WASP SE Module assignment 2025

Author: Arseni Ivanov

1 Introduction

My research area is in Computer Engineering, more specifically GPGPU(General-purpose computing on Graphics Processing Units) programming within a Computer Graphics group. The area includes writing code and algorithms that can run fast on Graphics Processing Units regardless of what manufacturer has created the card. My current research combines my grant (GPGPU), my supervisors expertise (Computer Graphics), and my own interests (Information Theory, Compression, Meta-learning, Systems Engineering) to construct a system that is able to compress a group of textures in a way where they can learn to share the compressed representation with each other. This is done by building on the single-texture NTC(Neural Texture Compression[11]) framework from NVIDIA but looking at it from a systems-level compression.

The architecture of NTC can be lazily summarized as a feature map, which learns a spatially-dependent downscaled version of a texture across N channels, and an MLP(Multilayer Perceptron) that uses the feature maps from N channels to reconstruct the target texture.

The purpose of NTC is to yield a real-time, random access lookup, neural texture compression. Which means that we should be able to get the value of any texture at any coordinate whenever we want without having to decompress the whole texture, in real-time (40 ms/frame). Neural texture compression as a term means that neural networks are used to approximate information contents in textures. The research idea builds on that the current NTC solution is non-general, every texture is treated the same, which does not make sense from an information-theoretical perspective. A high-frequency detailed texture, such as hair or fur, will need a different amount of information to represent itself compared to a single-colored box. This means that in the NTC framework, the low-frequency option could use significantly less feature map channels. Also, digital artists are sometimes lazy, and copy/color shift their textures. This adds minimal information to the **texture set as a whole**, but if treated separately, will increase the binary size linearly.

The current implementation is also completely written in CUDA[7], which limits the framework to be used on NVIDIA GPUs. A goal of the paper is to facilitate GPGPU training and run-time inference of the new system, for example by utilizing open-source solutions like Triton[10] for training and Vulkan[9] for inference.

2 Lecture Principles

2.1 Requirements for SE4ML

During the requirements part of the lecture, I thought a bit about the constraints and requirements for ML systems that are not fully covered by the traditional SE perspective. One such requirement is the hardware constraint. There is a very valid claim that the transformer architecture[12], was not a massive breakthrough in architecture, since its essentially an unrolled RNN with growing context instead of folding representation. This allowed systems using this architecture to be parallelized across the currently available compute medium (GPUs), both inter-and-intra-GPU, leading to the state of the field/world that we are in now.

This is quite relevant for my research, and should be for much other research in my biased view. The concept of compute is needed for any intelligent system, and there is a reality of the current existing compute. If you want to design an intelligent ML-system, you need to use the compute that you have available. A majority of today's ML workloads are memory-bound. This means that the GPU is waiting for data to be transferred to it instead of performing computation. Some of the fastest adapted research by the industry as of date, such as FlashAttention[5] and its successors, has built on reducing the memory-boundness of the computations.

In my research, this is expressed by looking at how we pass data to and from the GPU. Can data be loaded in chunks small enough for the gradients to stay on the device and be used for back propagation? Can parts of the computation be fused to avoid having to write back to slow GPU memory? Can a requirement on good enough GPU utilization drive a better design of the full solution? If we go to the extreme, requirements on low-bit representations of the architecture lower the amount of state changes that the transistors have to make(fewer possible values = higher chance of a transistor having the same state in two different computations), reducing the heat generated by the GPU, which increases throughput as less throttling needs to be done, this is mentioned in this blog[2].

I believe such questions will be more relevant as the field matures and information & computation as raw concepts take over the role of research focus from the network architectures. This is my suggestion for an example on how requirements in the field of SE4ML or AI Engineering will be different from SE.

2.2 Behavioral SE in academic communication

This might be a bit of a wide take on Behavioral SE, but I am going to try to fit it in here because its relevant for me and a fun concept.

Today, there are incredible amounts of information being created. You can have an LLM just generate information endlessly. In such a world, people get overwhelmed, and knowledge/information becomes bound by what you choose to engage with rather than what is available. This spills over into the academic realm as reviewers have to go through more and more work, and forces researchers to emphasize writing papers that will stand out in the sea of the 4000 other papers accepted to NeurIPS last year. A blog post has been written to guide general PhD students on how to get their papers accepted [4] by placing a lot of attention and work on the first page.

In the realm of Computer Graphics journals and conferences, this is taken to the extreme, as the reviewers are used to crisp-looking, high-quality images. Some papers are having to provide "low-res" versions of their work, as their original PDFs exceed 100 MB sizes.

We can see from the blog and from the evolution of writing styles of papers that the current strategy of successful communication builds on things like placing key information early on, high-lighting and bold text or optimizing figure design for a human eye-pleasing layout.

Perhaps this is something that will start permeating into SE standards, as peoples attention spans shrink and action needs to be taken to use this more limited resource while successfully communicating your intent with your programs/papers.

3 Guest-Lecture Principles

3.1 The distinction between Problem-space and Solution-space

This is a concept that is highly relevant for my work. The need for a random-access neural compressor is built on the fact that file sizes for video games are growing very large. A modern video game can take 350 GB of memory, out of which half are textures. This is a massive cost of both information transfer as you distribute the game, but also as memory on your local device is limited. Since most hardware is memory-bound and not compute-bound, it makes sense to offload the transfer to compute. In general the research-project is a very design science/engineering based, which makes it dependent on a observed problem.

3.2 Decomposing system goals into quality requirements

This research project in itself is a decomposition of various stakeholders(me, my supervisor, my grant givers, the industry) and our knowledge bases to requirements that yield a paper that will be meaningful for all the stake-holders.

4 Data Scientist versus Software Engineers

4.1 Are the differences essential?

I do agree on the differences, I have worked in R&D for a product for 3 years before my PhD, and it was clear that there was a lack of SE expertise amongst the product owners that the ML Engineers/Data Scientists had to pick up. It was also then obvious that a lot of the DS people had not been trained for SE tasks, which they viewed as boring, time-consuming and difficult without a clear purpose.

I think that in general for the cheapest improvement on both fronts, SE people would benefit from having insight into systems-wide DS concepts like data drift, concept drift or other ML/DS terms that are related to the system/process and not to the pure statistics or model architecture. The DS/ML people on the other hand should get an insight into requirements engineering, testing, CI/CD and UI design.

4.2 Specialization vs Co-evolution

I think this is a societal question more than an individual question. My personal opinion is different from what I see happening currently, but I will try to argue for it. Its a societal question because there is currently people who are not interested in doing 5 years of university to work at an office job. There are also people who love to focus on a single subject and become the complete domain experts in it. To cater to those people, there are shorter educational programs that specialize, as well as specialized positions at companies. However, I would argue that this will not be valuable in the future I see.

My view is that in the age of LLMs, or more generally the combined knowledge of humankind one chatbox away, a broad understanding of the world will be much more valuable than specialization. If I know the computer science, economic, or legal terms that I need to use to achieve my ends, I am able to prompt the vast LLM embedding space with more precision and understand/learn something quickly. The "unicorns" mentioned in chapter 1 of the book will not be mythical, but will be the expectation of the average engineer in a few years I believe.

5 Paper Analysis

5.1 Towards Understanding Model Quantization for Reliable Deep Neural Network Deployment

The paper[6] makes an evaluation of neural network model compression techniques such as quantization. The authors look into 4 datasets and 8 model architectures as well as 42 datasets that have been synthetically shifted in their distribution. Their key findings are:

- Distribution shifts during inference cause more disagreements. By slightly shifting or noising the data, the quantized models perform significantly worse.

- QAT(Quantization aware training) leads to more stable models. By teaching the model to handle uncertainty, the models perform better.
- Margin is a better metric for disagreement than other uncertainty measures for this problem such as entropy or gini index. Essentially, the authors argue for that the quantization error is a linear offset, which makes a linear metric more suitable to measure the uncertainty compared to a distribution uncertainty measurement.
- Retraining the model with the disagreements does not improve the model. By shifting the distribution to account for the disagreements, we introduce new disagreements.

The paper relates quite directly to my research as I am also performing low-bit quantization, the feature maps in the original paper are 4 and 2 bit-quantized. I am also performing QAT already by adding noise to the quantized values that is correlated to the bin size, I am during the training teaching the network that the value it reads from the feature map will have noise proportional to the quantization error after quantization, this significantly reduces the metric drop when parts of the network are quantized and frozen during the training.

I think that the papers ideas could fit into any kind of project at the systems-level design if it involves a quantization process. QAT is a common standard design for projects that are anticipated to be quantized at the start of the system design. The robustness of models is often a base requirement, bullet points 1 and 4 in the findings are directly competing, you want a robust, stable system but at the same time you do not want to overtrain towards outliers. This means that quite early on, it needs to be established what kind of problem that is being solved, what the requirements are, and what are real and not real data-points. Is it necessary to chase a specific use-case when it will reduce the system performance on the average case? Or does the case warrant its own sub-system?

I think that what could be adapted from the paper into my research is the consideration of the metric of error to optimize against. The Margin metric example shows that the evaluation target calls for a simpler, linear metric. Even if we were to use top-2 entropy instead of the top-2 margin to evaluate the uncertainty, we would skew the results with the non-linear log transform leading to disproportional uncertainty evaluation compared to the linearly distributed measurement target of the uncertainty. This is a good thing to think about, my project for example has two stages, so perhaps the compounded accumulated errors(feature map sampling + MLP) is non-linear after all, the errors are linear after the first stage(quantization), but perhaps the second stage(MLP) can account and adjust for the errors introduced in the first stage, and make them non-linear, leading to the need of a non-linear evaluation target metric.

It also becomes crucial to look at what we are evaluating. The paper is looking at the **robustness of the final output state**, so it is ignoring that the quantizations are non-linearly transformed inside of the network. In my work, the final output is a color pixel value that is later on aggregated with various other estimated channels(shininess, bumpiness, light emittance) into a material that is rendered on screen, the output might look fine in if just looking at errors, but when being placed in a scene with various lights, shadows and interactions, it might be more sensitive to those errors, what is a good metric here? The current answer is human qualitative evaluation with vibes and colleagues, but the paper suggests that there might be better ways, so an AI-engineering approach would be to think about the whole system and decide if its possible to look at a more quantiative evaluation of it.

5.2 How Do Model Export Formats Impact the Development of ML-Enabled Systems? A Case Study on Model Integration

The next paper[8] looks at different neural network model export formats and compares them across different criteria such as: Programming language integration/tech stacks(Python, JavaScript, TypeScript), model architecture/use-cases(referred to as systems in the paper), and export formats(ONNX, joblib, pickle, TF, PyTorch).

The result of the study was that ONNX was the most suitable option for most use-cases as it provided the most integration and portability, but that the Python specific-flavors like TF's SavedModel and PyTorch's TorchScript provided some Python-specific benefits.

The paper relates to my research as I am in the same way exporting my models between formats. My models are trained in Python, but the inference needs to run in a graphics library like Vulkan, hence the models need to be loadable in C++ and referable in GLSL C-like shader code. There is currently no general tool that does this, since the Vulkan and graphics library extensions rely on user-allocated specific layouts, buffers, and orchestration of jobs to the GPU. This leads to a current example process being: Manual export of tensors from Python using safetensor library(similar to joblib in the paper), loading the tensors in C++ using the safetensors header file, manually defining all the layouts and buffers for the necessary structures of the tensors in C++, and finally the implementation of all the neural network layer computation code in the GLSL language.

This question is actually a HUGE pain-point for graphics researchers, it makes any integration or research take much longer that needed and require a very wide toolkit for the average researcher to get anything done. There has been arguments since 16 years on this(OpenCL standard creation), and nothing has been done in practice. There is a fantastic breakdown and a sad read about this in this Reddit post[1].

Some of the bigger points are:

- If you want general software for specific hardware, you are going to have to find the least common denominator between all hardware, inevitably losing the hardware-specific features.
- Any standard work between the stakeholders will be **coopetition** as described by this blog[3]. Companies have their own interests and agendas in mind for such projects.
- Any standard work between the stakeholders will be slow. Glacier level slow. A company like Apple cant sit around and wait for 6 months until Intel or Qualcomm to agree to some point in the standard.

OpenCL is a prime example of this – slow committee and many big actors with their own interests that essentially gives way for the vendor-specific alternatives.

The integration of the ideas that are provided by ONNX from the paper into the Graphics API's, or even half-way into a general language for GPU's, would change the world, quite literally. It would democratize compute, and allow for compute consumers to not have to rely on closed-source monopoly software from specific actors. You would be able to write general compute workload code once and run it on any hardware.

My WASP research is based on this. The goal of the project is to make workloads run fast and reliably across different hardware vendors.

My current project is already integrating some of these thoughts by trying to make the workloads as general as possible on their specific stages(training, inference). The current fastest cross-hardware support for training(NVIDIA/AMD) in Python is offered by Triton, there are options suboptimal options like Apache TVM, and new not yet mature options like Mojolang are built for this purpose at the moment. The only mature option for cross-hardware graphics inference is Vulkan(NVIDIA/AMD/Apple/Intel). I think to further increase the generalizability, integration and application of AI Engineering techniques, it would be necessary to either try to integrate a non-mature training option that supports more backends, or to somehow combine the training-evaluation workflow into a single format like ONNX that is able to capture the compute graph, or use a SavedModel-like format that is able to capture preprocessing in a general enough way to support the translation between Python and C++.

6 Research Ethics and Synthesis Reflection

6.1 Search and Screening

The search and screening was done by providing all the titles from all years of CAIN into an LLM together with a description of my project as well as the task description. The top 10 most relevant results were selected, and their abstracts were placed into the same LLM conversation. The top 5 most relevant results were selected and the abstracts were read manually in order to find 2 good matches for analysis. Finally the 2 paper were read manually in their entirety.

6.2 Pitfalls and mitigations

It can be that the objective in the prompt is not clear enough for the LLM to make a good initial choice of articles. This is a risk when trading off a complete manual search-and-retrieve for a probabilistic estimate, and can be mitigated by doing this work manually.

6.3 Ethical considerations

All of the synthesis and text was written by me by hand, while the search and retrieval was done by LLMs, this is a good trade-off, as no text of the analysis or report was generated according to the rules while minimizing the time it would take to read through irrelevant articles. It is also sometimes possible that the title or abstract does not reflect the contents for a biased reader such as myself, which makes it useful to have an LLM to point out that this in fact can be of high relevance to me.

References

[1] Reddit comment on the state of gpgpu, 2023. URL https://www.reddit.com/r/vulkan/comments/11lklcx/comment/jbfm7jp/.

[2] Stange matrix multiplication, 2024. URL https://www.thonking.ai/p/strangely-matrix-multiplications.

[3] What about opencl and cuda c++ alternatives?, 2025. URL https://www.modular.com/blog/democratizing-ai-compute-part-5-what-about-cuda-c-alternatives.

[4] The phd metagame, 2025. URL https://maxwellforbes.com/posts/how-to-get-a-paper-accepted/.

[5] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL https://arxiv.org/abs/2205.14135.

[6] Qiang Hu, Yuejun Guo, Maxime Cordy, Xiaofei Xie, Wei Ma, Mike Papadakis, and Yves Le Traon. Towards understanding model quantization for reliable deep neural network deployment. In *2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 56–67, 2023. doi: 10.1109/CAIN58948.2023.00015.

[7] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. *ACM Queue*, 6(2):40–53, March 2008. doi: 10.1145/1365490.1365500. URL https://doi.org/10.1145/1365490.1365500.

[8] Shreyas Kumar Parida, Ilias Gerostathopoulos, and Justus Bogner. How do model export formats impact the development of ml-enabled systems? a case study on model integration, 2025. URL https://arxiv.org/abs/2502.00429.

[9] The Khronos Group. Vulkan 1.0 API Specification. https://www.khronos.org/news/press/khronos-releases-vulkan-1-0-specification, February 2016. Initial public release.

[10] Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

[11] Karthik Vaidyanathan, Marco Salvi, Bartlomiej Wronski, Tomas Akenine-Moller, Pontus Ebelin, and Aaron Lefohn. Random-access neural compression of material textures. *ACM Trans. Graph.*, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592407. URL https://doi.org/10.1145/3592407.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.