GLENFINNAN: SmartNIC-Accelerated Data Processing for Efficient Vision AI Pipelines

Mike Wong
Princeton University

Ulysses Butler
New York University

Emma Farkash
Princeton University

Praveen Tammana
Indian Institute of Technology Hyderabad

Anirudh Sivaraman
New York University

Ravi Netravali
Princeton University

Abstract

Modern AI vision deployments behave like continuous dataflow systems: thousands of camera streams require repeated data processing on the CPU before any neural network can run on the GPU. In multi-model DAG pipelines, these data processing steps multiply across stages, consuming significant CPU cycles and leaving GPUs underutilized. The CPU-bound nature of these tasks limits overall throughput, increases latency, and forces costly over-provisioning.

GLENFINNAN is a SmartNIC-accelerated data processing system that restructures common image transformations into a streaming, packet-local, lookup-table–driven pipeline. By exploiting pixel locality, memoizing expensive arithmetic, and maintaining only minimal per-packet state, GLENFINNAN performs key data processing operations directly on the NIC. Across six representative pipelines, including CNNs and multi-model DAGs, it reduces CPU utilization by up to 91%, lowers estimated data processing latency by 40–50%, and supports 2.1–3.4× more concurrent streams per node while consuming only 6–14% of NIC logic resources. These results show that programmable NICs can act as first-class compute stages for vision workloads, reshaping system bottlenecks and improving cluster-wide efficiency.

ACM Reference Format:
Mike Wong, Ulysses Butler, Emma Farkash, Praveen Tammana, Anirudh Sivaraman, and Ravi Netravali. 2026. GLENFINNAN: SmartNIC-Accelerated Data Processing for Efficient Vision AI Pipelines. In *ACM SIGCOMM Workshop on Networks for AI Computing (NAIC '26), August 17–21, 2026, Denver, CO, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3789240.3828735

1 Introduction

Modern vision applications, from large-scale video analytics to autonomous robotics, continuously ingest high-resolution image streams that must be transformed before neural network processing can occur. Every frame must undergo a specific sequence of operations—resizing, cropping, and normalization—to match the tensor layout of the target model. As deployments scale toward thousands of streams and complex multi-model Directed Acyclic Graphs (DAGs) [6, 14], the cumulative overhead of this data processing becomes a cluster-wide bottleneck.

Today, these image transformations are overwhelmingly executed on the host CPU. While GPUs are a natural target for acceleration, they are already oversubscribed by primary inference workloads, such as large-scale Vision Transformers (ViTs) and Multimodal LLMs. Offloading preprocessing to the GPU reduces its effective inference capacity, introduces redundant PCIe transfers of raw pixel data, and complicates kernel scheduling. Consequently, CPUs absorb the majority of this work, often consuming up to 95

Given this pressure, we ask: *can data processing be performed earlier in the data path, before frames reach host memory?* Every vision pipeline begins with frames arriving over the network. Modern datacenter interfaces, particularly FPGA-based SmartNICs, provide an opportunity to perform computation directly within the ingress pipeline. Executing transformations at the NIC avoids host-level memory traffic and yields model-ready tensors immediately upon arrival.

At first glance, image transformations appear ill-suited for the restricted environment of a NIC. These operations seemingly require full-frame context, floating-point (FP) intensity, and large memory buffers—resources that SmartNICs lack. However, we identify that the core operators of vision pipelines exhibit data-independent behavior and strict spatial locality. They apply deterministic arithmetic to small, fixed pixel neighborhoods using predetermined constants (e.g., interpolation weights).

We present GLENFINNAN, a SmartNIC-accelerated system that restructures image transformations into a streaming, packet-local pipeline. Unlike neural network inference, which requires high arithmetic intensity and massive parameter storage, vision preprocessing is structurally compatible with the deeply pipelined, deterministic execution model of FPGA-based SmartNICs. GLENFINNAN introduces three key architectural techniques:

- **Localized Windowing:** We implement a "sliding-window" pixel buffer that recovers spatial semantics from streaming chunks, avoiding the need to reassemble full video frames on-chip and thus bypassing BRAM capacity limits.
- **Memoized Arithmetic:** To address the high latency and resource cost of FPGA floating-point units, we replace expensive interpolation and normalization logic with area-efficient, lookup-table-driven (LUT) computation.
- **Boundary-Aware State Management:** We design a lightweight buffering scheme to maintain transformation state across packet boundaries, ensuring correctness even when pixel neighborhoods span multiple network packets.

By offloading preprocessing to the NIC, GLENFINNAN rebalances the serving substrate. Across six representative pipelines, including single-model CNNs and multi-model DAGs, GLENFINNAN reduces CPU utilization by up to 91

2 Background and Motivation

2.1 Model Serving and Image Processing

Modern vision applications rely on large-scale model serving infrastructures that process thousands to millions of image frames per second. A typical system accepts frames from remote devices, performs data-independent transformations (e,g., resize, crop, tensor conversion, normalization), and dispatches the resulting tensors to one or more neural networks. Although the ML models themselves are diverse—ranging from classical CNNs to vision transformers and VLMs—the *serving substrate* is remarkably uniform: every frame must be converted into the exact tensor representation the model was trained on.

For example, an ImageNet-style classifier typically applies `Resize(256)` (bilinear downsampling), `CenterCrop(224)`, `ToTensor` (uint8 → float scaling and layout conversion), and `Normalize` (per-channel affine transform) before inference. These operations are mandatory and run for every frame, on every server, across every stream. Multi-model pipelines compound this cost: in a DAG with an object detector followed by a classifier, data processing occurs both at ingress *and* between stages, as bounding boxes must be cropped, resized, and normalized before downstream models can operate. As pipelines evolve to combine segmentation, re-identification, text–image embedding, and generative components, the transformations required to chain together these models grow accordingly.

2.2 Data processing as a System Bottleneck

While data processing is conceptually straightforward, its cumulative cost makes it a major obstacle in practice. Model serving platforms have focused on GPU scheduling, batching, and concurrency management [8, 27, 36], but our measurements show that CPU-side data processing increasingly dominates CPU resource consumption. Figure 1 illustrates this effect on a 32-core Intel Xeon server: even a single 40–50 FPS stream consumes a substantial fraction of CPU time purely for data processing. As frame rates increase, as more streams are multiplexed, or as DAGs deepen (Figure 2), CPU utilization rises sharply.

At the same time, GPUs in production serving environments are heavily exercised. They run large batches of inference jobs, serve multiple models simultaneously, and frequently share memory among models using mechanisms such as CUDA MPS [20]. These accelerators assume a steady supply of inputs; any variability in data processing directly affects GPU efficiency by forcing smaller batches, longer inter-batch gaps, or increased kernel scheduling overhead.

Scaling CPU resources is an increasingly unattractive solution. Additional cores contend for memory bandwidth, shared last-level cache capacity, PCIe traffic, and CPU–GPU synchronization. Even servers with dozens of cores experience congestion when processing multiple high-resolution video streams or deep multi-model pipelines [1, 11].

These considerations highlight an alternative opportunity: data processing need not occur on the CPU at all. Every frame already traverses a network interface before it reaches the CPU or GPU. Modern network adapters, particularly FPGA-based SmartNICs, offer deeply pipelined execution, per-packet processing, and reconfigurable logic that can execute structured computations. Offloading data processing to the NIC has several advantages. First, it eliminates redundant data movement: instead of transferring raw frames to the CPU for preprocessing before forwarding tensors to the GPU, the NIC can produce model-ready tensors immediately upon packet arrival, reducing host memory traffic and PCIe transfers. Second, because preprocessing executes before GPU inference begins, it preserves GPU compute resources for neural network execution rather than competing with inference kernels for streaming multiprocessors and memory bandwidth. Finally, NICs provide compute resources "for free," since every server already requires a NIC; enabling their programmability adds computational capacity without proportionally increasing cluster cost.

2.3 Challenges

Despite the potential benefits of NICs for data processing, several challenges make realizing this vision non-trivial.

Challenge 1: Packetized execution vs. image-level semantics. Programmable NICs process packet data as small chunks (e.g., 32–64 B AXI records for many Xilinx platforms). Pixel neighborhoods required for resize or crop often span packet and chunk boundaries. Reassembling entire video frames on-chip would exhaust BRAM or stall the pipeline, significantly lowering throughput. A NIC design must recover image-level semantics from streaming chunks.

Challenge 2: Compute limitations and costly arithmetic. Floating-point arithmetic is expensive on FPGA NICs: FP add takes 4 cycles, FP divide up to 35 cycles (Figure 3, and both require significant LUTs and flip-flops (Figure 4). NIC pipelines run at modest clock speeds (200–300 MHz), achieving throughput only through pipelined or data parallelism. Incorporating naive FP operators prevents sustaining 100 Gbps data rates.

Challenge 3: Severe memory constraints. FPGA SmartNICs typically provide only tens of megabits of on-chip BRAM, far too little to store entire video frames or large intermediate images. Even modest lookup tables can exceed available BRAM, and off-chip DRAM is too slow for per-packet processing. Designs must rely on tiny window buffers and carefully budgeted on-chip storage.

3 Design

3.1 Overview

A naïve FPGA implementation would buffer an entire image on-chip before executing any data processing, but this approach conflicts with SmartNIC constraints: on-chip memory is limited, buffering introduces stalls, and full-frame storage requires off-chip DRAM, which severely degrades throughput.

Instead, we introduce GLENFINNAN, a SmartNIC-based accelerator for vision data processing that restructures image transformations into a deeply pipelined data path. The design is guided by a central observation: although data processing operations are defined over full images, *their computations depend only on small, fixed neighborhoods of pixels*. For example, bilinear interpolation in `Resize` computes each output pixel from exactly four spatially adjacent input pixels; `CenterCrop` simply selects a rectangular region without requiring global context; `ToTensor` performs a deterministic channel-wise reshaping and scaling; and `Normalize` applies an affine transform to each pixel independently. Figure 5 illustrates the resulting architecture: frames arrive at the NIC, are decoded

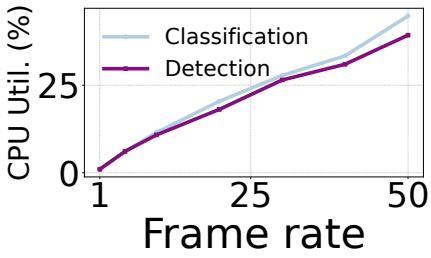


Figure 1: CPU usage of processing operations as frame rates increase.

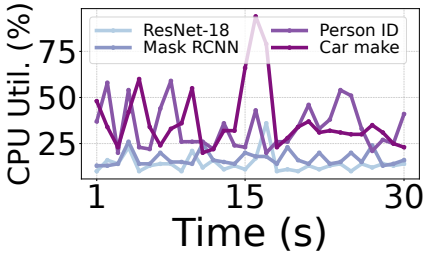


Figure 2: Data processing overhead increases with DAG depth.

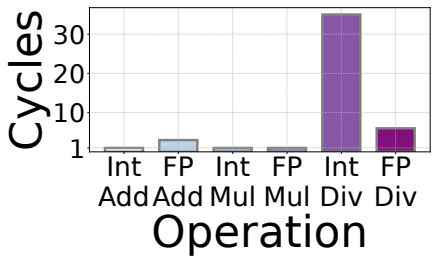


Figure 3: Cycle-level latency of arithmetic operations on FPGA.

Operation	DSP	FF	LUT
Int add	0	0	39
FP add	0	231	220
FP multiply	3	130	139
Int multiply	3	0	21
Int divide	0	2319	1740
FP divide	0	318	793

Figure 4: FPGA resource consumption for arithmetic operators.

into block-aligned pixel chunks, flow through GLENFINNAN's sequence of localized data processing stages, and emerge as GPU-ready tensors without ever traversing the CPU. Depending on the system configuration, this transfer may traverse the standard PCIe fabric connecting the NIC and GPU, or it can use a direct data path such as GPUDirect [21], which bypasses the CPU to reduce latency and memory copies. Both options are architecturally supported: in deployments with GPUDirect, the NIC can stream data directly into GPU memory, while in systems without GPUDirect, the PCIe path ensures compatibility with conventional host-mediated transfers. In multi-model pipelines, intermediate data processing stages can also be executed on the NIC, avoiding CPU involvement between models.

Our approach resolves the mismatch between image-level semantics and NIC-level packet processing by exploiting two key properties. First, operations exhibit strong **locality**: each stage reads only a bounded, localized window of pixels at a time, and no stage requires global frame context. Second, data processing exhibits **data-independence**: the sequence of transformations is fixed at deployment, allowing for specialization, memoization, and aggressive pipeline optimizations. GLENFINNAN leverages these characteristics to map image-level operators onto the packetized execution model of FPGA SmartNICs.

Rather than buffering whole frames, GLENFINNAN processes images *as they stream through the NIC*, operating directly on packetized pixel data produced by the JPEG decoder. Each hardware stage implements one transformation (e.g., `Resize`, `CenterCrop`, `ToTensor`, `Normalize`), consuming only the pixels available in the incoming packet while maintaining just enough state to cover pixel neighborhoods that span packet boundaries. Arithmetic that would traditionally require expensive computation is replaced with compact lookup-table evaluations tailored to the fixed geometry of the pipeline. As packets flow downstream, stages operate concurrently, providing pipeline parallelism, and process multiple pixels per cycle, providing data parallelism. The result is a modular, packet-parallel data processing pipeline that fits comfortably within the compute and memory constraints of commodity SmartNICs while delivering efficient, continuous streaming execution.

3.2 Match-Style Processing via Memoization

SmartNICs excel at table-driven, match/action execution but struggle with floating-point arithmetic, division, and other operations common in image processing. GLENFINNAN converts each processing function into a series of table lookups. This strategy leverages the fact that vision pipelines are data independent, with fixed input sizes, operations, and model expectations, and that pixel values are discrete (`uint8`), making many computations precomputable.

Before deployment, GLENFINNAN precomputes the results of scaler arithmetic for all relevant inputs and stores them in lookup tables on the NIC. For simple functions such as `ToTensor` or `Normalize`, each possible input pixel value maps to a corresponding output. For more complex functions like bilinear interpolation, GLENFINNAN memoizes each term in the interpolation formula separately rather than the final combined result, exploiting the independence of each pixel's contribution. This decomposition yields substantial compute savings by replacing runtime scaler arithmetic with integer-indexed table lookups, enabling SmartNIC-style processing with minimal logic and latency.

Even after decomposition, naïve tables would exceed on-chip memory limits due to the large range of possible geometric weights. GLENFINNAN compresses these tables by exploiting geometric redundancy and reduced precision. Interpolation weights depend only on pixel position and image geometry, which remain constant across frames. Hence, many rows of the table are identical or reusable across pixels and images. Additionally, while weights are computed in floating point, the final outputs are 8-bit integers, allowing moderately lower precision without sacrificing image quality. These insights reduce required memory by over two orders of magnitude, allowing all lookup tables to reside on-chip and enabling efficient execution.

3.3 Pixel Buffering Across Packets

Some operations require pixel neighborhoods that occasionally span packet boundaries. GLENFINNAN handles these cases with lightweight buffering strategies tailored to each function's locality. For window-based operations like bilinear interpolation, only the few pixels needed to complete the current window are stored and discarded immediately after use. For channel rearrangements such

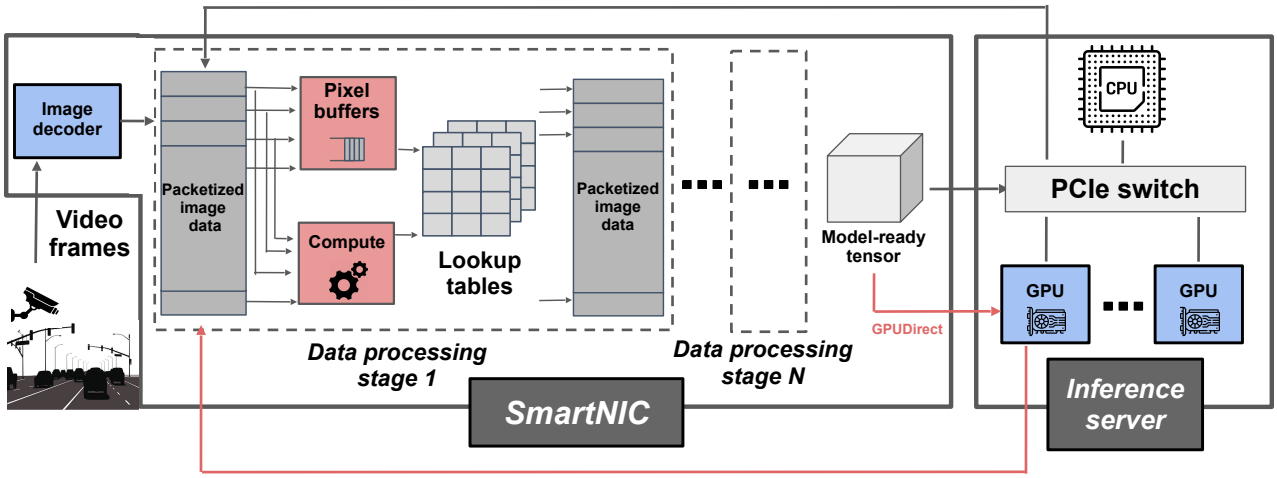


Figure 5: Overview of GLENFINNAN 's design. Incoming frames are decoded on the SmartNIC into raw pixel blocks, which flow through a pipelined sequence of data-processing stages (e.g., resize, crop, normalize). Each stage operates on packetized pixel data in a streaming fashion, producing a model-ready tensor that is delivered directly to the inference server. The tensor can traverse either the PCIe fabric (black arrow) or a direct path to the GPU such as GPUDirect (red arrow), bypassing the CPU in both cases. After model execution, results can optionally return to the NIC for additional processing in multi-model pipelines.

as `ToTensor`, which converts pixels from (H, W, C) to (C, H, W), GLENFINNAN exploits the fixed ordering: pixels for the first channel are emitted immediately, while the other channels are temporarily buffered in off-chip DRAM and pulled on demand. Because this transformation occurs after resizing and cropping, the data footprint remains small and DRAM bandwidth is sufficient. Thus, by tightly bounding the required state, GLENFINNAN avoids global frame reconstruction while supporting all operations required by production vision pipelines.

3.4 Practical Considerations and Limitations

GLENFINNAN demonstrates that a wide range of commonly deployed data processing operations can be restructured into packet-local, streaming transformations. However, not all image operations share this structure. Functions that require global image context, e.g., computing a global `min`/`max`, histogram equalization, or algorithms whose output depends on an accumulated global statistic, are fundamentally incompatible. Other operations, such as `RandomCrop` or `RandomRotation`, require non-local, data-dependent pixel access. These transformations are primarily used for data augmentation during training and are rarely part of real-time inference pipelines.

The broader takeaway is that the transformations most frequently used in production inference (e.g., `Resize`, `CenterCrop`, `ToTensor`, `Normalize`) exhibit strong locality and simple structure. These characteristics make them well-suited for SmartNIC offload. In contrast, full model inference, while an intuitive offload target, remains mismatched to NIC constraints due to large model footprints, irregular memory access patterns, and strict on-NIC resource budgets.

4 Implementation

We implement GLENFINNAN on the NetFPGA SUME platform, a PCIe SmartNIC built around a Xilinx Virtex-7 FPGA with 10/100 Gbps Ethernet [37]. We combine Vivado High-Level Synthesis (HLS) for structured, per-pixel processing pipelines (including arithmetic stages and lookup-table integration) with hand-written Verilog for packet-stream interfaces, buffering, and inter-stage control. Each preprocessing operator mirrors its `torchvision` counterpart and synthesizes into a fully pipelined module with a fixed initiation interval capable of accepting continuous pixel streams; offline-generated lookup tables are embedded in on-chip BRAM or ROM. The resulting design replaces the default SUME packet-processing path with GLENFINNAN 's multi-stage preprocessing pipeline. Correctness is verified through exhaustive RTL simulation against the reference PyTorch/`torchvision` implementations.

5 Evaluation

We evaluate six representative pipelines—ResNet-18 [12], Mask R-CNN [16], RMBG [23], CLIP [24], and two multi-model DAGs for car-make classification and person identification [29]—on a 28-core Intel Xeon Gold 6132 with 2× NVIDIA A6000 GPUs. All use standard `torchvision` preprocessing (resize, crop, tensor conversion, normalization) with 1280×720 inputs. We implement GLENFINNAN on NetFPGA SUME (Xilinx Virtex-7 690T) [37]. CPU utilization is measured by running each pipeline with and without preprocessing enabled; the difference isolates the cost GLENFINNAN eliminates. Latency is derived from hardware-accurate operator synthesis at 100–190 MHz.

5.1 CPU Utilization Reductions

Figure 6 shows that NIC offload dramatically reduces CPU load across all workloads. For fast pipelines where preprocessing dominates—ResNet-18 and CLIP—CPU usage drops from 91.1% to 8.7% and from 93.9% to 5.3%, respectively. Multi-model DAGs see similar reductions: car-make classification falls from 37.4% to 1.4%, and person identification from 33.3% to 1.9%. Even for heavier models

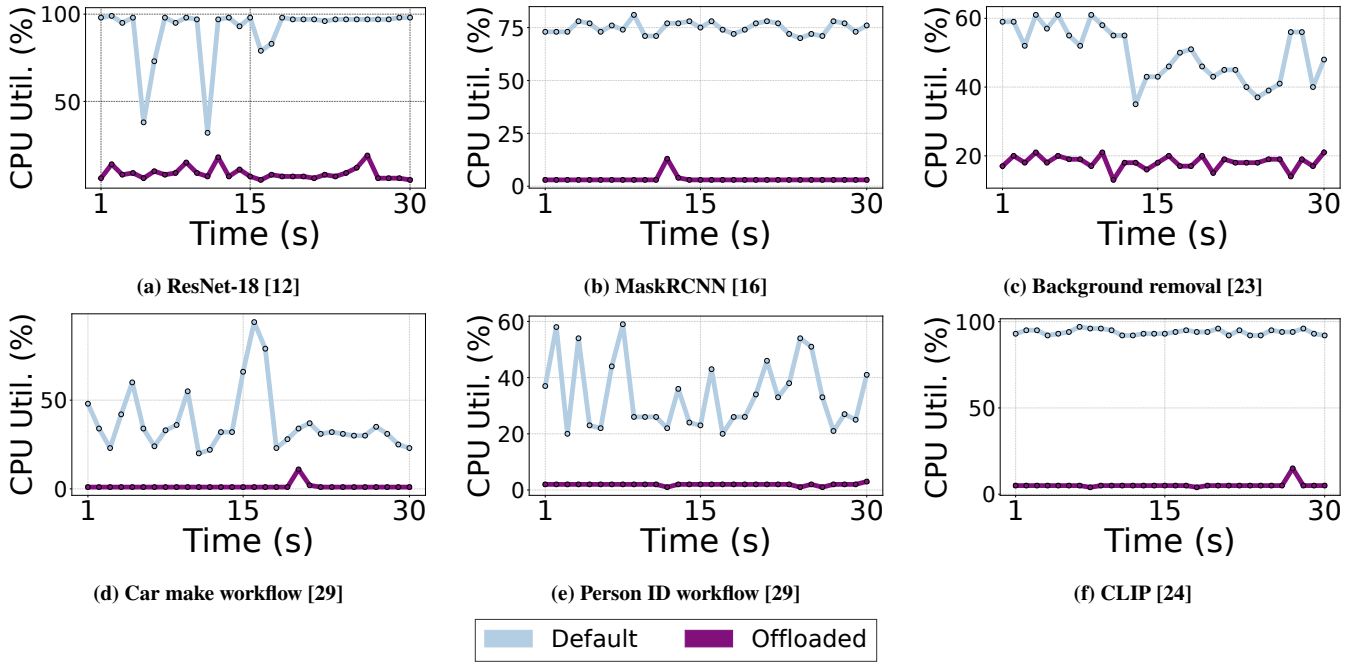


Figure 6: CPU utilization for six representative vision pipelines, with and without NIC-side preprocessing offload.

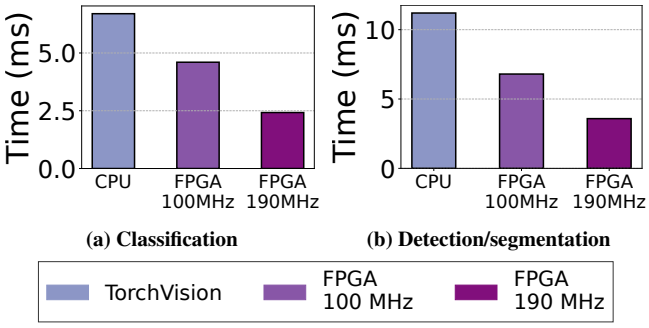


Figure 7: Comparing `torchvision` measurements with hardware-latency estimates obtained from synthesized NIC-side operators.

like background removal, which operate at lower frame rates, offloading cuts utilization from 49.6% to 18.1%, freeing headroom for inference and batching. The residual CPU load in this workload reflects host-side coordination logic outside GLENFINNAN's scope.

By performing data processing at the NIC, GLENFINNAN systematically eliminates the most expensive per-frame CPU work, enabling higher throughput, lower queuing delays, and denser packing of models per host in cluster deployments. The benefits arise from GLENFINNAN's key design principles, packet-local streaming computation and lookup-based transformation, which replace expensive per-frame arithmetic with compact table lookups and execute operations directly on incoming pixel packets. Together, these features allow the SmartNIC pipeline to sustain line-rate data processing while removing a persistent CPU bottleneck.

While the relative impact of data processing offload diminishes for large, high-latency models, the system-wide benefits in utilization and throughput remain substantial. The largest improvements occur in workloads with fast models, high frame rates, or multi-stage pipelines composed of lightweight operators, where data processing constitutes the primary CPU bottleneck and NIC offload directly unlocks higher end-to-end throughput.

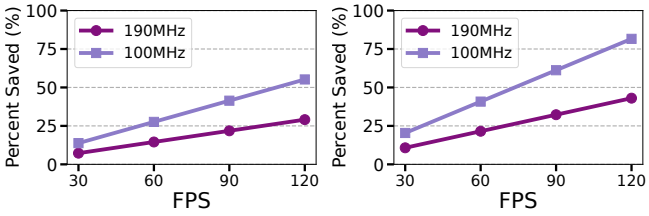
5.2 Image Latency

We next measure processing latency for two representative pipelines: a classification pipeline and a detection/segmentation pipeline. We use hardware-accurate cost models derived from synthesizing each data processing operator to the NetFPGA SUME using Vivado HLS at 100–190 MHz, typical of SmartNIC-class accelerators. We compare these costs against equivalent CPU baselines using PyTorch and `torchvision` on a 32-core Intel Xeon server. As the de facto standard for production vision pipelines, `torchvision` constitutes a strong baseline: its operators are heavily optimized, multithreaded, and tuned for large-scale deployments.

Only a subset of packets generate output pixels and traverse the "slow" interpolation path; the rest follow a "fast" forwarding path. Consequently, the per-frame latency is computed as a weighted combination of slow- and fast-path processing times. Let N denote the total number of packets, with N_{slow} and N_{fast} for each path. The total latency is:

$$T_{\text{frame}} = N_{\text{slow}} \cdot L_{\text{slow}} + N_{\text{fast}} \cdot L_{\text{fast}} + L_{\text{fill}},$$

where L_{slow} and L_{fast} are measured per-packet latencies, and L_{fill} accounts for draining the pipeline. This reflects the streaming execution model, in which all stages operate concurrently on packetized data and sustain line-rate throughput once the pipeline is full.



(a) Classification **(b) Detection/segmentation**

Figure 8: Estimated SLO latency wins over CPU processing with `torchvision`.

Component	BRAM	DSP	FF	LUT
NetFPGA	335	0	48308	31895
JPEG decoder	10	31	3469	5791
Resize	563	0	963	3492
CenterCrop	0	0	66	321
Reshape	65	0	1594	22289
Normalize	8	0	2	15
Total	**981**	**31**	**54402**	**63803**
Available	**2940**	**3600**	**866400**	**433200**
Utilization (%)	**33.4**	**0.9**	**6.3**	**14.7**

Figure 9: FPGA resource usage for data processing operations and total utilization on NetFPGA SUME.

Applying this model, classification data processing drops from 6.8 ms on CPU to 4.4 ms on the FPGA, while detection/segmentation decreases from 11.2 ms to 6.8 ms. These improvements result from GLENFINNAN 's design principles: memoization, which replaces runtime floating-point arithmetic with compact table lookups, and pipeline specialization, which leverages fixed data processing configurations and pixel-level locality.

5.3 End-to-End Latency Budget Analysis

While CPU utilization is an important systems metric, it does not directly capture the performance perceived by users. Vision serving systems are typically governed by strict per-frame latency Service Level Objectives (SLOs), which shrink rapidly as frame rates increase. For example, applications operating at 30, 60, 90, and 120 FPS must complete all processing for a frame within approximately 33.3 ms, 16.7 ms, 11.1 ms, and 8.3 ms, respectively. Consequently, even modest reductions in preprocessing latency can translate into a substantial fraction of the available end-to-end latency budget.

To quantify this benefit, we measure the latency reduction achieved by GLENFINNAN's preprocessing pipeline and express it as a percentage of the per-frame latency budget using the per-image latencies from Section 5.2. This metric captures how much additional latency budget becomes available for the remaining stages of the serving pipeline, including batching, GPU inference, scheduling, and communication.

Figures 8a and 8b show the resulting savings for representative image classification and object detection pipelines. As expected, the benefit grows with increasing frame rate because the same absolute latency reduction represents a larger fraction of the available end-to-end budget. For image classification, GLENFINNAN operating at 190 MHz frees over 29% of the latency budget at 120 FPS, while the 100 MHz implementation frees approximately 14%. Object detection exhibits a similar trend, with the 190 MHz implementation recovering over 38% of the latency budget at 120 FPS.

As frame rates increase and latency budgets tighten, CPU-side preprocessing consumes an increasingly significant fraction of the available execution time. By performing these operations in the network data path, GLENFINNAN returns a meaningful portion of the latency budget to downstream inference and scheduling, enabling higher throughput while providing additional slack for meeting stringent end-to-end latency SLOs.

5.4 Resource Utilization

Figure 9 shows the hardware footprint. The design occupies under 1% of DSP slices and 15% of LUTs, confirming that memoization eliminates all runtime floating-point computation. Eliminating floating-point arithmetic is especially impactful because FP units, whether implemented using DSP slices or synthesized in LUT fabric, consume disproportionate resources and introduce long-latency operations that complicate timing closure. By replacing these operators with precomputed lookup tables, GLENFINNAN reduces both the spatial and temporal cost of each transformation stage. Moreover, Restructuring image-level operations into fixed-size sliding windows avoids large buffers and frame-level state, reducing BRAM pressure and control complexity. Replacing irregular control flow with simple lookup tables enables aggressive pipeline optimization and efficient logic packing.

6 Related Work

Model serving. A large body of work scales ML inference through request scheduling, batching, and resource orchestration [3, 8, 15, 27, 33, 35, 36], or through model compression and specialized deployment [4, 9, 10, 13, 22, 25, 29]. GLENFINNAN addresses a different, often overlooked bottleneck: pre-inference data processing, which dominates CPU usage in vision pipelines.

Efficient data processing. Prior work examines CPU bottlenecks in ML training [1, 2, 5, 7, 11]. FPGA-based preprocessing has been explored for recommendation models [34] and general serving [17], but these target specific workloads or do not optimize for line-rate parallelism. GLENFINNAN differs by focusing on inference pipelines with strict latency SLOs, and by aggressively optimizing packet-level image preprocessing for resource-constrained SmartNICs.

In-network acceleration for ML. Systems like Taurus, Homunculus, and N3IC [26, 28, 30–32] execute neural models on switches or NICs, while others aggregate gradients in-network [18, 19]. GLENFINNAN differs by targeting preprocessing rather than model execution, enabling high-throughput pipelines without consuming CPU or GPU resources.

7 Conclusion

GLENFINNAN shows that vision data processing can be moved from the CPU into a streaming, packet-local pipeline on commodity SmartNICs. This shift eliminates most per-frame CPU overhead, reduces data movement and latency, and fits within NIC resource

budgets. More broadly, it demonstrates that programmable NICs can serve as practical compute stages for scalable, data-intensive AI systems.

References

[1] Alexander Isenko, Ruben Mayer, Jeffrey Jedele, Hans-Arno Jacobsen . Where is my training bottleneck? hidden trade-offs in deep learning preprocessing pipelines. In *SIGMOD*, 2022.

[2] Jayashree Mohan, Amar Phanishayee, Ashish Raniwala, Vijay Chidambaram . Analyzing and mitigating data stalls in dnn training. In *VLDB*, 2021.

[3] Sohaib Ahmad, Hui Guan, Brian D. Friedman, Thomas Williams, Ramesh K. Sitaraman, and Thomas Woo. Proteus: A high-throughput inference-serving system with accuracy scaling. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, ASPLOS '24, page 318–334, New York, NY, USA, 2024. Association for Computing Machinery.

[4] Michaela Blott, Thomas B. Preußer, Nicholas J. Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, Miriam Leeser, and Kees Vissers. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Trans. Reconfigurable Technol. Syst.*, 11(3), dec 2018.

[5] Dan Graur, Damien Aymon, Dan Kluser, Tanguy Albrici, Chandramohan A Thekkath, and Ana Klimovic . Cachew: Machine learning input data processing as a service. . In *USENIX ATC*, 2022.

[6] Daniel Crankshaw, Gur-Eyal Sela, Xiangxi Mo, Corey Zumar, Ion Stoica, Joseph Gonzalez, Alexey Tumanov . InferLine: Latency-Aware Provisioning and Scaling for Prediction Serving Pipelines. In *ACM SoCC*, 2020.

[7] Derek G. Murray, Jiří Šimša, Ana Klimovic, and Ihor Indyk. tf.data: A Machine Learning Data Processing Framework. In *Proc. VLDB Endow. 14, 12*, 2021.

[8] Arpan Gujarati, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, and Jonathan Mace. Serving dnns like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 443–462. USENIX Association, November 2020.

[9] Mathew Hall and Vaughn Betz. Hpipe: Heterogeneous layer-pipelined and sparse-aware cnn inference for fpgas. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '20, page 320, New York, NY, USA, 2020. Association for Computing Machinery.

[10] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[11] Hanyu Zhao, Zhi Yang, Yu Cheng, Chao Tian, Shiru Ren, Wencong Xiao, Man Yuan, Langshi Chen, Kaibo Liu, Yang Zhang, Yong Li, and Wei Lin. Goldminer: Elastic scaling of training data pre-processing pipelines for deep learning. In *Proc. ACM Manag. Data, 1(2)*, 2023.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[14] Ionel Gog, Sukrit Kalra, Peter Schafhalter, Matthew A. Wright, Joseph E. Gonzalez, Ion Stoica. Pylot: A Modular Platform for Exploring Latency-Accuracy Tradeoffs in Autonomous Vehicle. In *IEEE ICRA*, 2021.

[15] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: Scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 253–266, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask r-cnn. In *IEEE ICCV*, 2017.

[17] ChonLam Lao, Jiaqi Gao, Ganesh Ananthanarayanan, Aditya Akella, and Minlan Yu. EdgeSight: Enabling Modeless and Cost-Efficient Inference at the Edge, 2024.

[18] ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, and Michael Swift. ATP: In-network aggregation for multi-tenant learning. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 741–761. USENIX Association, April 2021.

[19] Shuo Liu, Qiaoling Wang, Junyi Zhang, Wenfei Wu, Qinliang Lin, Yao Liu, Meng Xu, Marco Canini, Ray C. C. Cheung, and Jianfei He. In-network aggregation with transport transparency for distributed training. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, page 376–391, New York, NY, USA, 2023. Association for Computing Machinery.

[20] NVIDIA. CUDA MPS. https://docs.nvidia.com/deploy/mps/index.html.

[21] NVIDIA Corporation. Gpudirect. https://developer.nvidia.com/gpudirect, 2025. Accessed: 2025-02-11.

[22] Arthi Padmanabhan, Neil Agarwal, Anand Iyer, Ganesh Ananthanarayanan, Yuanchao Shu, Nikolaos Karianakis, Guoqing Harry Xu, and Ravi Netravali. Gemel: Model merging for Memory-Efficient, Real-Time video analytics at the edge. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 973–994, Boston, MA, April 2023. USENIX Association.

[23] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Smith. U^2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognition*, 106:107404, 2020.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[25] Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, Ion Stoica. Ekya: Continuous learning of video analytics models on edge compute servers. In *USENIX NSDI*, 2022.

[26] Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. Can the network be the ai accelerator? In *Proceedings of the 2018 Morning Workshop on In-Network Computing*, NetCompute '18, page 20–25, New York, NY, USA, 2018. Association for Computing Machinery.

[27] Haichen Shen, Lequn Chen, Yuchen Jin, Liangyu Zhao, Bingyu Kong, Matthai Philipose, Arvind Krishnamurthy, and Ravi Sundaram. Nexus: A gpu cluster engine for accelerating dnn-based video analysis. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pages 322–337, New York, NY, USA, 2019. Association for Computing Machinery.

[28] Giuseppe Siracusano, Salvator Galea, Davide Sanvito, Mohammad Malekzadeh, Gianni Antichi, Paolo Costa, Hamed Haddadi, and Roberto Bifulco. Re-architecting traffic analysis with neural network interface cards. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 513–533, Renton, WA, April 2022. USENIX Association.

[29] Sudipta Saha Shubha, Haiying Shen . Adainf: Data drift adaptive scheduling for accurate and slo-guaranteed multiple-model inference serving at edge servers. In *ACM SIGCOMM*, 2023.

[30] Tushar Swamy, Alexander Rucker, Muhammad Shahbaz, Ishan Gaur, and Kunle Olukotun. Taurus: a data plane architecture for per-packet ml. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 1099–1114, New York, NY, USA, 2022. Association for Computing Machinery.

[31] Tushar Swamy, Annus Zulfiqar, Luigi Nardi, Muhammad Shahbaz, and Kunle Olukotun. Homunculus: Auto-generating efficient data-plane ml pipelines for datacenter networks. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS 2023, page 329–342, New York, NY, USA, 2023. Association for Computing Machinery.

[32] Zhaoqi Xiong and Noa Zilberman. Do switches dream of machine learning? toward in-network classification. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, HotNets '19, page 25–33, New York, NY, USA, 2019. Association for Computing Machinery.

[33] Yitao Hu, Rajrup Ghosh, Ramesh Govindan. Scrooge: A cost-effective deep learning inference system. In *ACM SoCC*, 2021.

[34] Gustavo Alonso Yu Zhu, Wenqi Jiang. Multi-Tenant SmartNICs for In-Network Preprocessing of Recommender Systems, 2025.

[35] Yunseong Kim, Yujeong Choi, and Minsoo Rhu. Paris and elsa: An elastic scheduling algorithm for reconfigurable multi-gpu inference servers. In *DAC*, 2022.

[36] Hong Zhang, Yupeng Tang, Anurag Khandelwal, and Ion Stoica. SHEPHERD: Serving DNNs in the wild. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 787–808, Boston, MA, April 2023. USENIX Association.

[37] Noa Zilberman, Yury Audzevich, G. Adam Covington, and Andrew W. Moore. Netfpga sume: Toward 100 gbps as research commodity. *IEEE Micro*, 34(5):32–41, 2014.