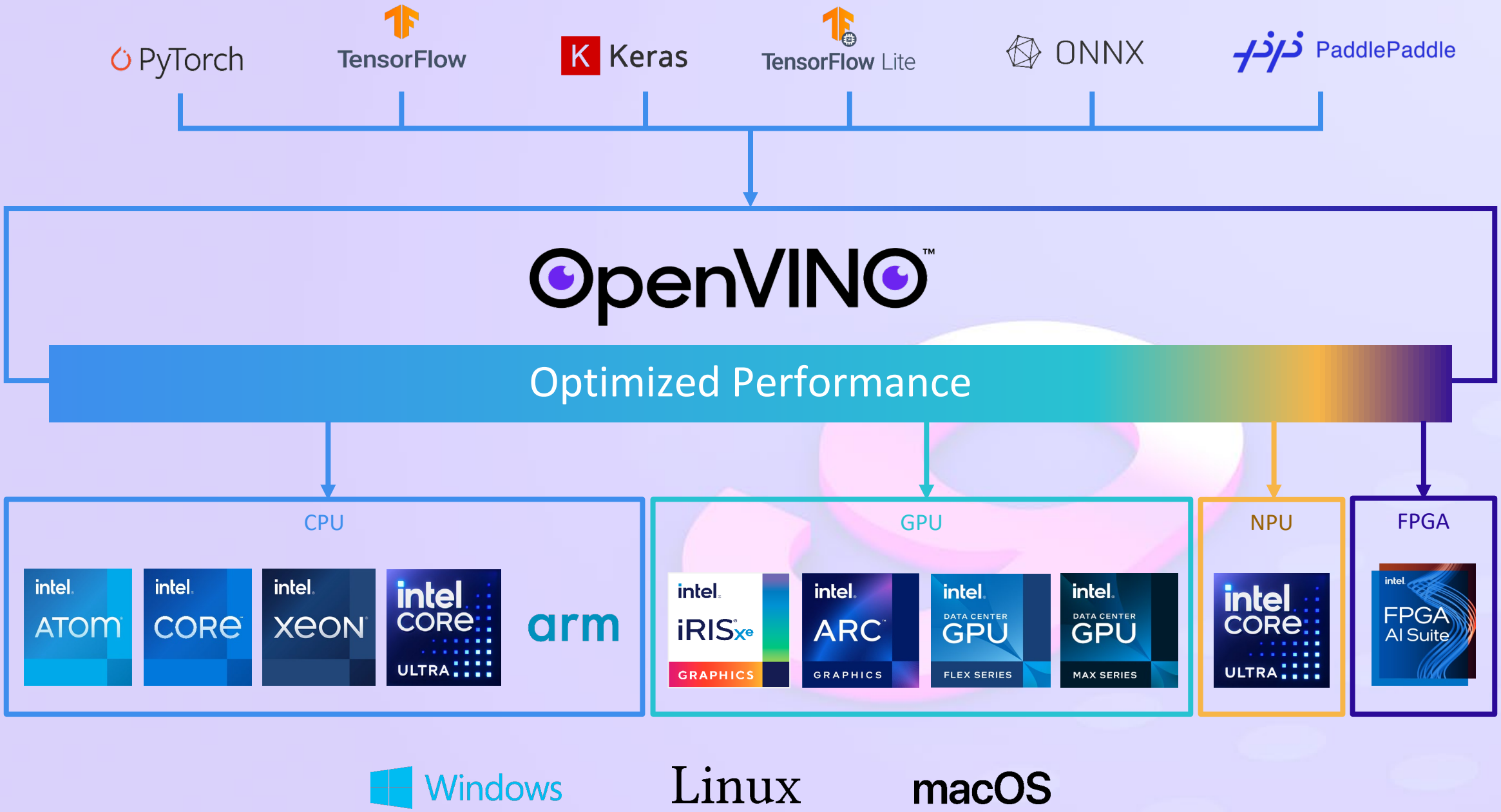
Smaller model and binary size



Reduce memory footprint



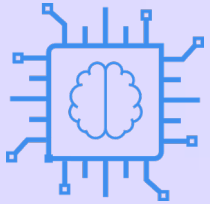
Ability to scale to many nodes with serving



What's New in the 2024.3 Release?

1 Model

- OpenVINO Model Hub on Hugging Face with OpenVINO optimized Gen AI models
- Integration with LlamaIndex, Langchain
- Noteworthy notebooks added: LLama3.1, GLM-4, Phi-3-Vision, PixArt, Parler-TTS, etc.



2 Optimize

- MXFP4 data format supported in NNCF
- BF16 supported in PTQ
- Significant LLM performance improvements for built-in GPUs and dGPUs

3 Deploy

- Tokenizers and StringTensor supported for LLM inference for Node.js API
- Support running LLM on multiple devices with PIPELINE_PARALLEL
- LLM performance improvements on OVMS, with vLLM integration

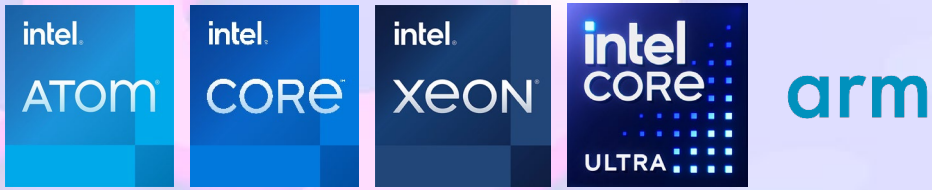
Supported Devices

```
compiled_model = core.compile_model(model=model, device_name="CPU")
```

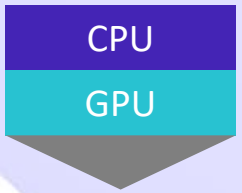
Supported Devices

CPU

```
compiled_model = core.compile_model(model=model, device_name="CPU")
```

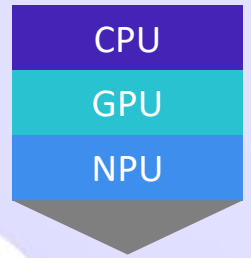


Supported Devices



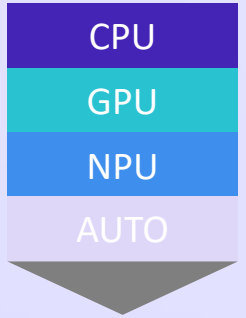

```
compiled_model = core.compile_model(model=model, device_name="GPU")
```

Supported Devices

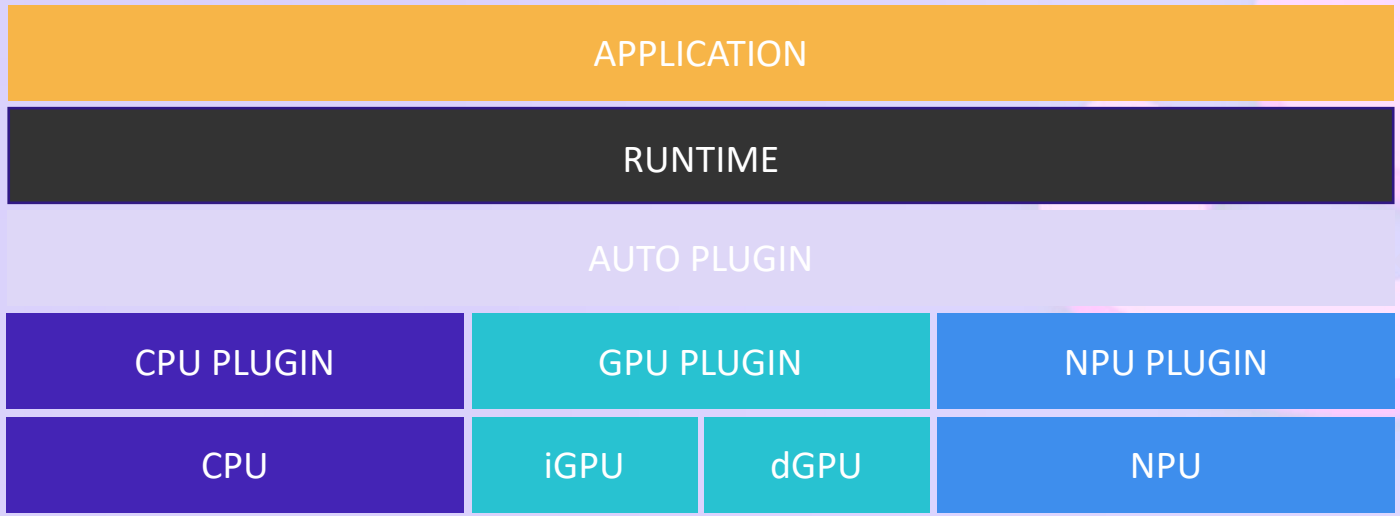



```
compiled_model = core.compile_model(model=model, device_name="NPU")
```

Auto Device




```
compiled_model = core.compile_model(model=model, device_name="AUTO")
```



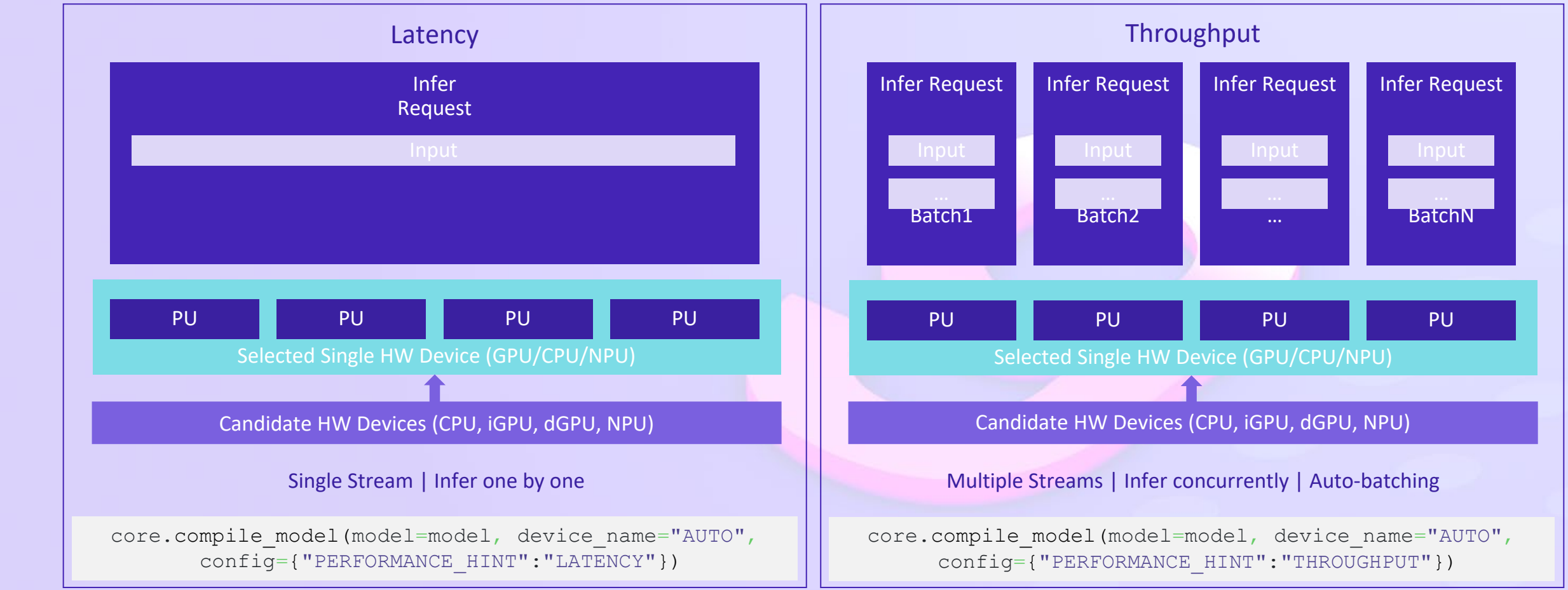
AUTO chooses a device best-suited for the task

AUTO configures the device based on hints

AUTO handles the exec logic on multiple devices

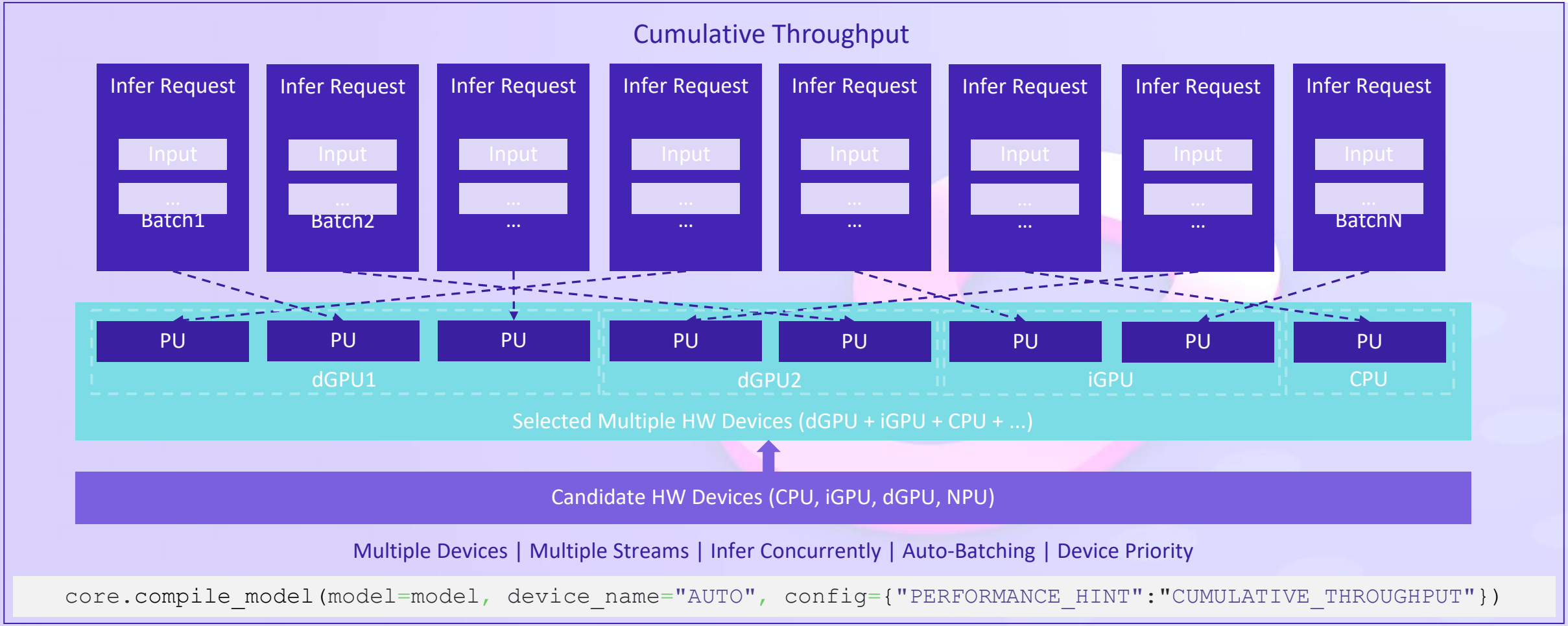
Excluded for now

Performance Hints



PU = Processing Unit

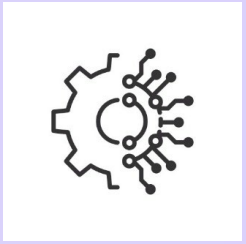
Performance Hints



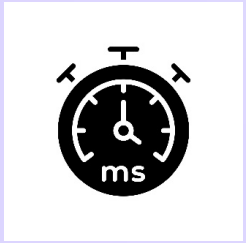

```
core.compile_model(model=model, device_name="AUTO", config={"PERFORMANCE_HINT":"CUMULATIVE_THROUGHPUT"})
```

PU = Processing Unit

Benefits of Unlocking Full Hardware Potential



Intelligent
Automation

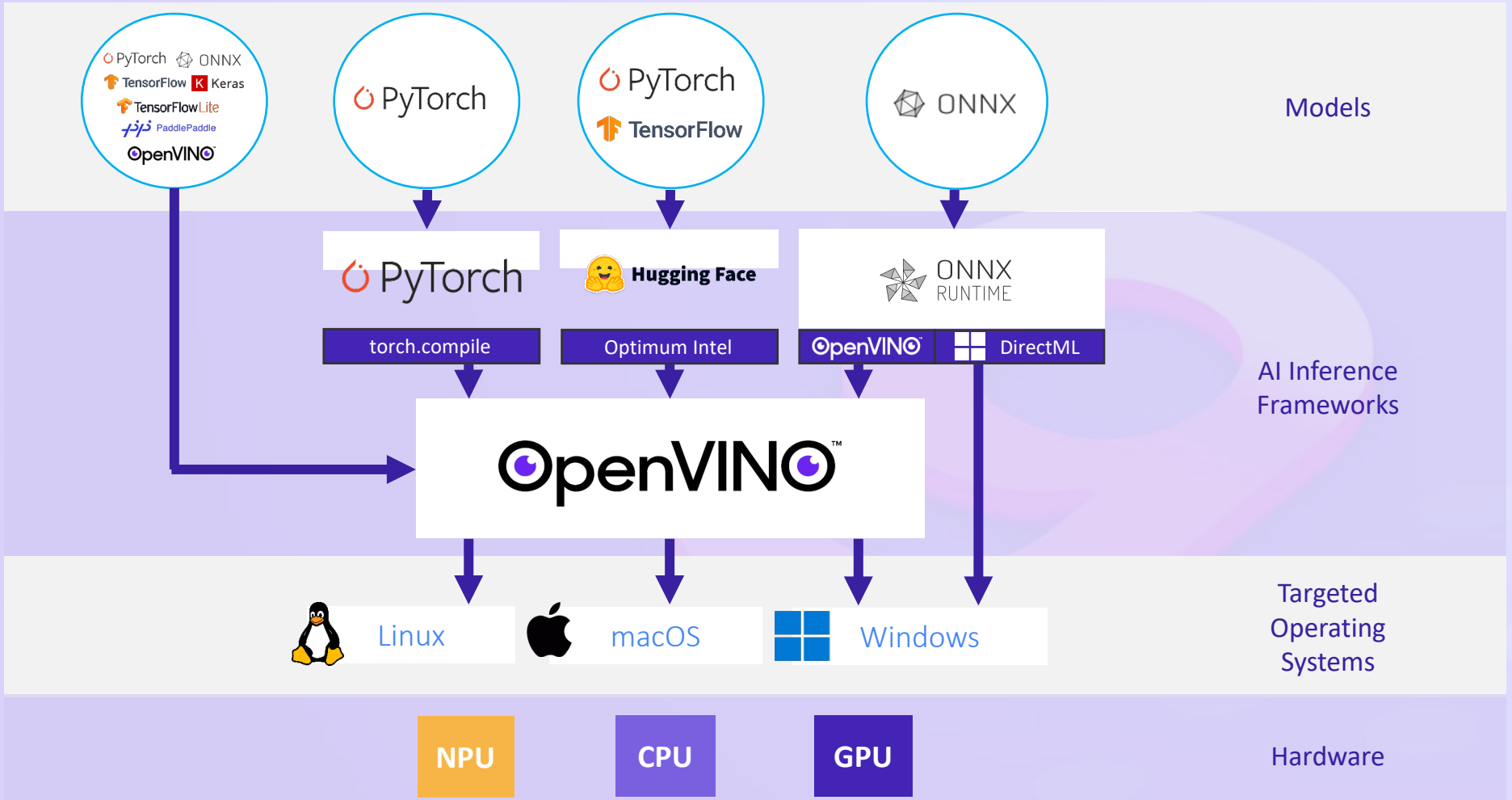


Low Latency
Inference

Maximized
Data Throughput

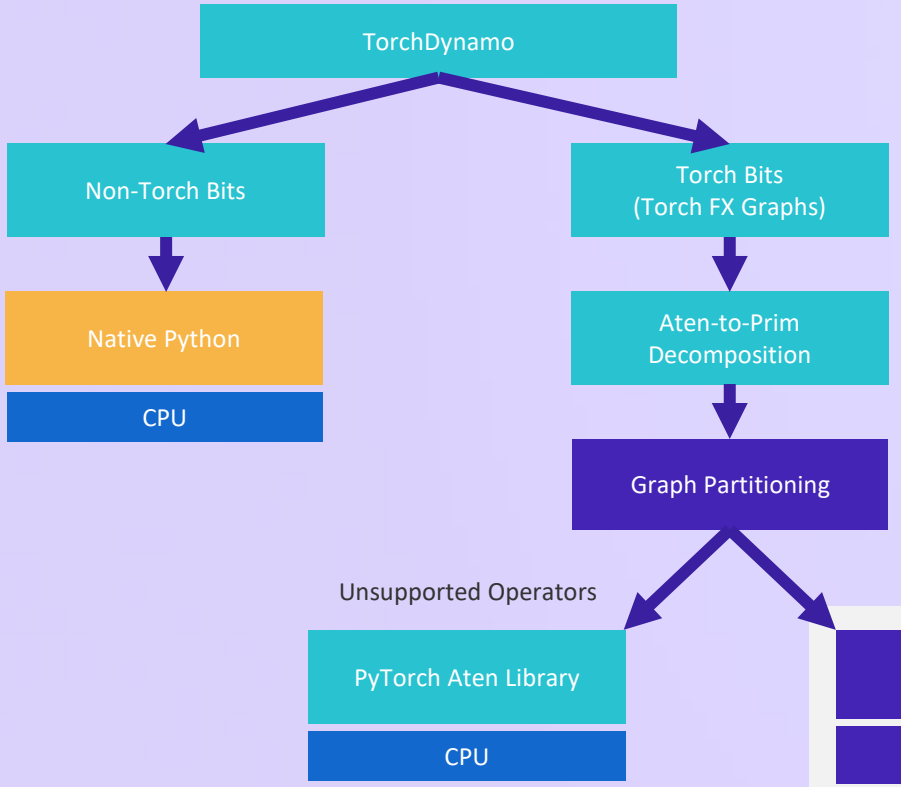
Aggregated Device
Utilization

OpenVINO™ as Backend



OpenVINO™ as Backend

PyTorch 2.0

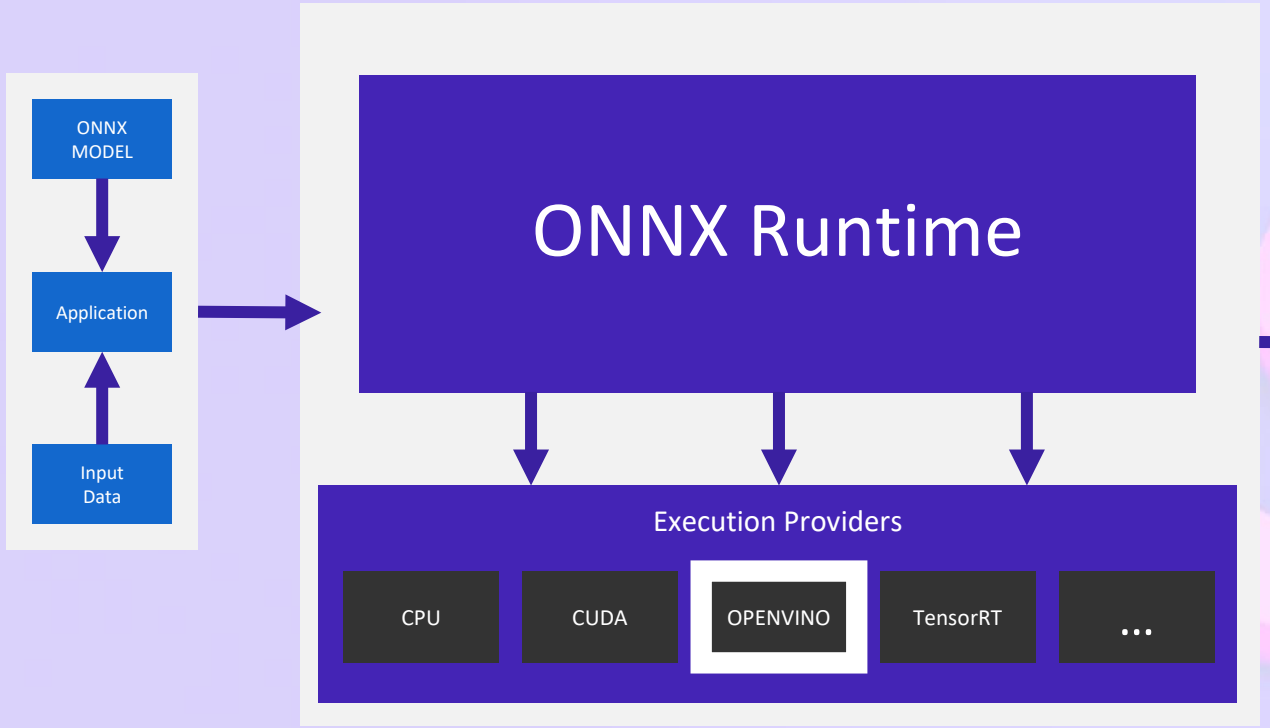



```python
import torch
import torchvision.models as models
import openvino.torch

model = models.resnet50(pretrained=True)
input = torch.rand((1,3,224,224))
model = torch.compile(model, backend="openvino")
pred = model(input)
```

Only 2 lines of codes

OpenVINO™ Execution Provider for ONNX



Installation

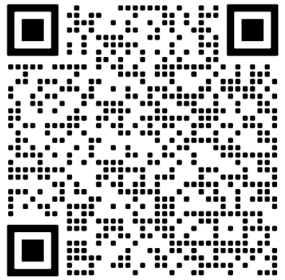
Linux

```
pip install onnxruntime-openvino
```

Windows

```
pip install onnxruntime-openvino openvino
```

OpenVINO™ Execution Provider for ONNX



```python
import onnxruntime as rt

# Any hardware target of 'CPU_FP32', 'GPU_FP32', 'GPU_FP16', 'MYRIAD_FP16', 'VAD-M_FP16'
device = "CPU_FP32"

# Specify the path to the ONNX model, register the OpenVINO EP and device for inference
sess = rt.InferenceSession("model.onnx", providers=["OpenVINOExecutionProvider"],
                           provider_options=[{"device_type" : device}])

# Preprocessing the input frame and reshaping it
preprocessed_image = image_preprocess(frame)

# Running the session by passing in the input data of the model
out = sess.run(None, {input_name: preprocessed_image})
```

OpenVINO™ Integration with Optimum

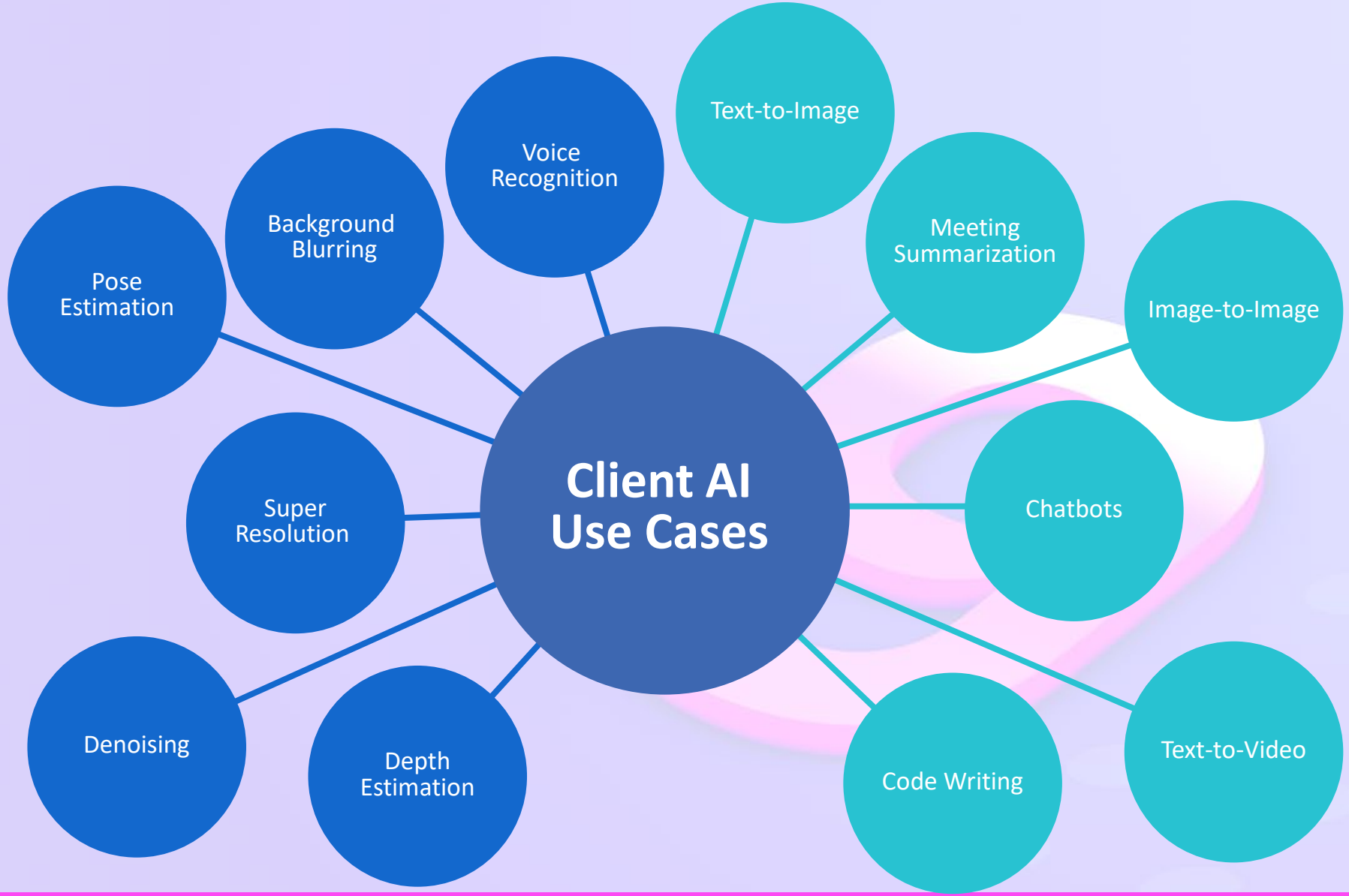
+ OpenVINO™

Combine the convenience of Hugging Face API with the efficiency of OpenVINO™!

Client AI Use Cases

Conventional AI

Generative AI



Conventional and Generative AI Comparison

Model Type	Goal	Model Size	User Interactions	Latency
Conventional AI	Output predication based on input and the model	From thousands to hundreds of millions of parameters	Inference runs continuously in the background in many cases	Low latency
Generative AI	Generates unique content: text, code, music	From single digit billions to trillions of parameters	Users interact with the model	Higher latency is accepted

Generative AI Introduces New Capabilities and Use Cases

	Conventional AI	Generative AI
Goal	Output predication based on input	Create original content
Capabilities	• Object Detection • Speech Recognition • Text Classification	• Text Generation • Image Generation • Audio Generation
Use Cases	• Background Blurring • Pose Estimation • Denoising	• Chatbots • Code Writing • Video Creation

Conventional AI and Generative AI Have Different Requirements

	Conventional AI	Generative AI
Model Size	From thousands to hundreds of millions of parameters	From billions to trillions of parameters
User Interaction	Ongoing background inference	Inference prompted by users
Latency	Low latency	Higher latency is accepted

AI in the Cloud

AI Service Client Module

AI Service Server Module

AI Trained Model

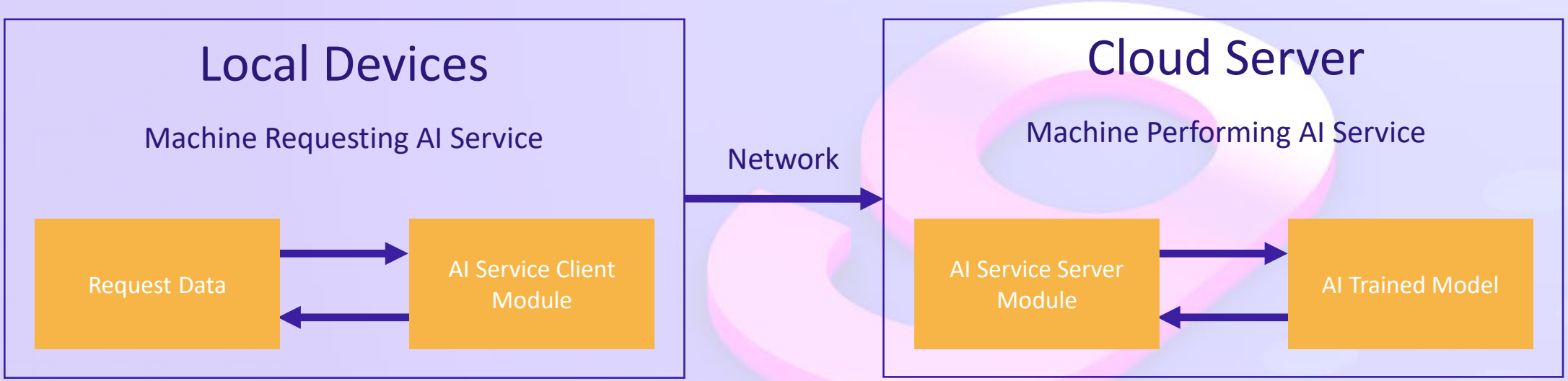
Request Data

Network

Machine Requesting AI Service (Client Device)

Machine Performing AI Service (Cloud Server)

AI Workloads Have Typically Relied on Cloud Computing

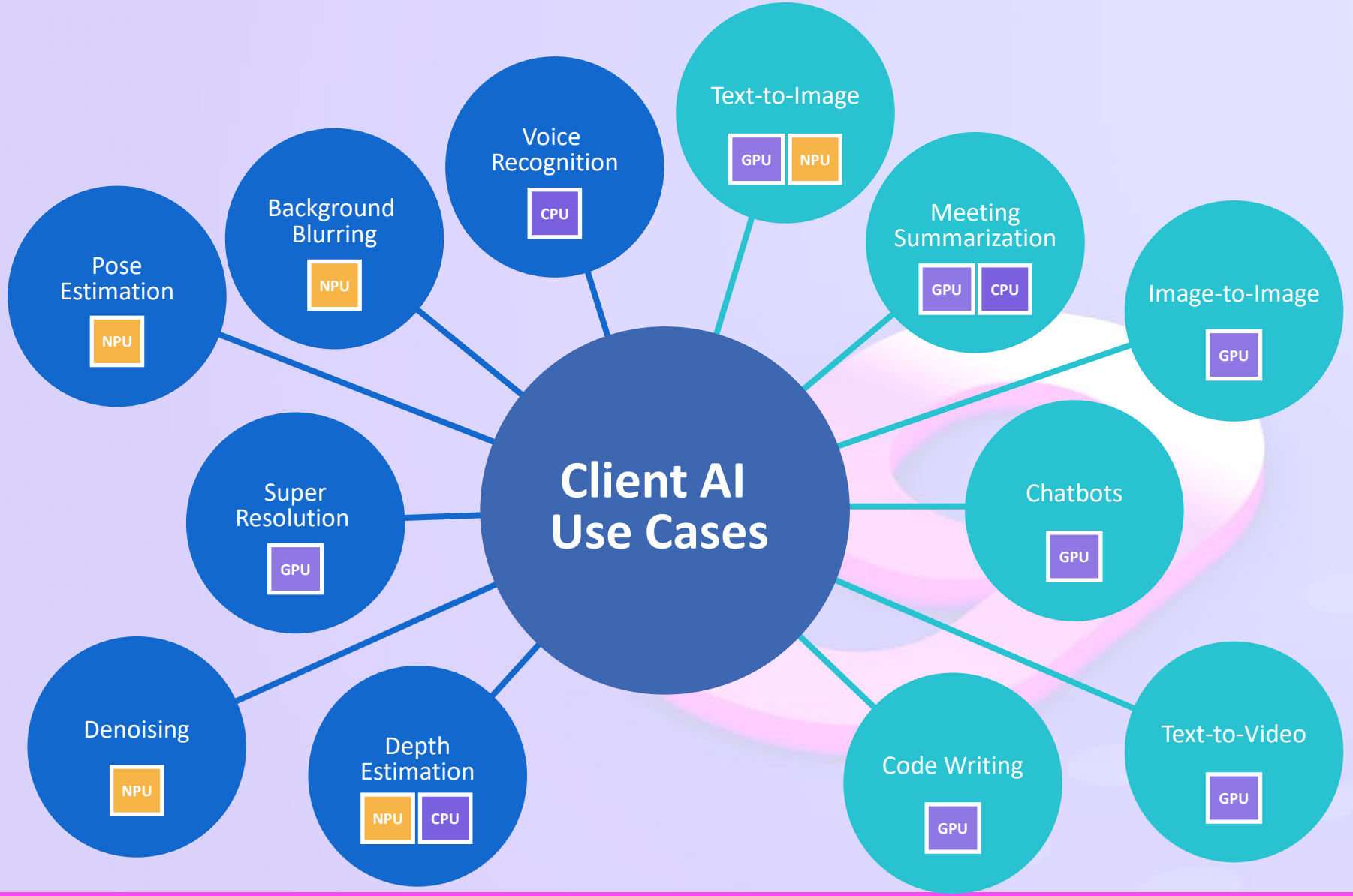


Edge Computing Bridges the Gap

	Edge	AI PC	Cloud
Pros	• Data sovereignty • Cost efficiency • Increased control • Real-time data processing • Wider reach • Autonomous execution	• Data sovereignty • Cost efficiency • Increased control • Optimal energy consumption • Special computing capabilities	• Limitless computing on demand
Cons	• Limited local computing	• Limited computing resources	• Risk of data privacy • High latency • Internet dependency

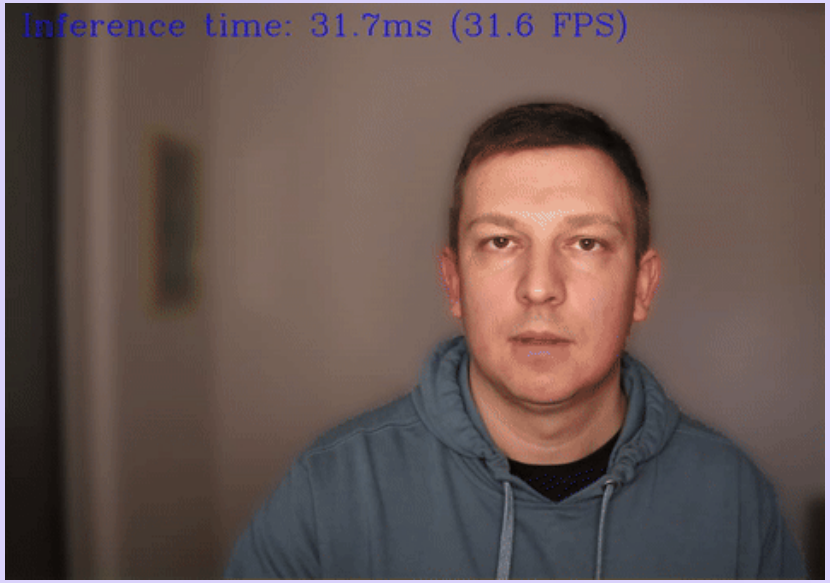
Conventional AI

Generative AI

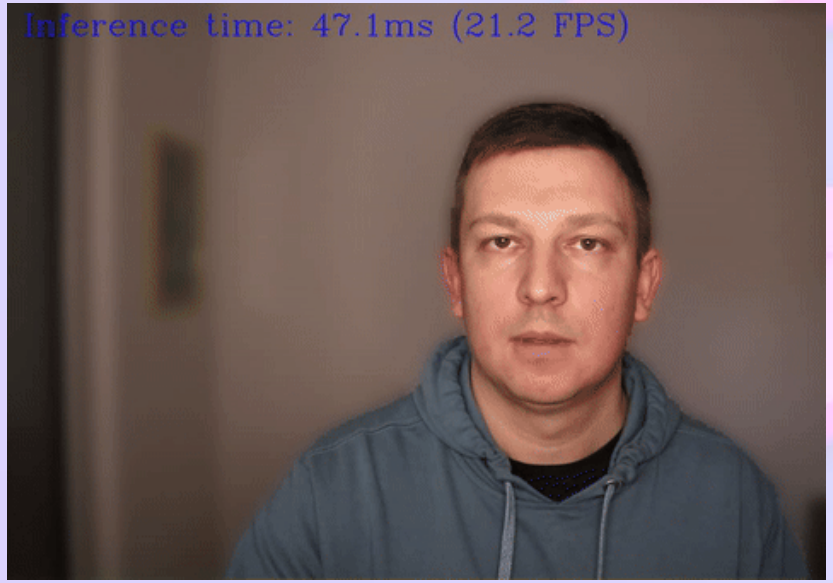


CPU/NPU/GPU Comparison

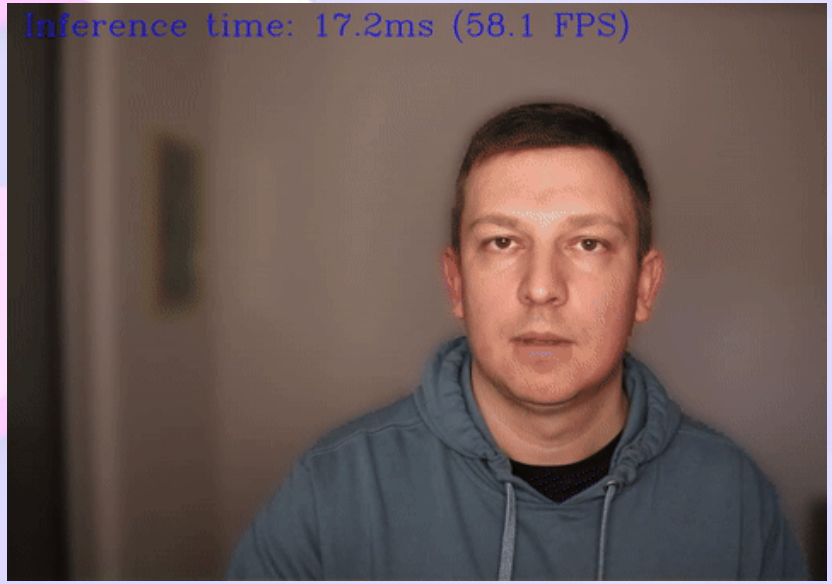
NPU



CPU

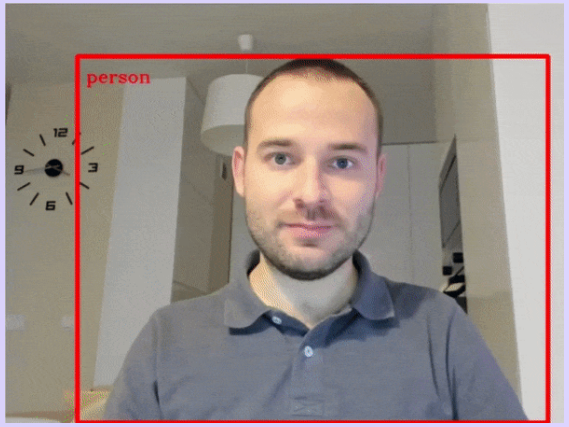


GPU

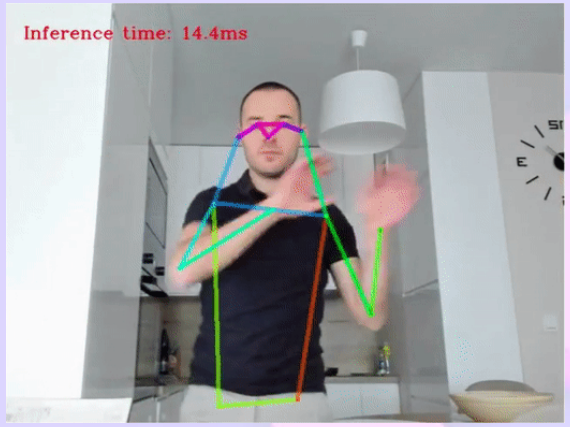


OpenVINO™ Notebooks

Object Detection



Pose Estimation



Action Classification

altitudo

beanTech
IT moves your business

[stesi]
Powered by Innovation

CONSORZIO UNIVERSITARIO DI PORDENONE
MOLTIPLICATORE DI VALORE

Questions?

- 30+ years in IT (Developer, Architect, Consultant, PM, Trainer)
- Speaker, Community addicted
- IoT Influencer
- Microsoft Certified Trainer

Marco Dal Pino
Technical Consultant
Microsoft

https://www.linkedin.com/in/marcodalpino
https://about.me/marcodalpino
https://twitter.com/marcodalpino
info@contoso.blog
https://www.twitch.tv/dpcons
https://www.twitch.tv/techchat

Microsoft MVP Most Valuable Professional Reconnect

DotNet Toscana

altitudo beanTech IT moves your business [stesi] Powered by Innovation

CONSORZIO UNIVERSITARIO DI PORDENONE MOLTIPLICATORE DI VALORE