VINROBOTICS

Edge AI for Robotics

Khanh Nguyen

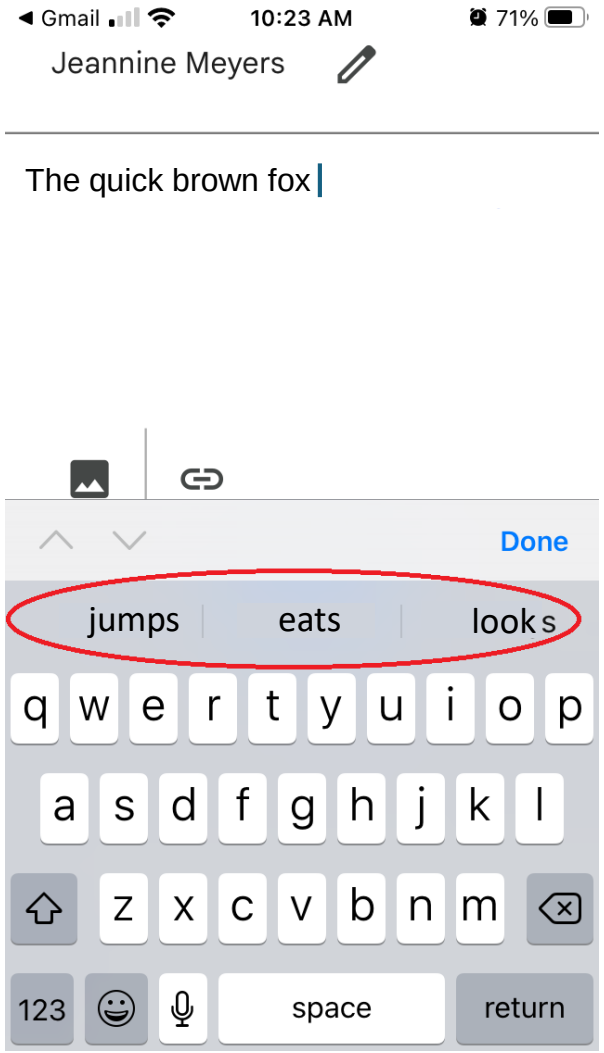
INTRODUCTION

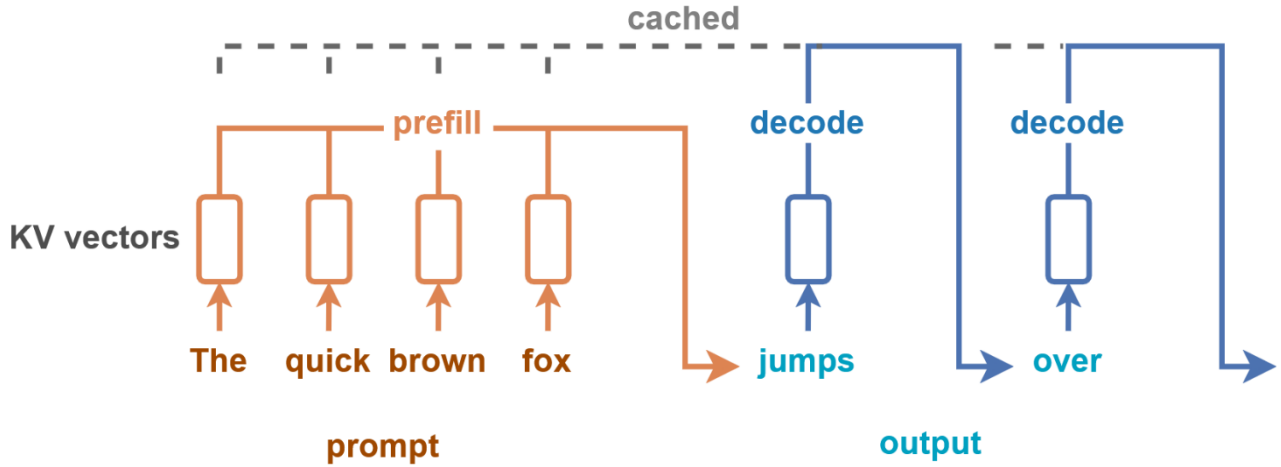
From LLM to VLA

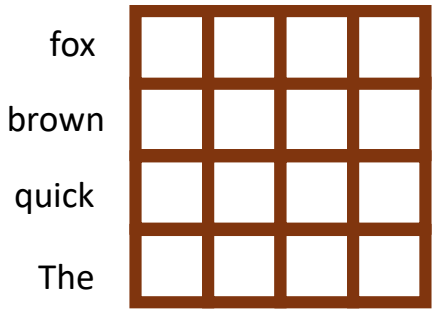
01

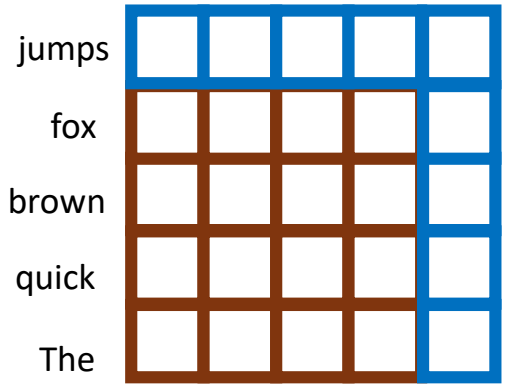
LLMs

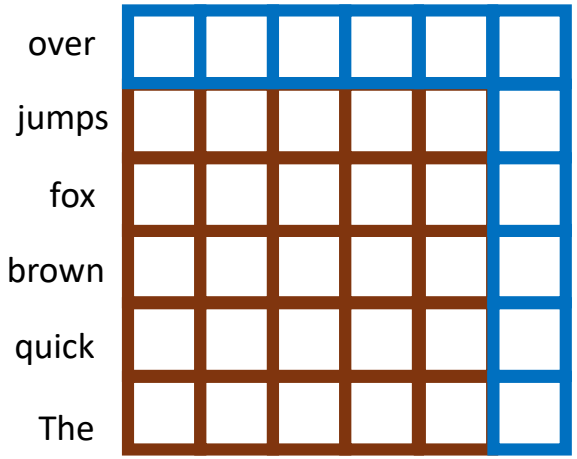
VINROBOTICS





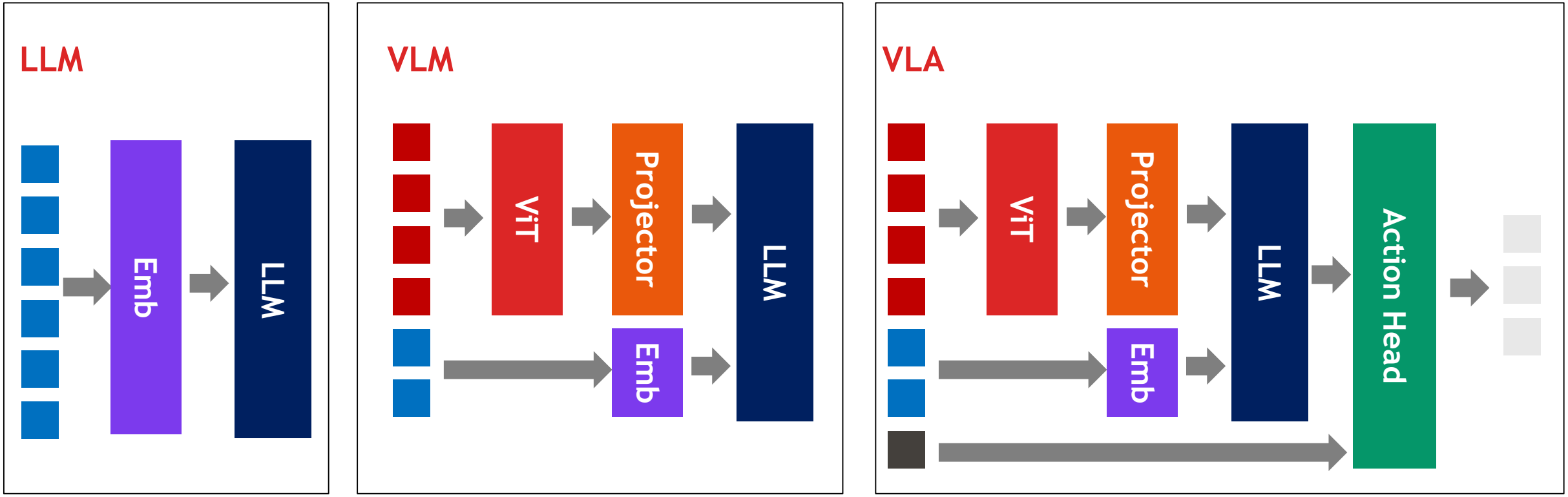






From LLM to VLM, then VLA

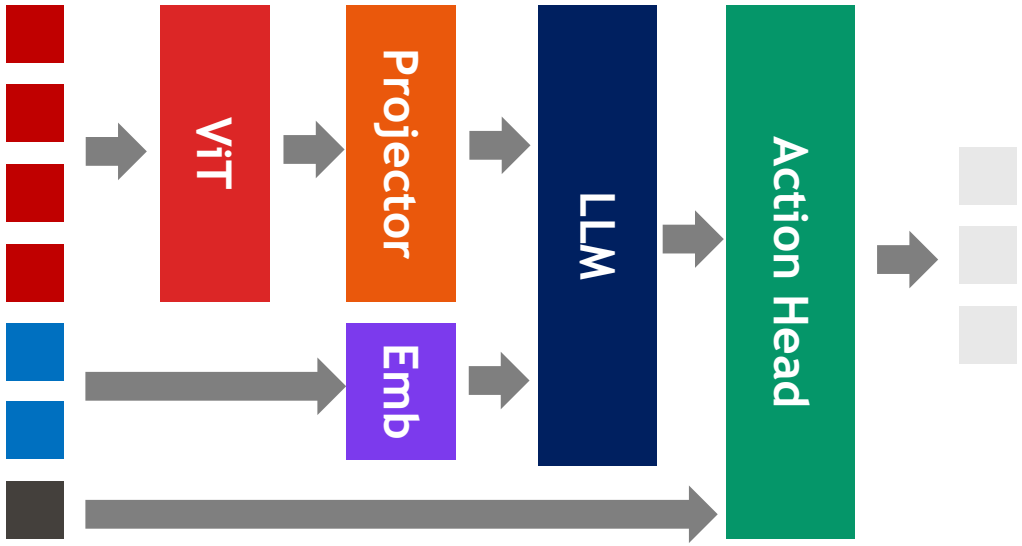
VINROBOTICS



VLA Architecture & Landscape

VINROBOTICS

Typical VLA Architecture



Input state

Output actions

Image patches

Language input

Notable VLA Models

- **RT-2 (Google)**
  PaLM-E backbone, web-scale pre-training, 55B parameters
- **OpenVLA**
  Open-source 7B VLA, Prismatic ViT + Llama 2 backbone
- **π0 (Physical Intelligence)**
  Flow matching action head, pre-trained across 7 robot types
- **GR00T (NVIDIA)**
  Foundation model for humanoid robots, multimodal I/O
- **Octo (Berkeley)**
  Transformer-based, generalizes across embodiments
- **SmolVLA (LeRobot)**
  Sub-billion foundation VLA model

VR VLA models with NVIDIA GR00T N1.5

VINROBOTICS

Fine-tuning Groot-N1.5 for humanoid service product.

VR's System 0+1+2 Architecture:

- Full speech to low-level actions
- System 2: Task reasoning at 1Hz
- System 1: The fine-tuned GrootN1.5 at 30Hz
- System 0: Very fast and smooth reactive controller at 200Hz

EDGE AI

Running AI models at the edge

AI Deployment Tiers

VINROBOTICS

Edge AI runs machine learning models directly on local devices - smartphones, robots, cameras - instead of relying on remote cloud servers.

Cloud AI

- kW to MW per rack
- 100s of GB HBM
- 10B to 1T params
- BF16, FP8
- vLLM, CUDA, XLA
- Training, LLM serving

Edge AI

- 5 to 300 W
- 4 to 64 GB LPDDR
- 0.1B to 10B params
- INT8, INT4, ternary
- TensorRT, ggml, NPU
- Robotics, ADAS, AR/VR

TinyML

- 0.1 to 100 mW
- 32 KB to 2 MB SRAM
- under 1M params
- INT8, binary, no FPU
- TFLM, CMSIS-NN
- Wake word, anomaly

LLM deployment

VINROBOTICS

SGLang

REMEMBER THE PREFIX

A serving engine built around **RadixAttention**: an LRU radix tree over KV blocks that shares cached prefixes across requests, not just within one.

vLLM

THE THROUGHPUT DEFAULT

A GPU serving engine whose core idea, **PagedAttention**, treats KV cache like virtual memory so continuous batching can pack hundreds of concurrent requests without fragmentation waste.

llama.cpp

INFERENCE ANY WHERE

Took large-model **inference** off the GPU and **onto commodity CPUs** through quantized kernels, putting local LLMs within reach of anyone with a laptop.

Unsloth

LOW-COST TRAINING

Cut **fine-tuning** memory and time by a large factor with custom kernels and PEFT, so **a single consumer card** can train models that once needed a cluster.

No dedicated engine for VLA deployment (yet)

Deployment gap of LLM and VLA

VINROBOTICS

Optimizing LLM and optimizing VLA share common techniques

LLM

- KV cache grows.
- Weights don't fit.
- Decode is serial.

VLA

- Hard real time.
- Errors compound
- Batch size is 1.

Not all LLM techniques work with VLA

Model Optimization

Making large models fit small devices

Edge AI Optimization Techniques

VINROBOTICS

Quantization

4x

Compression

Reduce precision from FP32 to INT8/INT4. Up to 4x smaller models with minimal accuracy loss. Post-training & quantization-aware training.

Pruning

50-90%

Sparsity

Remove redundant weights and neurons. Structured pruning removes entire channels for hardware-friendly speedup.

Knowledge Distillation

10x

Smaller Model

Train a small 'student' model to mimic a large 'teacher'. Captures complex behaviors in a lightweight architecture.

Kernel Optimization

Faster

Implementation

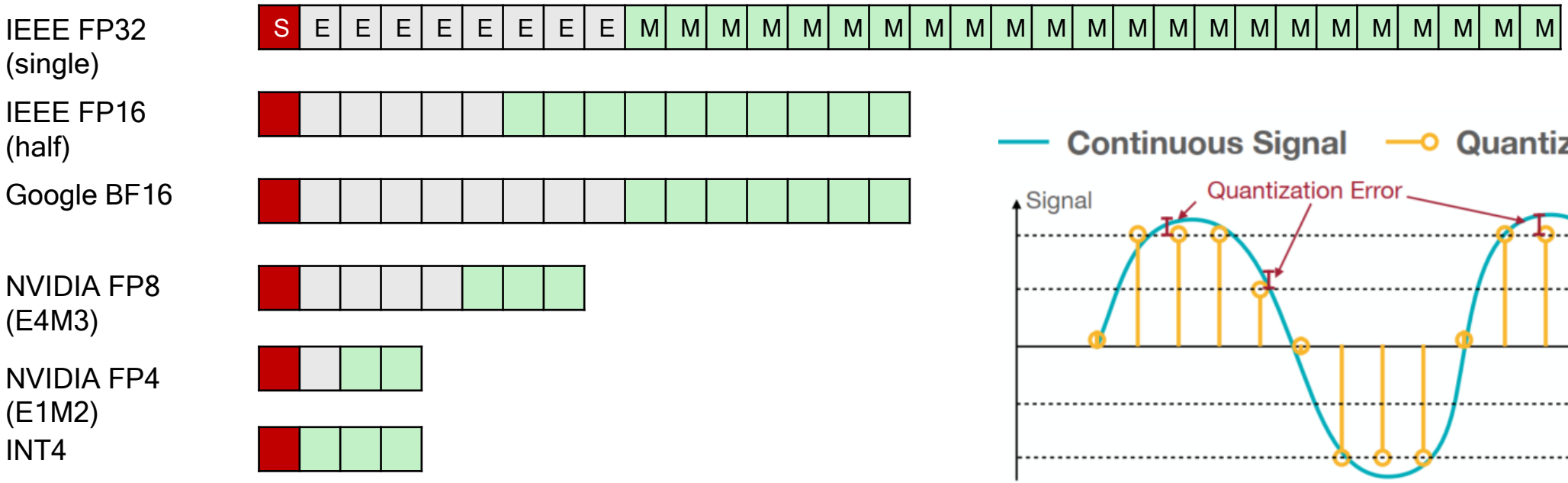
Kernel optimization is more hardware-constrained and usually speeds up the computation without changing the math.

Quantization

VINROBOTICS

Fewer bits, smaller models, faster inference

Quantization reduces the numerical precision of model weights and activations — from 32-bit floats to 8-bit, 4-bit, or even 1.58-bit ternary — shrinking model size and accelerating computation on edge hardware.



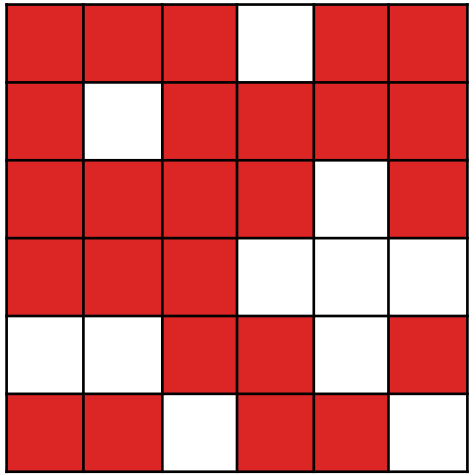
Pruning

VINROBOTICS

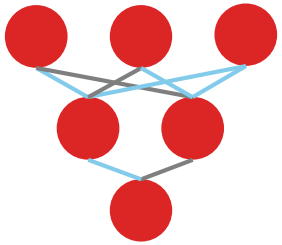
Removing redundant weights without losing performance

Pruning identifies and removes weights, neurons, or entire channels that contribute least to model output. The result is a sparser, smaller network that runs faster — especially when combined with hardware-aware structured pruning.

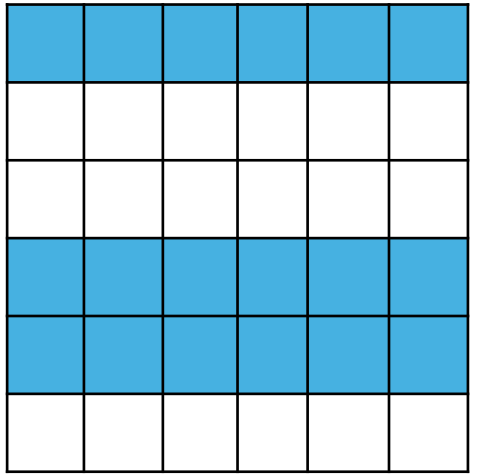
Unstructured Pruning



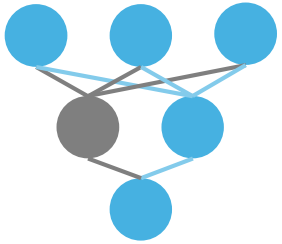
- Remove individual weights
- Requires sparse matrix HW
- Can reach high sparsity



Structured Pruning



- Remove filters, channels, heads
- Hardware friendly
- Lower sparsity



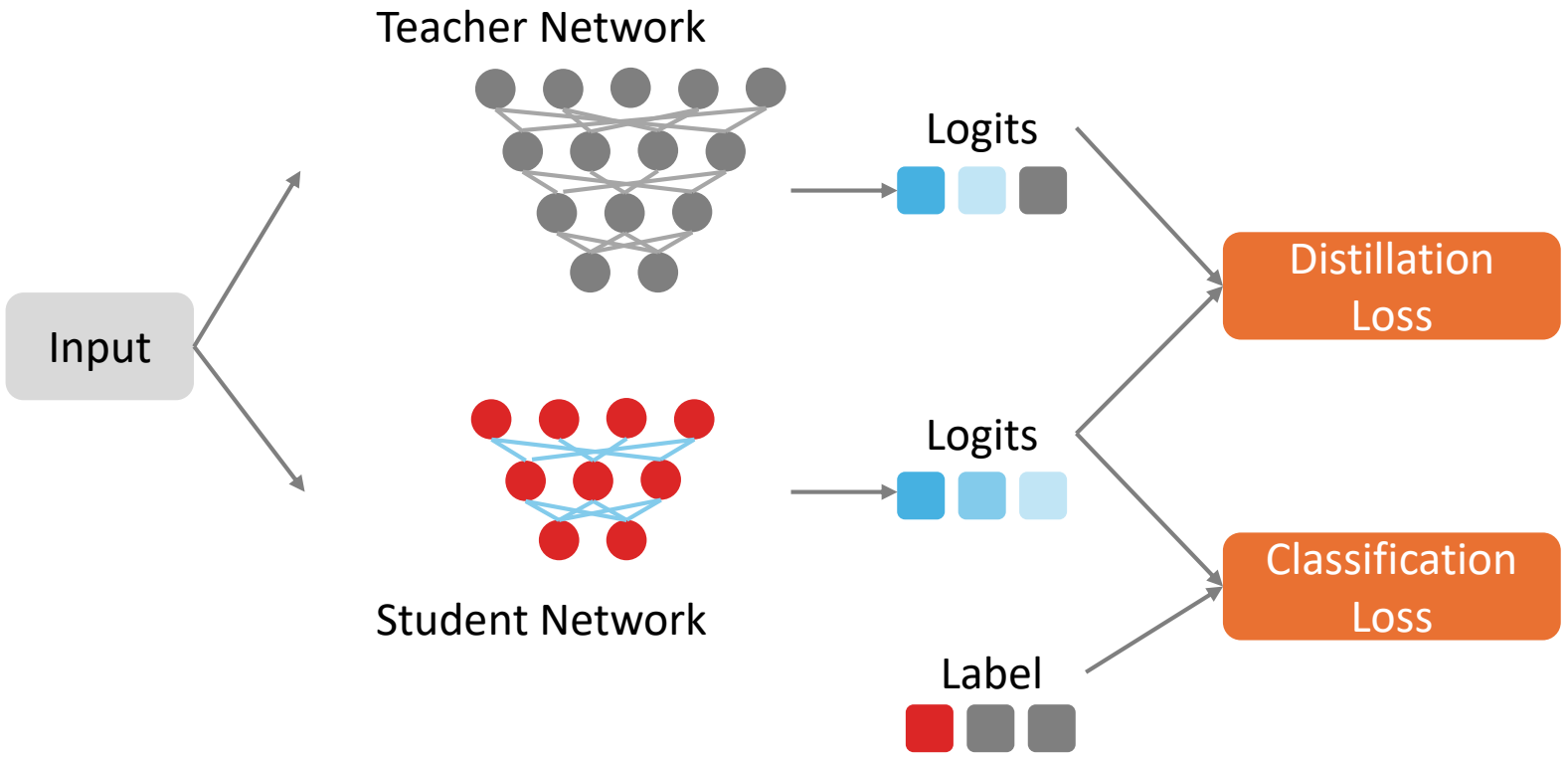
Train
Full model
→
Score
Rank importance
→
Prune
Remove low-rank
→
Fine-tune
Recover accuracy

Knowledge Distillation

VINROBOTICS

Teaching a small model to mimic a large one

A large, accurate "teacher" model transfers its learned knowledge to a smaller, faster "student" model. The student trains on the teacher's soft probability outputs (logits), capturing richer information than hard labels alone. The goal of knowledge distillation is to align the class probability distributions from teacher and student networks.



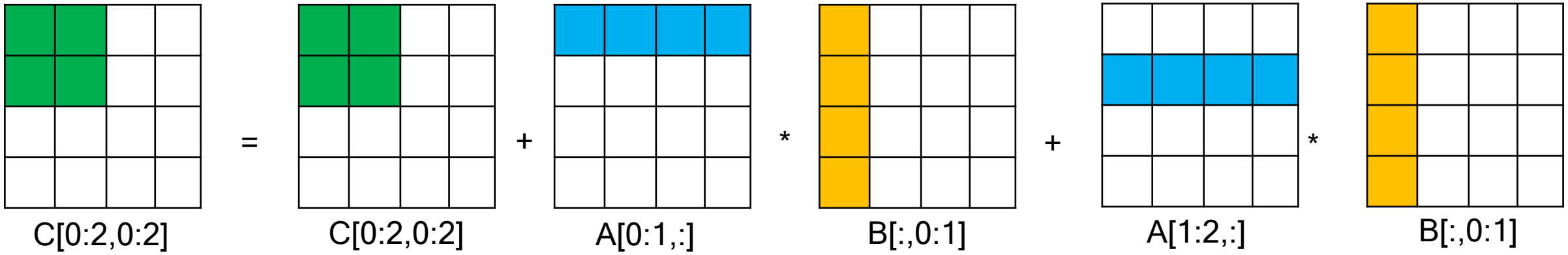
Teacher not only gives **answer**, but also teaches **knowledge**.

Kernel Optimization

VINROBOTICS

Make the same math run faster on the GPU

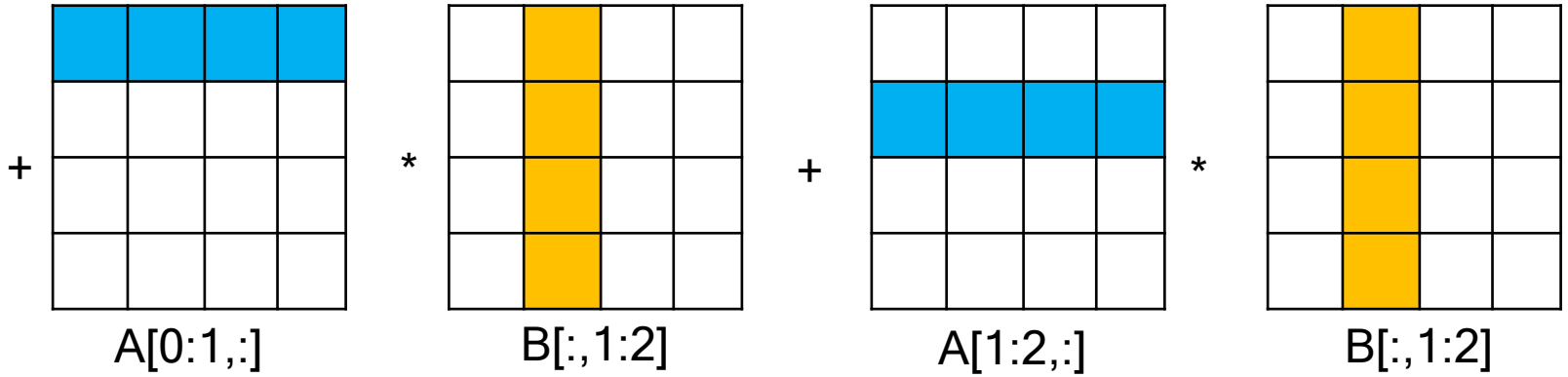
Kernel optimization is more hardware-constrained and usually speeds up the computation without changing the math.
Common techniques: tiling, grid layout, memory access



Example of MAC:

C = C+A*B

(A,B,C are 4x4 matrices)

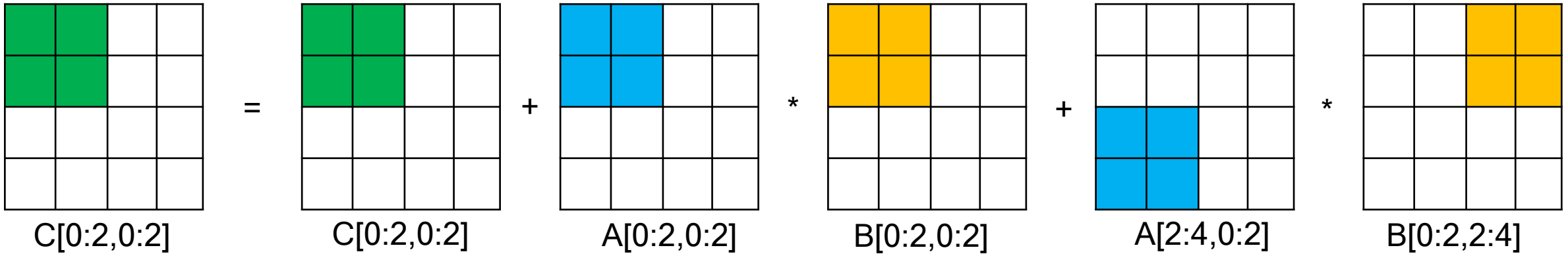


Kernel Optimization

VINROBOTICS

Make the same math run faster on the GPU

Kernel optimization is more hardware-constrained and usually speeds up the computation without changing the math.
Common techniques: tiling, grid layout, memory access



Example of MAC: (block matrix-multiplication)

C = C+A*B

(A,B,C are 4x4 matrices)

Our Research

What we're building and why it matters

VLA deployment engine - vla.cpp

VINROBOTICS

A C++ inference engine for Vision-Language-Action (VLA) models, built on llama.cpp

Github: https://github.com/VinRobotics/vla.cpp Star for us!

Visualization: https://fai-modelopt-tech.github.io/learn-vla-cpp/

HuggingFace: https://huggingface.co/collections/vrfai/vlacpp-model-bundles

Paper: https://arxiv.org/abs/2606.08094

>10
Architecture

Support various SOTA VLAs

From foundation models like Gr00t families, Pi families, SmolVLA to efficient policies like VLA-Adapter, Evo1, BitVLA

>5
Platforms

From Consumer GPU to Edge

NVIDIA products: RTX 30x, 50x, AGX Orin, Orin Nano.
non-NVIDIA products: Intel Arc GPU, Apple Mac Mini.

25%
Compression

New quantization schemes

Reduce precision from FP32 to INT8/INT4. 25% smaller models with minimal accuracy loss.
New quantization scheme: Per-row Quantization

CPU
Deployment

Extreme deployment

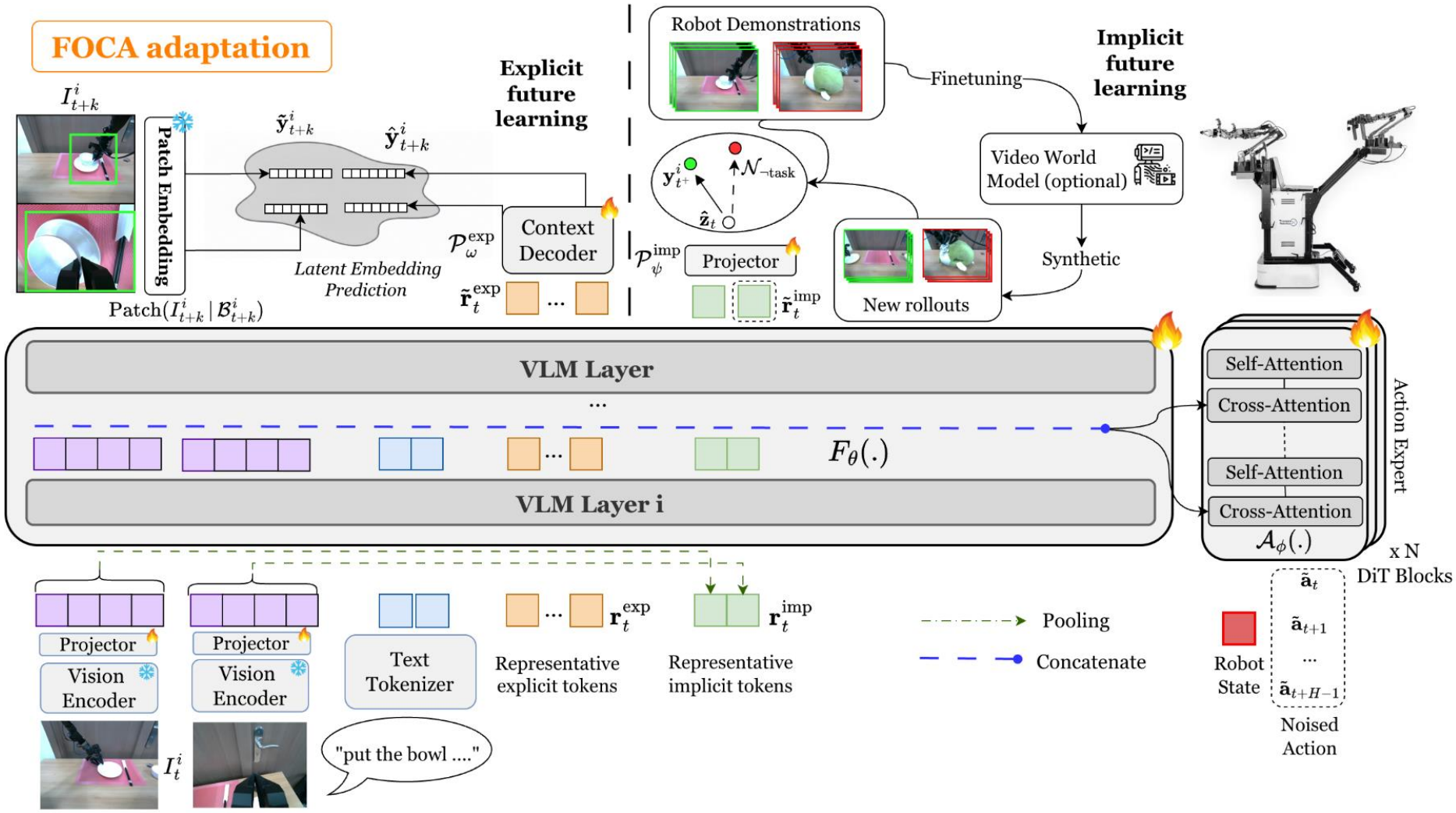
Deploying VLA with CPU is possible with vla.cpp. Tested with MAC M4, Ultrabook with AMD Ryzen7 and Laptop with Intel Core I9

ICML 2026 - FOCA

Future-Oriented Conditioning for Data-Efficient Vision-Language-Action Adaptation

VINROBOTICS

focavla.github.io

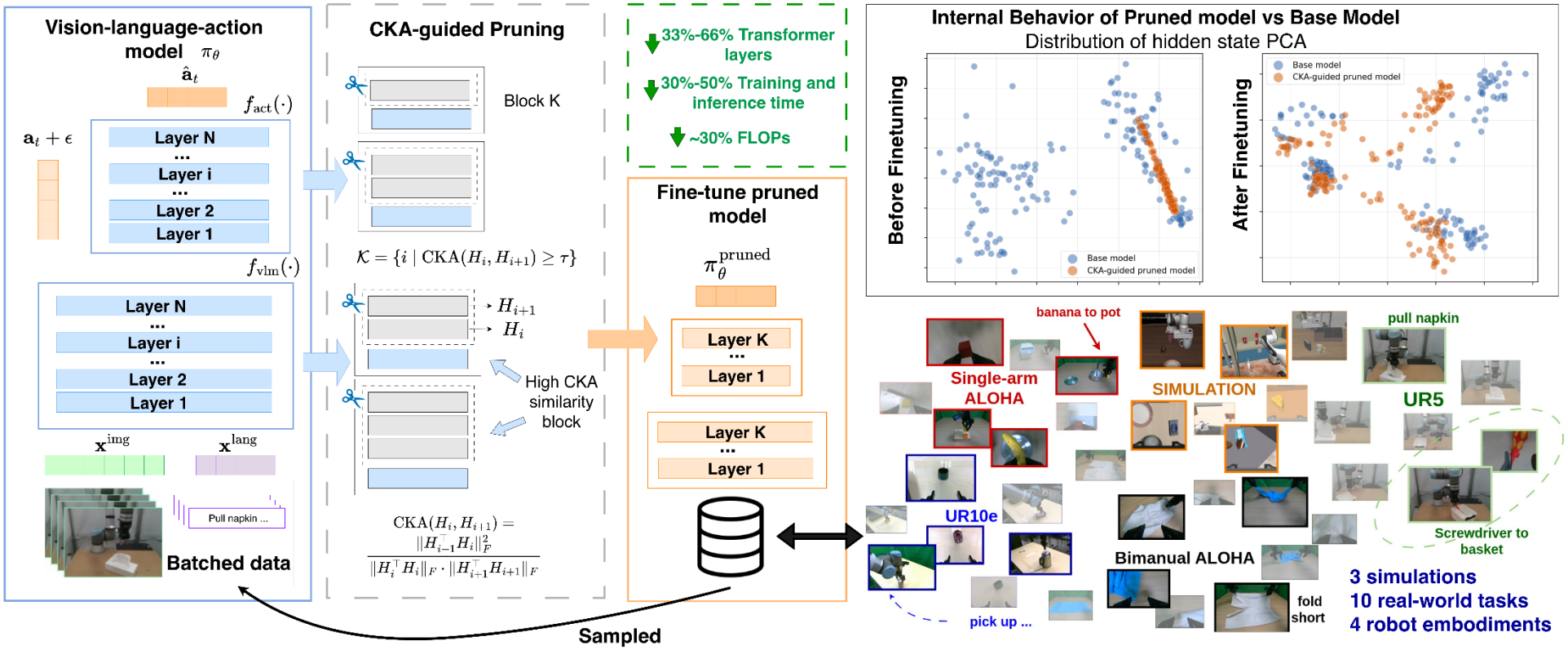


CLP Pruning

Future-Oriented Conditioning for Data-Efficient Vision-Language-Action Adaptation

VINROBOTICS

clpvla.github.io

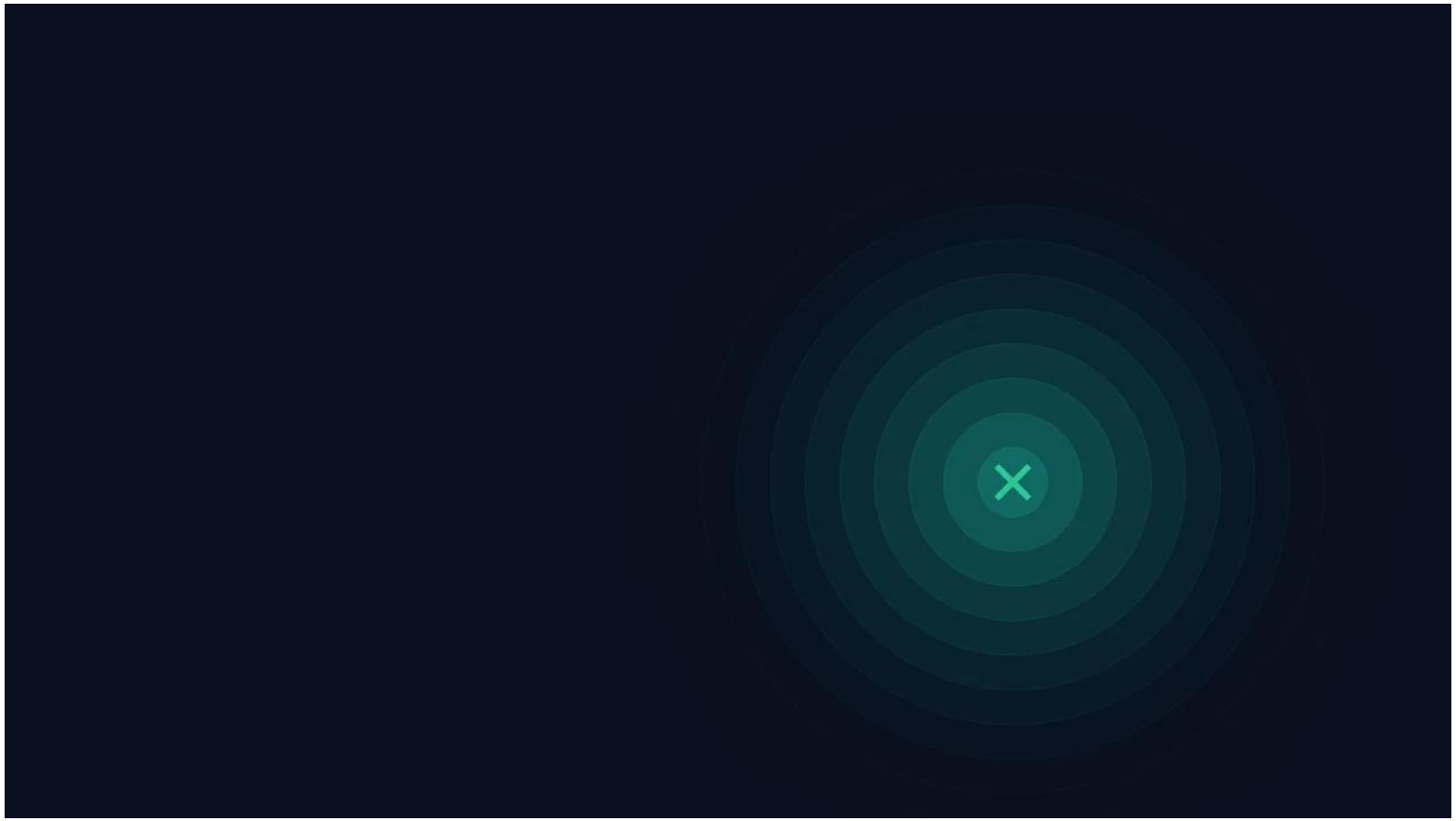


PolyStep

Gradient-free neural network training via optimal transport.

VINROBOTICS

github.com/anindex/polystep



Visualization by: https://vietngth.github.io/polystep-visualization/

VINROBOTICS

Optimization framework - VLAOpt

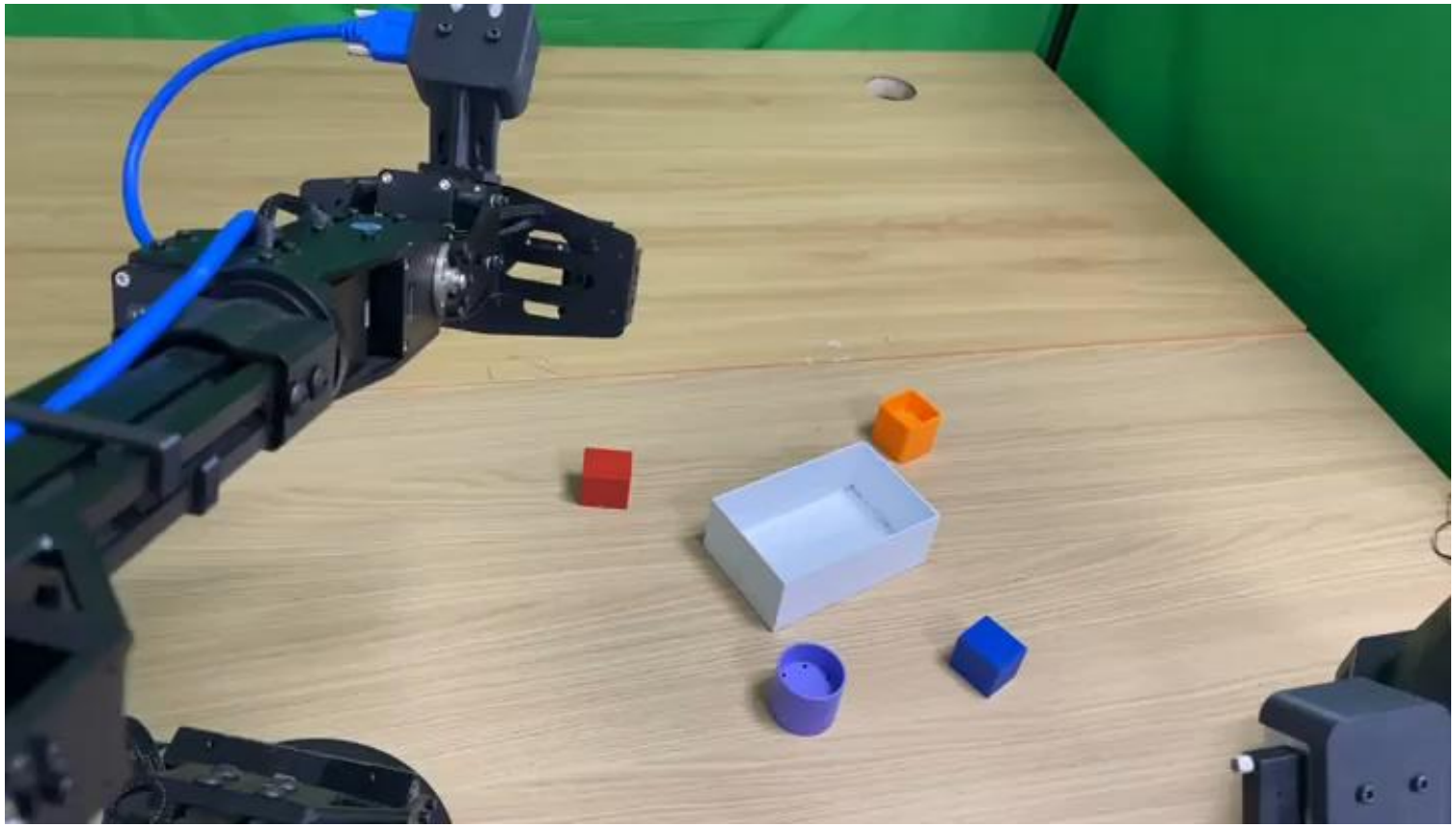
A C++ inference engine for Vision-Language-Action (VLA) models, built on llama.cpp

VLA
Gr00t-N1.6-**3B**

5Hz
Inference

INT8
Quantization

95%
Quality



Finetuned by IsaacGr00t. Deploy with ALOHA with Jetson AGX Orin

VINROBOTICS

MeanFlow with Steering

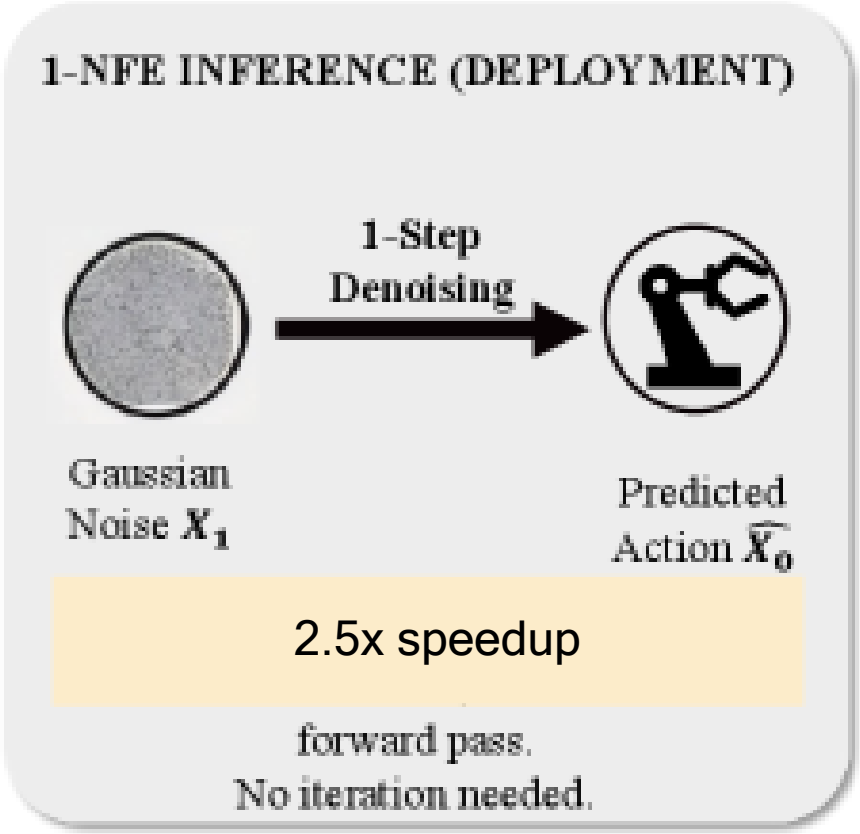
Distill FlowMatching policy to one solver step with Offline Reforcement Learning

95%
Success Rate

17GB
Training VRAM

480M
Parameter

57ms
Inference Time



Tested with SmolVLA. Figure borrowed from SnapFlow

HANDS-ON

Your turn!

MODEL OPTIMIZATION 101

Model Optimization on Edge Deivce

VINROBOTICS

#	Exercise	One line	Core to focus on
1	Quantization	Use fewer bits to represent numbers	quantization scheme & formula, compression ratio
2	Knowledge Distillation	A big model teaches a small one	distillation loss formula, teacher-student architecture
3	Pruning	Remove unused signals	granularity, ratio, criteria
4	Neural Architecture Search	Search for an optimal architecture	search space & strategy
5	Kernel Optimization	Make the same math run faster on the GPU	tiling, grid layout, memory access

Star -> fork -> do the exercises -> commit

The end

Thank you!

Contact: https://www.linkedin.com/in/khanhnd185