Approximate Computing for Efficient Generative AI Inference

Processing Generative AI (GenAI) models, including large language models and diffusion models, is expensive. Every token or image they produce requires moving billions of parameters through memory and executing enormous numbers of multiply-accumulate operations, which costs energy and execution time. However, these models do not need perfectly exact arithmetic, because their outputs tolerate small numerical errors. Quantization already exploits this tolerance by lowering numerical precision, and it is now a standard practice. *Approximate Computing* goes one step further. It redesigns the arithmetic circuits themselves, using multipliers and accumulators built from fewer logic gates than their exact counterparts, and can therefore save area, power, and latency beyond what quantization alone can achieve.

Not every part of a computation tolerates the same amount of error, and not every arithmetic operation has the same hardware cost. The best place to approximate therefore shifts across the intermediate steps of an inference, across the parts of a model, and across the arithmetic itself. This is where the research lies. A diffusion model runs the same network tens of times per image, and each denoising step tolerates a different amount of error. A mixture-of-experts language model contains many experts whose activation frequency and sensitivity differ by orders of magnitude. New low-bit floating-point formats such as MXFP and NVFP4 make multipliers nearly free and shift the hardware cost to accumulation. In all these cases, the central question is the same. Where, when, and by how much arithmetic can be approximated while the generated text or images stay close to the exact baseline?

Key Tasks: This project explores one concrete direction within this theme, selected according to the student's interests and background. The following is a tentative task list,

- Study the state-of-the-art quantization and approximation methods
- Build or adapt a PyTorch-based simulator that injects approximate arithmetic into a real open model
- Analyze which parts of the model tolerate errors and estimate the hardware savings of promising designs using analytical models or standard chip design tools
- **For M-EMSYS and M-EE:** Implement a representative part of the architecture to analyze hardware efficiency, e.g., energy efficiency. **For CS students:** the project runs entirely in software using Python, PyTorch, and open models. The student will gain experience at the intersection of machine learning systems and computer architecture, a profile in high demand in the embedded AI field.

Eligibility Criterion: Interest in deep learning models such as transformers or diffusion models; Python, PyTorch, computer architecture, and efficient AI processing.

Supervisor's Contact: Dr. ir. Ghayoor Gillani (s.ghayoor.gillani@utwente.nl), Zilverling 5078.