**Hackathon Problem Statement: AI-Optimized Neural Accelerator Design for Edge Inference**

**Problem Overview**

In 2025, the explosion of edge AI deployments—driven by IoT, autonomous drones, and smart manufacturing—has created a bottleneck in hardware efficiency. Neural network accelerators (e.g., TPUs or NPUs) struggle with the computational demands of large models on resource-constrained devices, leading to high latency, excessive power draw, and scalability issues. Traditional design tools rely on manual iteration, taking weeks to optimize for specific workloads like real-time inference in video analytics or sensor fusion. Companies like NVIDIA, Qualcomm, and Intel are pushing for automated, AI-driven design flows to accelerate chip innovation, but current solutions lack the speed and adaptability for rapid prototyping in dynamic environments.

The challenge: Create an AI tool that automates the optimization of neural accelerator architectures for edge inference tasks, generating efficient hardware configurations (e.g., systolic array sizes, memory hierarchies) tailored to a given ML model and constraints like power budget or latency targets. The tool should simulate performance metrics and suggest design trade-offs, enabling faster iteration for hardware engineers.

**Why This Problem is Significant in 2025**

* **Scale of Impact**: Edge AI market is projected to hit $100B by 2028 (per Gartner 2025 reports), but 70% of deployments fail due to inefficiency (IDC analysis). Automating accelerator design could cut development time by 50%, boosting innovation in semiconductors amid chip shortages.
* **Current Gaps**: Tools like Synopsys or Cadence are expensive and manual; open-source alternatives (e.g., TVM) don't leverage cutting-edge ML for auto-optimization. This aligns with tech giants' focuses—NVIDIA's CUDA-X for AI hardware, Qualcomm's Snapdragon NPU challenges in edge hackathons, and Intel's oneAPI for open accelerator innovation.
* **Hackathon Fit**: Mirrors real challenges from 2025 events like Qualcomm's Edge AI Developer Hackathon (optimizing models on Snapdragon X), NVIDIA's GTC Hackathon (hands-on with newest AI tools for hardware), and Intel's oneAPI Hackathon (curated problems on ML acceleration). It's a "tech-first" pursuit, emphasizing research in AI-for-hardware over end-user apps.

**Proposed Solution Idea: "AccelForge" – AI-Driven Accelerator Optimizer**

Develop **AccelForge**, a web-based simulator that:

* **Inputs Model Specs**: User uploads a simple ML model (e.g., ONNX format for object detection) and constraints (e.g., <1W power, <50ms latency).
* **Generates Designs**: Uses AI to search and propose optimized accelerator configs, outputting visualizations like array layouts or performance graphs.
* **Simulates & Compares**: Runs quick cycle-accurate simulations to benchmark designs against baselines.

**Key Features for MVP**:

* Upload interface for model/constraints.
* Output dashboard with design proposals, sim results, and trade-off plots.
* Exportable config files (e.g., JSON for Verilog generation stub).

**Technologies from the List to Leverage**

Use **7. Advanced Neural Architecture Search (NAS)** + **6. Advanced Mixture of Experts (MoE) and Sparse Models** for a synergistic, hardware-aware approach:

* **NAS**: Automates discovery of optimal accelerator architectures (e.g., varying compute units, interconnects) using meta-learning for quick convergence on edge constraints. This handles the "search space explosion" in design optimization.
* **MoE**: Integrates sparse experts for domain-specific routing (e.g., one expert for memory optimization, another for compute scaling), keeping inference lightweight—perfect for simulating on a laptop without full HDL tools.

**Why This Combo?**

* Synergy: NAS explores vast design spaces efficiently, while MoE prunes to feasible configs, mimicking real chip co-design. Both are research frontiers, aligning with NVIDIA/Qualcomm's GPU/NPU innovation.
* Feasibility: Leverage open-source like AutoKeras (for NAS) or Hugging Face's sparse MoE implementations; simulate hardware via PyTorch's proxy models (no real FPGA needed).

**MVP Implementation Roadmap (12-16 Working Hours)**

Target: A Streamlit web app for demo on laptop—use synthetic/simulated data (e.g., pre-defined ML workloads like MobileNet). Assume 2-4 person team; prioritize core search/sim over full accuracy.

| **Phase** | **Tasks** | **Estimated Time** | **Tools/Libs** |
| --- | --- | --- | --- |
| **Setup & Data Prep (Hours 1-2)** | - Set up env (Python 3.10+). - Load sample models (e.g., ONNX from Torch Hub). - Define constraints/simulator: Simple cycle model (e.g., MACs \* latency calc). | 2 hours | Hugging Face (for MoE), ONNX Runtime, Streamlit |
| **Core NAS + MoE Integration (Hours 3-7)** | - Implement NAS loop: Use meta-learning to sample 10-20 designs (e.g., vary array size 4x4 to 16x16). - Add MoE: Route to 3-4 "experts" (sub-models) for scoring (e.g., power via FLOPs proxy). - Output top-3 configs with metrics. | 5 hours | AutoGluon/AutoKeras for NAS, PyTorch for MoE (pre-trained sparse layers) |
| **UI & Simulation (Hours 8-11)** | - Build Streamlit: File upload, constraint sliders, generate button. - Sim engine: Run quick evals (e.g., 100 cycles) and plot (Matplotlib: latency vs. power Pareto). - Add visuals: Simple ASCII/array diagrams or Plotly heatmaps. | 4 hours | Streamlit, Matplotlib/Plotly |
| **Testing & Polish (Hours 12-14)** | - Test 3-5 scenarios (e.g., CNN vs. RNN workload). - Edge cases: Invalid constraints → graceful error. - Metrics log: "Search time <30s". | 3 hours | Manual runs, Jupyter for tweaks |
| **Presentation Prep (Hours 15-16)** | - Demo script: 2-min walkthrough (upload → optimize → sim). - Slides: Problem, tech (NAS+MoE), benchmarks (e.g., "30% better efficiency vs. baseline"). | 2 hours | Google Slides, screen record |

**Total Effort**: 16 hours max. If short on time, fix NAS to 5 iterations and use a single MoE expert.

**Success Metrics for Judging**:

* **Technical**: Valid designs (e.g., 20%+ efficiency gain on sims), fast search (<1min).
* **Impact**: Ties to industry (e.g., "Reduces Qualcomm NPU design cycles"), extensibility (e.g., API for real tools).
* **Innovation**: Toggle "sparse mode" to demo MoE routing; highlight research tie-ins like quantum-NAS hints.

This draws directly from 2025 hackathons (e.g., Qualcomm's GenAI Chip Hackathon for design automation, Intel's oneAPI for accelerators), delivering a crisp, tech-deep MVP that showcases bleeding-edge ML for hardware R&D. For team strengths in VLSI, it could evolve to full RTL export!