1. Introduction

I am an industrial PhD student at Ericsson and Lund University. The research I conduct revolve around increasing efficiency with a core focus on Neuromorphic computing. Currently, methods with how to discriminate between models with respect to input are explored. It is our goal to continue this exploration in pursuit of a method that can effectively incorporate efficient neuromorphic computing.

Neuromorphic computing is an umbrella term for the full stack of computing, from hardware to software that attempts to increase efficiency by mimicking biological neurons. In terms of hardware there are for example memristors, a novel analogue compenent that enables more scalable analogue computing. However, my research is explicitly within the software domain. That is, how to model neurons for neural networks and the methods with which to train such novel networks. One of the leading neural models are Spiking Neural Networks (SNN). In contrast to the perceptron of classical neural networks it also models the spiking behaviour observed in biological neurons. It has already been observed that such SNN enable orders of magnitude lower energy consumption when deployed to co-designed analogue hardware.

The stated reasons above so far only apply to inference time, the reasons that wide adoption have not yet ocurred revolve around a few core issues. The first issue is gradient descent. Perceptrons have seen the extrem sucess that can be observed today because of gradient descent. It is thus the first tool ML engineers and researchers turn to when adopting a new type of neural network. However, gradient descent cannot be immediately adopted into SNN due to exploding gradient of the spike function. Secondly, while the classical neural networks typically don't contend with temporal data, it only has a single set of input features to compute. As SNN are inherently temporal, a set of input features that aren't temporal have to be extended and so the input dimension increases. This makes SNN notoriously slow to train, increasing training costs. The second issue is that no method so far have been able to consistenly show better or on par performance of perceptrons. Meaning that it is known beforehand that deployed SNN will have worse performance than a State of the Art perceptron model. Because of the order of magnitudes less energy consumption, there are many applications where this tradeoff is desirable.

The methods with which to solve gradient descent or training in general for SNN are still up for debate. My project however attempts to gain the benefits of SNN but at the performance of perceptrons. To do this, the projects explores hybridization and discrimination methods for choosing between SNN and perceptron models.

2. Lecture Principles

Grey Box - Semantic Flow

As stated a multitude of times throughout the course, AI/ML deployments intertwine and compound the issues already associated with Software Engineering. The enormous issue of understanding how a model makes decision is an intertwined plague of AI/ML deployment. Changes in software can affect the model and vice versa, even more so for non-deterministic models. The idea of representing the latent space as the control flow is enlightening. Something that could, and I assume the paper brings up, alleviate the issues of model updating. As the model is changed over time, through ad-hoc training, or with other methods, this visualization can help developers understand how the model is

actively changing.

In my own research, I'm exploring model discrimination. In other words how to best choose between models. This visualization method could be used for myself to understand the difficulties models have in seperating clusters. In other words it could provide a tool for myself to develop better models for an already difficult task.

Group Dynamics

Focusing on group dynamics to improve the work efficiency. It's an interesting topic, I agree that without it a group could devolve into an unproductive blob. At the same time I'm of the mind that many organizations that care to take care of such things can often put too much emphasize on it. How does one know if a group is even able to reach what would be considered optimal efficiency or working conditions? Are there groups who are fundamentally incompatible? It is most likely that a majority of humanity is able to at least to some level cooperate. However, I'm also of the mind that during recruitment or changes in group dynamics it's more fundamental to ensure the new individuals are to some extent already compatible with the group culture.

I have met resistance within my own research group due to culture or fear of group dynamics. It is my experience that PhD students more often work in solitude within Sweden. This mode of behaviour decreases the dependency on each other, potential conflicts, but is also prohibitive in cooperation. I myself have tried to increase cooperation through joint projects, only to be met with very cautious optimism. The caution is of course not unfounded, as any potential conflicts could escalate when a group is forced to work together either way.

3. Guest Lecture Principles

Technology adoption

To consider the varying behaviours of people in groups and how it plays out in the work environment is again important. An aspect which I think many might sooner or later in their life come to experience, the reluctance or love to new technology. It of course plays a role in how quickly a team can adopt new ways, integrate it into the workplace and more. I believe there are sound reasons behind the diversity of less and more curiosity among a group as it increases stability over time. Not all ideas are sound, neither are ways of working nor technologies. If some are reluctant and the old methods are superior it is more likely it will survive if some are reluctant to it, and vice versa.

Despite the perceived novelty of the field of ML/AI, the field can be extremely rigid at times. It is common within our group that our research is often met with criticism with respect to underwhelming performance, despite the majority of our research not aiming to achieve increased performance. For a field that is in need of larger exploration, the single-mindedness of increasing an arbitrary set of metrics for a set of datasets is revealing. It is, like many other fields, also a slave under perceived standards, and met with fierce resistance when a different goal is set.

Cost of removing defects

As someone without great experience in software deployment, it was eye opening the exponential cost of removing defects. In hindsight it becomes apparent, I reminiscence the issue that the MIPS architecture have, which still have to consider empty instructions in compilation, despite the hardware not having the same issue anymore. I also become reflective of my own software pipeline. While I do have a process in which I actually

deploy software, the defects I leave over time become increasingly hard to rid oneself of. It is not uncommon to face the common hole of not pulling branches into main, but rather moving main into a branch. Something I believe was raised as well in the lectures.

4. Data Scientists versus Software Engineers.

I concur with the simplifications made to describe the differences between Data Scientists and Software engineers as a generalization. As software tools have become more accomodating to Data Scientists, the need for a background in software development have diminished. This is not necessarily a worse thing as it have enabled a deeper specialization in the field. We find ourselves in a time where Data Scientists are becoming ML engineers, it is thus not odd to find the Software Engineer more preoccupied with customer deliveries as the field is generally well understood. The Data Scientist is just now coming out of the field of research, so I do believe some of the qualities found among other engineering fields will soon find itself in ML engineers. Moreover, I don't find it odd that there will be a continuation in specialization. For example Software engineers need profound knowledge of a multitude of systems, hardware being a continously evolving field among them. On the other hand so do Data Scientists currently, the field is heavily researched and rapidly evolving, with a great many tools available to understand and process data. However, as tools progress, with for example LLMs, I wouldn't be surprised that we can find more and more "prodigies" in both fields. I think overall that both fields will continue to specialize further. Software Engineering being the glue that holds everything together won't necessarily rely on ML engineering, but ML engineering will have to understand Software Engineering to communicate efficiently with their Software Engineering peers. In other words, I think there will be more ML engineers that know a little more about software Engineering than vice versa due to the dependency graph.

5. Paper Analysis

5.1 Investigating Issues that Lead to Code Technical Debt in Machine Learning Systems

The authors identify the established idea of technical debt in ML engineering, like that of SW engineering. They reference previous work that there is technical debt uniquely associated with ML engineering. Thus they attempt to map the parts of the ML engineering workflow that cause it. Inviting personel working actively with ML engineering, they hosted multiple session with predefined areas of discussion surrounding ML technical debt. If pervasive sources across industry can be identified, solutions to such issues could be found to alleviate or raise awareness of such problems for most ML projects going forward. The authors broke down the ML engineering workflow into 7 core steps, and 30+ candidate issues to discuss in each session. In summary, the core issues related to ML engineering surround data pre-processing, with data collection, and model training as secondary culprits. For anyone working with ML, it is not surprising that the conclusion reasserts what is already the consensus, that pre-processing is the most sensitive part of ML engineering and research. Of course it might be confirmation bias.

Two common themes in the solutions proposed by the authors appear, scalability and automation. They propose to take scalability into account from the start of a project to avoid a majority of technical debt. Another is to implement automation where appropriate to avoid technical overhead. This ties into scalability, as automation allows for greater scalability. However, I believe the authors might have missed a critical part of why technical debt is accrued in these areas. If an absolute majority of ML projects never even

make it to production, are engineers not incentivized to cast a large net of ML projects. That is, in order to make it to production many eggs need to be spoiled. In that case, focusing on scalability, when it cost time, time that could be spent on other parallel ML projects is ill afforded by those ML engineers. Is it not the case then that it is incredibly hard to avoid such technical debt? Could it not be argued that it might be unavoidable technical debt?

As an industrial PhD it is an obviously related issue in my research. While not explicitly stated, it is expected that my research will have a buisness impact and be applicable to core buisness activities. Activities related to ML deployment are already planned in my research project. It is therefor pertinent that I, for my own sake and also for potential future stakeholders, take appropriate action to avoid unecessary technical debt. As my activities relate to research in how to implement ML on the fog, that is collaborative inference between edge devices and cloud, it can be argued that it will be twice the code than normal ML projects. Since both edge and cloud have different implementation, technical debt could accrue faster than a typical ML project. Morever, while technical debt is mainly relevant for industry, the concept can also be extended to research. I notice that I too accrue technical debt in projects where I am even alone. To identify implementable and solid solutions to such issue can thus not only help industry but the ML field as a whole.

A potential project could be routing data efficiently across a network of an uncountable number of interconnected nodes. It is an incredibly hard problem to compute and takes a lot of resources. Moreover, routing needs to efficiently take congestion into consideration, and individual nodes need to account for malicious actors. Data driven routing could alleviate the issues observed in networks today, both on individual nodes but also on the full network. Because of the scalability of such a project, the implementation of automation of pre-processing, and data collection will significantly impact how the network behaves as whole as the model is deployed to more and more devices. Since the authors are intentionally vague in their "solutions" it's hard to specify how such project would implement such solutions rather than keep it in mind as the project progresses. Due to the criticality of the network as a whole, if such a project would fail to account for this type of scalability it could lead to cascading node failure. Data, refinement of data pre-processing, and model retraining will need to be a continous process as the threat and network topology constantly changes. As more models adopt such data driven routing, it will further affect how other nodes behave, and there might even be loops between nodes as such a model might not take into account other nodes being data-driven, for example. To take advantage of the network nodes, the engineers could make each nodes responsible for it's own data collection and pre-processing. This can help identify anomalies associated with specific nodes, moreover the engineers might account for data distribution differences between nodes in an automated fashion to account for such anomalies. It might also be pertinent to rely on a hybrid approach between classical method, data driven methods, and human intervention to find issues related to individual nodes. Project developers should thus consider how for example maintainers of such nodes report issues and how this data can be incorporated into training in an automated fashion.

My own research is mainly with respect to cascaded models, that is fast models for easily identifiable data, and slow models for complex data. This is novel within the sphere of routing. It is not hard to imagine that an absolute majority of packages are extremely similar, should always be routed the same way and be easily identifiable, at least for specific nodes. In that case using a cascading model that can rapidly make a routing decision without hogging compute resources is very beneficial. Because of the added complexity of such models, it would probably be an afterthought though in an already

working deployment to increase energy-efficiency. In other words it could be a "technical debt" with respect to energy efficiency being an afterthought. With now the extra layer of complexity, an already solid automated data-preprocessing pipeline would help in aiding the data-preprocessing of multi-layered decisions.

5.2 On-Device or Remote? On the Energy Efficiency of Fetching LLM-Generated Content

The authors have identified a need to measure the tradeoffs between on-device and cloud generated LLM content. The advantages of on-device generation should not be a surprise to any user of such models, no network needed nor privacy leaked. In a world increasingly aware of the intrusive data collection from not only ML engineers but companies and malicious actors as a whole, distribution of models become increasingly in demand. Moreover, as data become saturated in AI/ML companies, that is, it doesn't add much more value, the processing of data become expensive. It is in AI/ML relevant companies interest to decentralize work to decrease compute cost on their own servers. Thus the authors make a first attempt to see what the tradeoffs between edge and cloud is, with respect to experience and energy-efficiency. The authors consider only the energy consumption of the user device, and the effect it has on user experience. It is clear that offloading is significantly cheaper, in order of magnitudes than computing it on device. This has implications on the full battery life of a mobile device, and can thus impact the user negatively. The authors correctly conclude that it is an increasingly complex landscape for engineers to design a system that is transparent, and discriminates efficiently between cloud and on-device compute for optimal user experience and feedback. Something that the authors have obviously left out, is the energy-consumption of the cloud compute itself. While not something that affect user experience, it is something ML engineers need to consider as it affect the compute cost of the company. Moreover, if the energy consumption is larger, because the model on cloud is not as quantized and small, it has a significant impact on sustainability too.

This is something that is directly connected to my research. I'm actively trying to find solutions in how to optimize model discrimination in near lossless performance for increased energy-efficiency. If successful, the research could lead to datadriven methods in how LLM discrimination between on-device and cloud compute is determined. As the authors do not implement a new idea but rather highlights a new aspect of the complexity of the field, it is informative to know what else to consider in such a system.

To take my own work as an example for a larger AI-intensive software project, a small efficient discrimination model could be trained to pre-process the input prompts. This discrimination model would consider different aspects of the input, such as privacy, enery-efficiency, phone status, or context to decide on wether to do inference on-device or in cloud. While some inference would still need to be considered on-device, the model could be made significantly smaller than a full LLM. Furthermore, the pipeline could use cheap classical methods that might decide before the discrimination model, decreasing energy consumption further. As memory is the main culprit in energy consumption for digital devices, the work behind optimization for edge devices during development of a discrimination model would have to focus on that area.

Hard to say how to better integrate the ideas highlighted in this paper as we are already considering it in my current work. The largest change would rather be to actually consider transformer based models as we are still in the early phases of the project.

6. Research Ethics & Synthesis Reflection

To find the research papers I followed the provided link, looked at the program of the latest CAIN conference, picked two titles that interested me and read them. While aware of the task at hand I decided the best course of action was to do what seemed most interesting, as I noticed that a disclosure on "how I avoided LLMs" were part of the assignment. Technical Debt, something that was already brought up in the course, was the most interesting of the days in Gothenburg, thus I picked a paper on that. The second paper seemed most relevant to my own research and so I thought I could hit two birds with one stone. There were no misleading titles, I only checked two and read those two. Though relevant to ML engineering and SW engineering, there were no novelty to the ideas, both mainly investigated issues related to the topic. To ensure orginality, no LLMs were used in the writing of this paper, neither search, summaries, nor the actual writing. While no source control to check if my random scribbles is novel or not have been done, I take my chances in submitting them as is. To ensure no copying from sources were made, the reading was done one day before writing. Only short checkups were made to ensure validity of my arguments.