Annapurna Labs Hacker Starter Pack

CalHacks 2025 / October 24-26, 2025

Amazon's AI systems, developed by the Annapurna Labs division, are backed by custom hardware and software solutions including machine learning chips and the AWS Neuron SDK. These components are designed to accelerate AI workloads in cloud environments and support development across various computing platforms, including open-source implementations.

The Challenge

Transform a basic language model into an efficient AI system using open-source implementations of popular models. Consider applying optimization techniques inspired by AWS Neuron's public documentation to improve model performance and computational efficiency. Your focus should be on measurable improvements such as reduced inference latency, optimized memory usage, enhanced throughput, or improved model accuracy while maintaining functionality.

Competition Requirements:

- Use only open source tools and documentation
- Address real-world AI optimization challenges
- Include complete documentation and setup instructions
- Host all code on GitHub with open source license (Apache 2.0 recommended)
- Make code publicly available without additional permissions required
- Demonstrate measurable performance improvements with reproducible benchmarks (see evaluation criteria below)

Getting Started

1. Review the AWS Neuron documentation

    - AWS Neuron SDK GitHub Repository
    - AWS Neuron Documentation
    - Neuron Kernel Interface (NKI) Guide

2. Explore example implementations in the samples repository

3. Choose your technical track (below) and start hacking!

Technical Tracks

1. Model Performance Optimization

This track focuses on implementing measurable improvements to language model computational efficiency through specific optimization techniques:

- **Quantization and pruning implementation**: Apply weight quantization (INT8/FP16) and structured/unstructured pruning to reduce model size and inference latency
- **Inference pipeline optimization**: Develop efficient batching strategies, memory management, and compute graph optimizations for production deployment
- **Architecture modifications**: Implement attention mechanisms, layer reduction, or knowledge distillation techniques that maintain accuracy while improving throughput
- **Resource optimization**: Optimize memory usage patterns, CPU/GPU utilization, and I/O operations for target hardware constraints

Evaluation Criteria: Projects will be measured on quantifiable metrics such as inference latency reduction (ms), memory footprint decrease (MB/GB), throughput improvement (tokens/second), and model accuracy retention (%). Participants should benchmark their optimizations against baseline implementations using standard evaluation datasets.

What We're Looking For:

- Demonstrable performance improvements with clear before/after metrics
- Creative optimization techniques that maintain model quality while improving efficiency
- Well-documented benchmarking methodology and reproducible results
- Understanding of trade-offs between different optimization approaches
- Implementation that can be applied to various model architectures

2. System Architecture & Integration

This track focuses on building production-ready deployment and monitoring systems for optimized language models using open-source tools and frameworks:

- **API integration frameworks:** Implement RESTful APIs using FastAPI, Flask, or Django; containerization with Docker; orchestration with Kubernetes or Docker Compose for scalable model serving
- **Real-time inference optimization:** Deploy models using open-source serving frameworks like TorchServe, TensorFlow Serving, or ONNX Runtime; implement load balancing and caching strategies
- **Performance monitoring and analytics:** Build monitoring dashboards using Prometheus + Grafana; implement logging with ELK stack (Elasticsearch, Logstash, Kibana); track inference latency, throughput, and resource utilization
- **Cross-platform deployment solutions:** Create deployment pipelines using GitHub Actions or GitLab CI/CD; support multiple environments (local, cloud, edge) using containerization and infrastructure-as-code tools like Terraform

Evaluation Criteria: Projects will be measured on deployment reliability, API response times (ms), system scalability (requests/second), monitoring completeness, and successful multi-environment deployment. Participants should demonstrate working deployments with comprehensive monitoring and documentation.

What We're Looking For:

- Robust, production-ready systems that can handle real-world traffic patterns
- Comprehensive monitoring and alerting capabilities with meaningful metrics
- Scalable architecture that demonstrates understanding of distributed systems principles
- Clean, maintainable code with proper documentation and testing
- Evidence of performance under load with stress testing results

3. **Hardware-Aware AI Optimization & Deployment**

This track focuses on implementing hardware-aware optimization and deployment strategies for AI models using open-source tools and frameworks:

- **Hardware-aware model optimization**: Implement quantization schemes optimized for specific chip architectures (INT8, FP16, BF16); develop pruning strategies that consider hardware memory hierarchies; create model compression techniques that leverage custom silicon capabilities using PyTorch, TensorFlow, and ONNX optimization tools
- **System-level performance monitoring**: Build comprehensive monitoring systems for hardware utilization, memory bandwidth, compute efficiency, and thermal management using Prometheus, Grafana, and custom hardware profiling tools; implement real-time performance tracking for inference workloads
- **Infrastructure deployment optimization**: Develop containerized deployment strategies using Docker and Kubernetes that optimize for hardware-specific configurations; implement auto-scaling solutions that consider hardware constraints and capabilities; create deployment pipelines optimized for custom silicon environments
- **Hardware-software co-design simulation**: Build simulation frameworks to model the interaction between AI workloads and custom hardware architectures; implement performance prediction models for different chip configurations; create optimization algorithms that balance software efficiency with hardware capabilities

Evaluation Criteria: Projects will be measured on hardware utilization efficiency (%), inference throughput improvements (ops/second), memory bandwidth optimization, energy efficiency gains, and successful deployment across different hardware configurations. Participants should demonstrate measurable performance improvements when optimizing for specific hardware architectures.

What We're Looking For:

- Deep understanding of hardware-software interactions and optimization opportunities
- Innovative approaches to modeling and predicting performance across different hardware architectures
- Practical solutions that demonstrate measurable improvements in hardware efficiency
- Simulation or modeling frameworks that can guide real-world optimization decisions

- Evidence of thinking beyond pure software optimization to consider the full system stack

Have an idea that doesn't fit these categories? We welcome novel solutions to AI optimization challenges. If you've identified a technical opportunity in AI development that addresses real-world performance, deployment, or hardware-software integration problems, we want to see it!