"Accelerating CNN Inference on ASICs: A Survey" categorizes existing optimization strategies into three main areas: reducing computation time, minimizing memory access latency, and optimizing memory footprint. The paper also explores various architectural approaches, including dataflows and different parallelization strategies. The survey concludes with a discussion particularly in energy efficiency, performance, and scalability. I'm not sure this will help much other than to talk about why somebody might use an ASIC.

"AMD XDNA NPU in Ryzen AI Processors" discusses the architecture and software integration of AMD's XDNA NPU, the first dedicated AI acceleration engine in an x86 processor. The XDNA NPU employs a scalable dataflow architecture optimized for AI workloads, minimizing memory bandwidth usage through an explicit data movement strategy. From a software perspective, programmers can leverage AMD's ONNX-based AI execution framework, which integrates with PyTorch and TensorFlow through Vitis AI EP.

"Cerebras Architecture" The paper presents Cerebras' wafer-scale engine (WSE-2), a processor designed specifically for deep learning workloads. By distributing memory locally to cores and implementing a high-bandwidth, low-latency on-chip fabric, Cerebras enables full performance across all BLAS operations, supporting even the largest neural networks on a single chip without complex model partitioning.

"Cross-vendor programming abstraction for diverse heterogeneous platforms" proposes a software abstraction based on OpenCL. They use ONNX to enable high performance code on a wide range of hardware platforms, such as FPGAs. There's a great comparison to other works that I'll be pulling more papers from (oneAPI from Intel mentioned in 2.1, etc). There's good discussion on compilers as well.

"Energy-Efficient Computing Acceleration of Unmanned Aerial Vehicles Based on a CPU/FPGA/NPU Heterogeneous System" says "we propose a heterogeneous CPU/FPGA/NPU system aimed at realizing energy-efficient computing acceleration for computationally intensive UAV tasks." This should further illustrate why hardware choice is important, and discusses how to balance use of these hardwares for different tasks while trying to use all of them in one platform. I haven't seen a system with both FPGA and NPU, otherwise.

"HPVM: Hardware-Agnostic Programming for Heterogeneous Parallel Systems" is about a compiler framework that focuses on many types of parallelism and hardware targets – similar to some other implementations of LLVM and MLIR from what I understand. They talk about it as another IR. Of course, the only GPUs that they support are Nvidia at least from a cursory read. Sigh. They proffer a C++ dialect targeted by their front end. LLVM is utilized. I wonder what makes this much different from Modular's approach.

"HTVM: Efficient Neural Network Deployment On Heterogeneous TinyML Platforms" I'm hoping this paper will offer more insight on writing code for edge devices; most seem to focus on HPC systems. After reading more, it seems like it's a pretty specific problem they're solving and might not have implications for portable development.

"Assessing Intel OneAPI capabilities and cloud-performance for heterogeneous computing" is using Intel's DPC++ language to compare an iGPU vs FPGA implementation of some diffusion problems. It gives insight into the capabilites of the oneAPI and highlights how you still need to write code specific to each platform. The related works section might have some other good references.

“Interoperability in Deep Learning: A User Survey and Failure Analysis of ONNX Model Converters” should help me understand how to move models between frameworks and the problems that arise currently. I didn’t realize that this could fail and I’m not sure exactly how that works. They talk about a lot of open issues in libraries like onnx2torch as well as validation. Portability I thought was the whole point of ONNX, knowing its failures would be good.

“High-level GPU code: a case study examining JAX and OpenMP” directly addresses using Python along with some other (commonly systems-level) language to dive into specific hardware kernels and platform optimizations. This is what Mojo is trying to solve. This group tried the two titular libraries and summarizes their experience in development, costs, performance, etc. They seem to really want framework agnostic code.

“MLIR: Scaling Compiler Infrastructure for Domaint Specific Computation” of course I have to talk about MLIR, LLVM, and anything that Chris Lattner has touched. MLIR is part of the LLVM project and aims to allow for great abstractions in designing compilers that makes it really easy and powerful to implement domain specific tools. LLVM is fantastic, but doesn’t always work for every application. MLIR is now being implemented in many code generating stacks.

“NVIDIA Tensor Core Programmability, Performance & Precision” I need a way to complain about how Nvidia doesn’t support Tensor Core utilization in the open source computation libraries that they claim to support, locking you in to writing non-portable PTX. This paper talks more about how those Tensor Cores work and official library support.

“ONNC: A Compilation Framework Connecting ONNX to Proprietary Deep Learning Accelerators” ONNC means Open Neural Network Compiler, which is of obvious interest. LLVM is on the backend. There are scheduling internals and other utilities like Docker containers for testing available. Case studies show the viability of their work.

“O penCL: A pArALLeL prOgrAmming StAndArd fOr HeterOgeneOuS COmputing SyStemS”. The title copied over this way, wow. Anyways, it’s a foundational paper for an open source computational library that is still in use and very relevant today. There are many operations available, target devices, tools, etc. I’m hoping it will plainly explain the motivations and challenges and offer a great reference to start from.

“Performance analysis of CUDA, OpenACC and OpenMP programming models on TESLA V100 GPU” a good comparison that will hopefully show that all libraries have tradeoffs, performance considerations, and the like. I think a case study will provide great insight into how needed some of these tools are – depending on your specific concerns and needs.

“Performance and Metrics Analysis Between Python3 v/s Mojo” I struggled to find any literature on Mojo. I’ve done a bit of coding with it and it’s a big reason I wanted to do this survey project. I should probably send Modular a bill for my marketing work by the end of this survey. Mojo is meant to become a Python superset that can be compiled with a ton of other great features designed specifically with AI, heterogenous compute, and portability in mind. It exists on a large stack of other technologies. I will spend some good time on it specifically; I wish they had more formal academic releases.

“pocl: A Performance-Portable OpenCL Implementation” I picked because the pocl project was recommended to me as a more academic approach to the topic I’m surveying by a kind member of the

Modular / Mojo Discord server. It will offer more insight into writing performant, portable code. LLVM IR is utilized. I'm curious to see how this differs from the above OpenCL paper.

"QPU integration in OpenCL for heterogeneous programming" why not include some Quantum papers? That's a fun topic. I'm hoping it'll tell me more about writing on exotic hardwares as well as being forward looking on cutting edge research areas. QPUs are still very rare.

"SYCL in the edge: performance and energy evaluation for heterogeneous acceleration" SYCL I also ran into multiple times in preliminary reading. Edge computing is just as interesting to me (sometimes more) than HPC, so I'm very excited to read this. I've worked with some IoT devices that this addresses specifically. There are a lot of good comparison charts as well on Ampere GPUs (CUDA) along with Intel devices.

"The Groq Software-defined Scale-out Tensor Streaming Multiprocessor" I thought it would be good to look at the latest generation of AI ASICs. This is more of a presentation than a paper but they really have incredible information about hardware and software support, compilation, cloud operation, the kinds of models that they've optimized for (most recent AI ASICs target LLMs/ transformer operation optimizations).

"Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads" this is the paper for Groq, it seems. It has a lot more depth, case studies comparing it to other hardwares, microarchitectural decisions, cloud implementations, and the like. It should be a perfect example ASIC to show why some teams might choose to go this route. I don't think it talks about the compiler too much, however. Shame.

"Accelerated linear algebra compiler for computationally efficient numerical models: Success and potential area of improvement" this appears to be "the" XLA paper – a common Accelerated Linear Algebra library that is targeted by the likes of JAX, TF, PyTorch, and many others. It targets CPU and GPU shared memory platforms. LLVM is utilized. Most of this covers performance of operations like vector additions. It also talks about overhead costs of Python in some implementations.