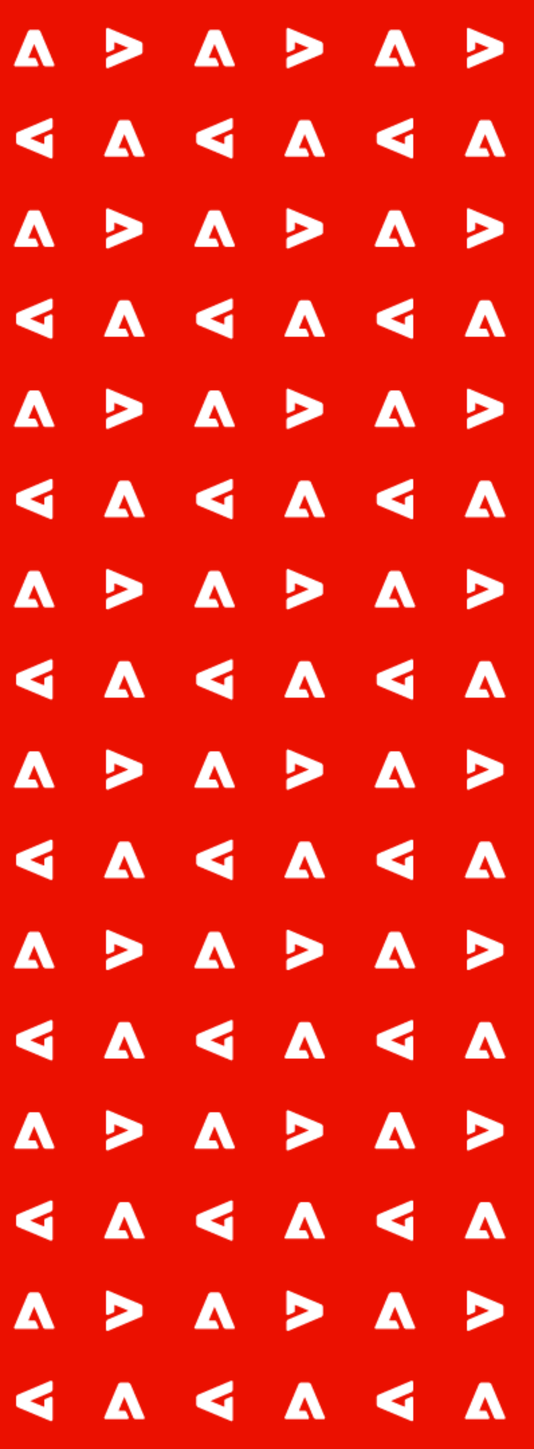
High-performance ML for video & audio

Nikhil Kalra | Sr. Software Engineer

nikalra@adobe.com

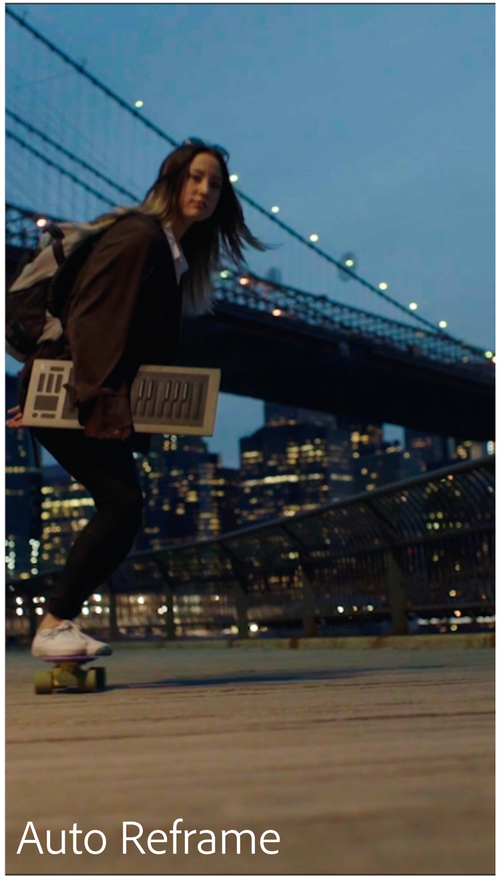


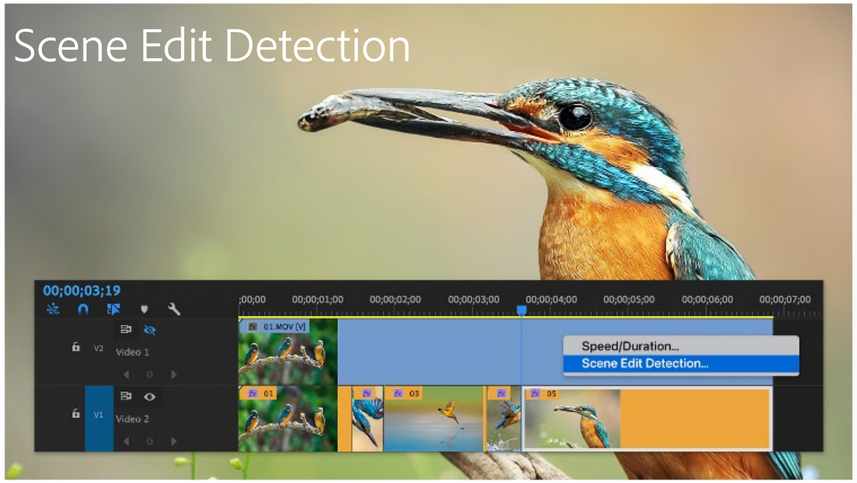
Adobe DVA

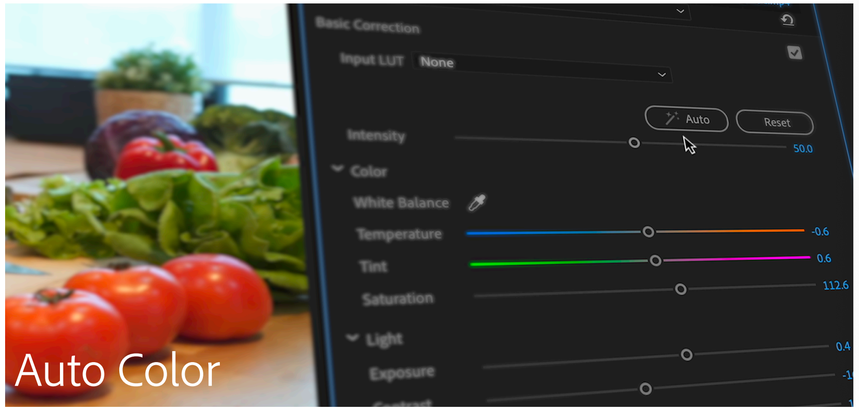
- Digital video and audio for Creative Cloud:
  - Premiere Pro (video editor)
  - After Effects (compositor)
  - Audition (audio editor)
  - Character Animator (animation)

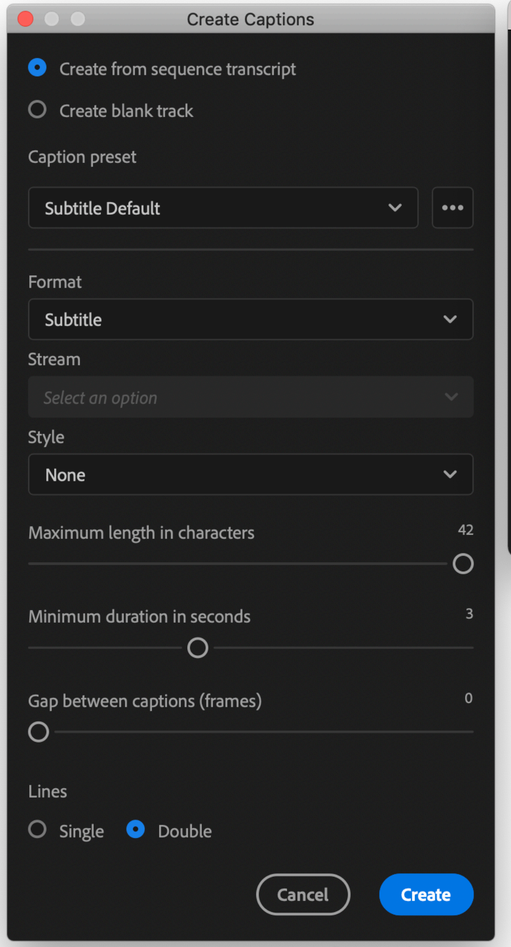
- ML workflows enable us to empower creators by reducing the amount of time spent in redundant and repetitive work so that more time can be spent on the edit and the creative process.

Some of our features…

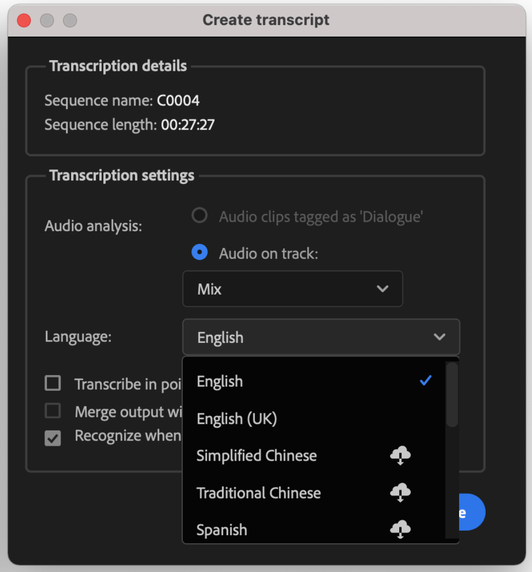








Auto Captions

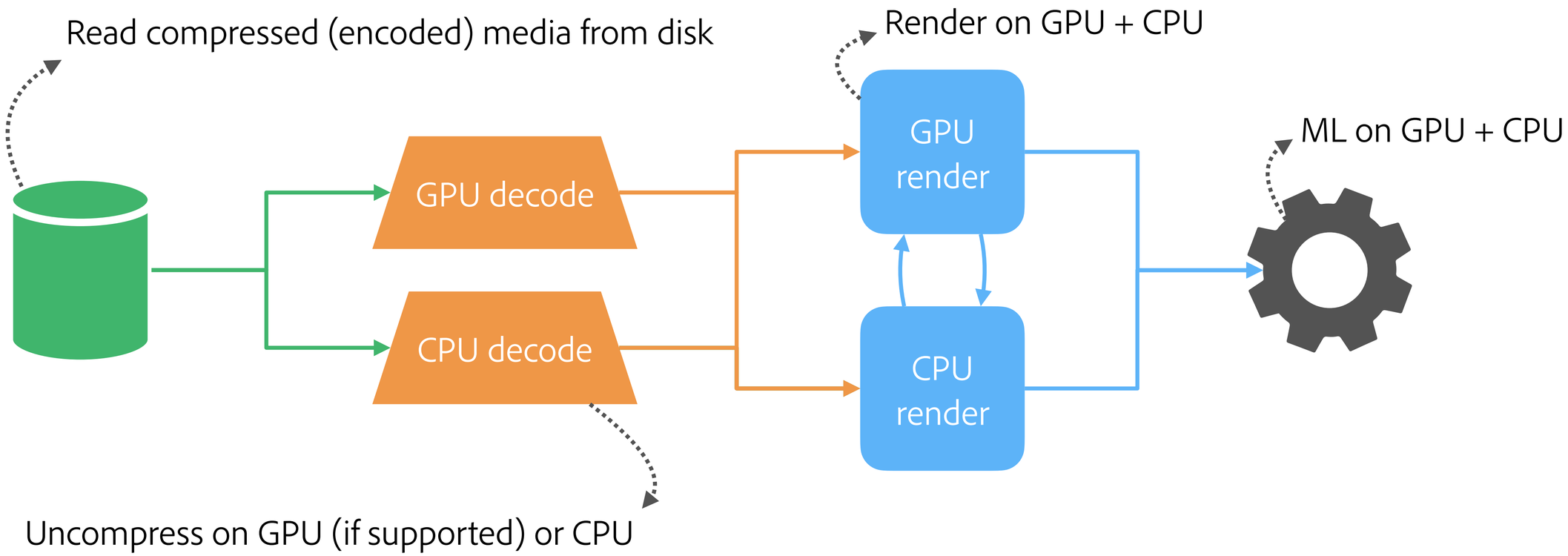


Transcription

What makes video workflows unique?

- Inherently resource limited
  - CPU+GPU concurrently power **decode**, **render**, **encode**, and **ML inference**.
  - Biggest bottleneck is PCI bus saturation
- Data intensive
  - Common formats include 4k @ 60fps and 8k @ 30fps — 6 GB/s of uncompressed footage
- Compute heavy
  - At minimum, every ML workflow requires the render pipeline to resize or re-layout a frame

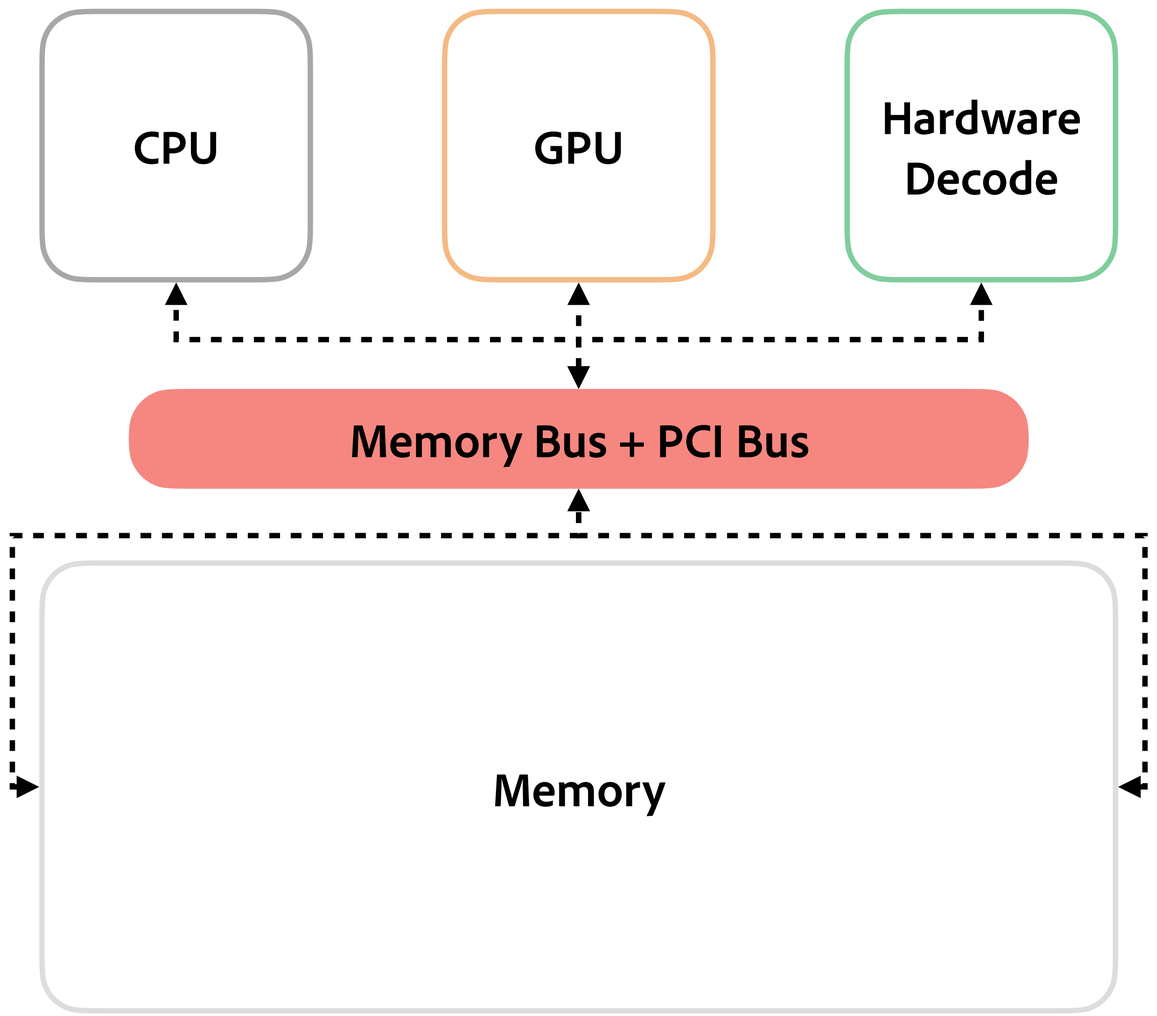
Video processing pipeline for ML



Bottlenecks

- Pr has a lot of concurrent GPU activity running on various technologies
  - Hardware decode (NVDEC, Intel Media SDK, AMD Media SDK)
  - Render (Cuda, OpenCL)
  - Machine Learning (DirectX / DirectML)

- Significant impact on the shared memory bus and/or PCI bus



ONNX Runtime

- Powers ***all*** ML workflows on Windows
  - DirectML EP is used for GPU acceleration
  - Targets a wide variety of hardware with minimal engineering & support overhead

DirectML EP

- On Windows, the Creative Cloud apps target the entire ecosystem as a single platform
  - Feature parity across all major IHVs that support Windows (Nvidia, Intel, AMD)
  - Functional parity across equivalent hardware from different IHVs

- Deploy ONNX models on all hardware in a common runtime vs adopting CuDNN + OpenVINO + MIGraphX

- Performance: Metacommands enable use of vendor specific hardware
  - e.g. TensorCores on Nvidia GPUs
  - Leaves the FP and ALU hardware free for other async compute workflows including render

ORT + DML EP

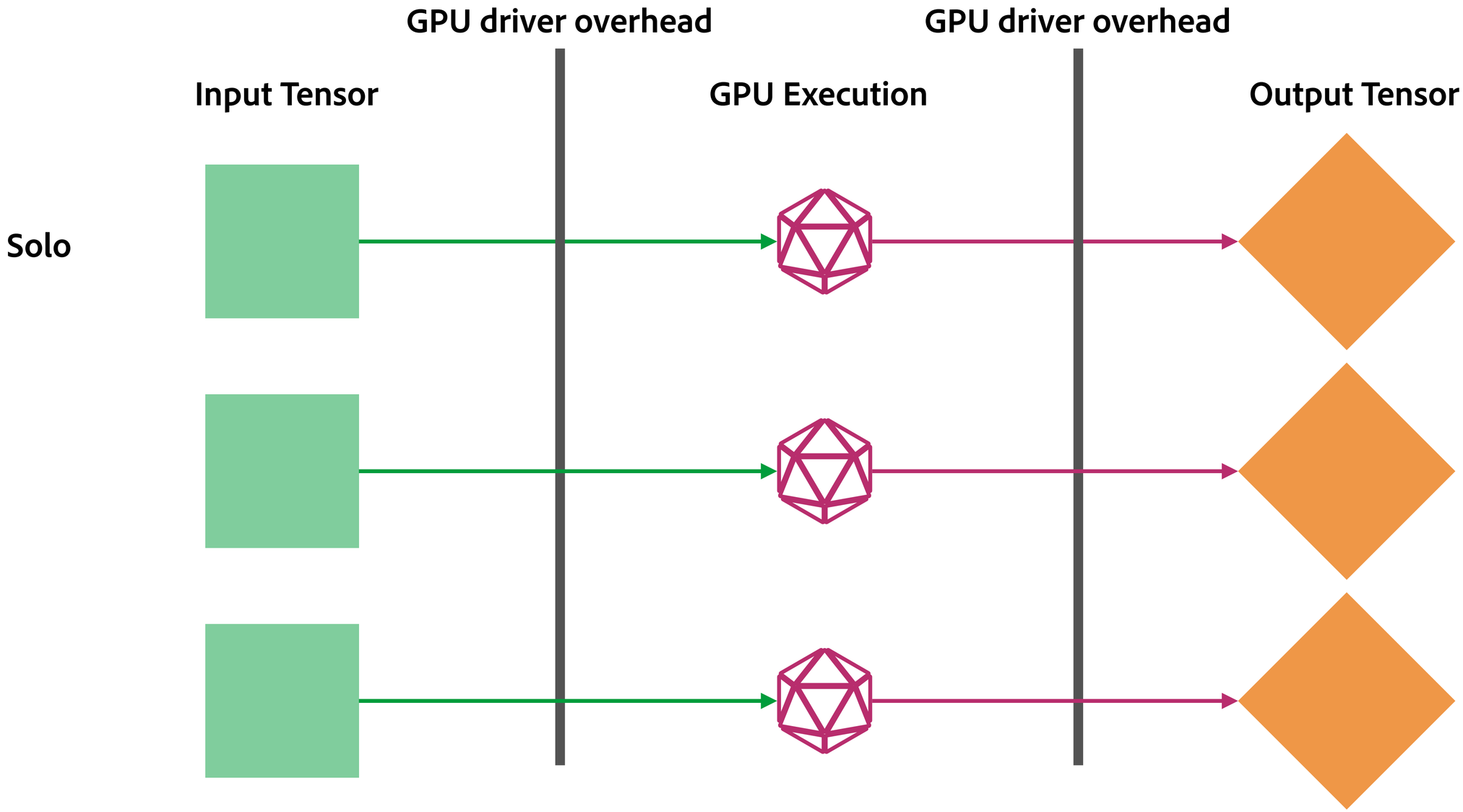
- ORT's historical bottlenecks
  - No GPU input support: required a GPU -> CPU read-back to pass the frame buffer to ORT
  - Batched workflows require contiguous memory buffer forcing additional memory copies

- New DML bindings in ORT have been essential for enabling performant workflows
  - GPU frames can be used as inputs to the inference pipeline using Cuda/OpenCL -> DX12 interop
  - Batches can be assembled async on the GPU using DX12 copy queues increasing available bus bandwidth for the application
  - IOBindings enable DML output buffers to be directly consumed on DX12 enabling the use of ML-based models for GPU workflows

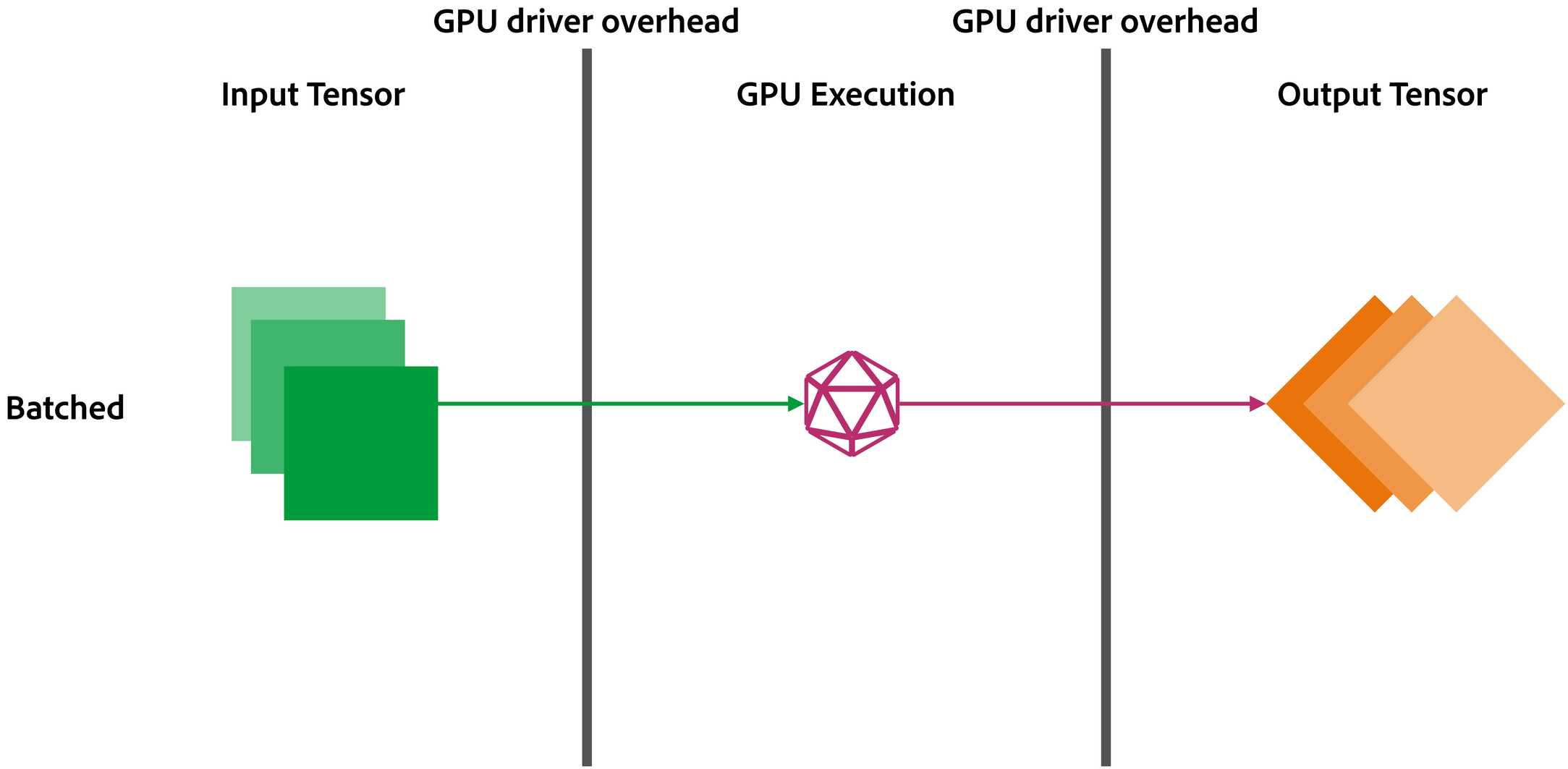
Performance Optimizations

- Frameworks on top of ORT to allow efficient interop with everything else running in the application
  - Render & decode always have the highest priority
  - Back-off inference when the GPU is busy or starved for resources (GPU memory)
  - Assembles inference requests into batched workflows reduce resource contention with the GPU and minimize driver overhead

Inference: solo execution



Inference: batched execution



Let’s look at Scene Edit Detection…

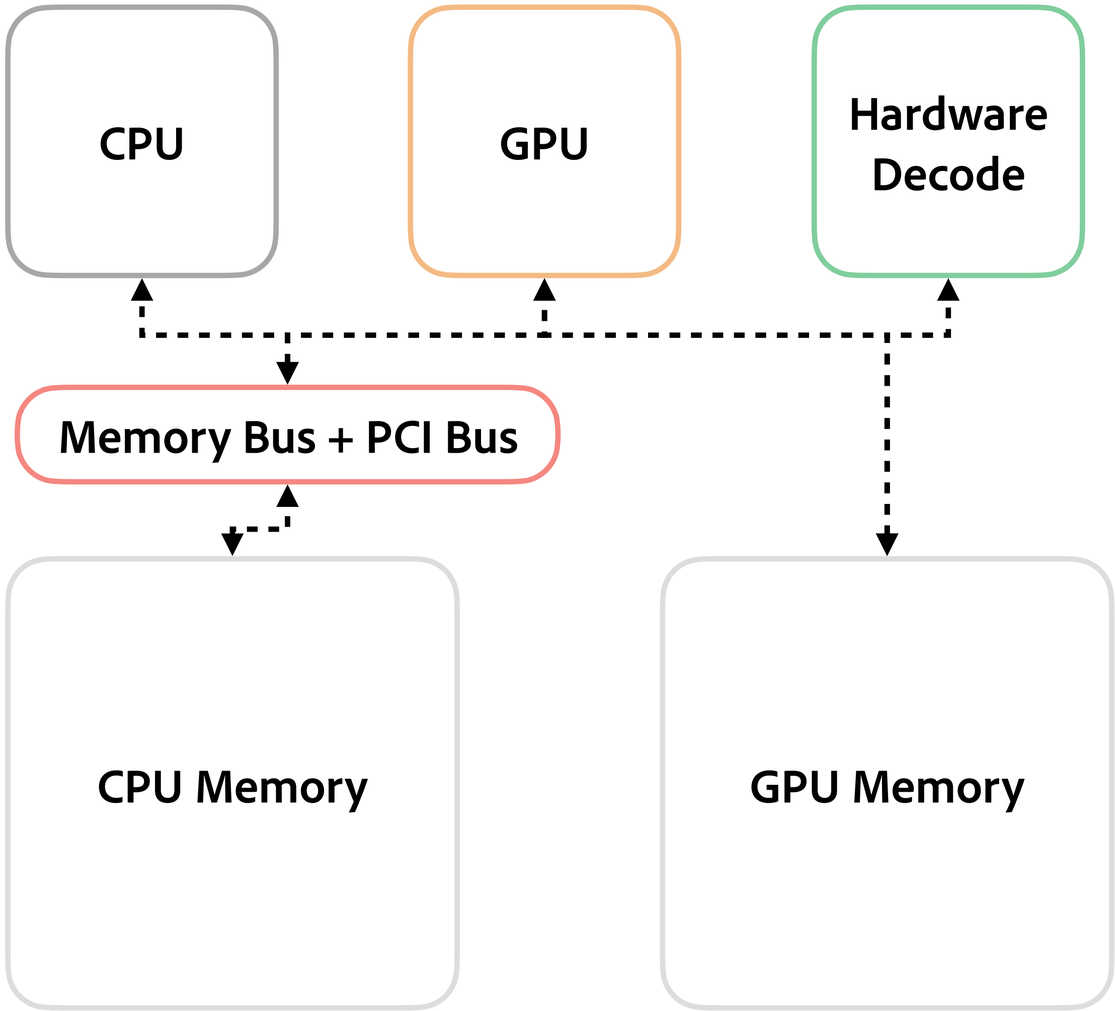
- Decode and renders every frame in a clip
- Runs each of those frames through a network to determine if there is a cut between 2 frames
  - Automates a lot of manual work in common color grading and rendition workflows

- Extremely resource intensive
  - For a **1 minute 4k @ 60fps** video, this results in **3600** inference requests on a total of **6GB** of video data
  - Runs in ~10s or **~6x** realtime

Looking forward: video

- Leverage ORT to enable ML-based workflows during GPU render

- As more of our GPU compute transitions to DX12…
  - Minimize cost by removing the overhead associated with OpenCL/Cuda interop




Adobe