Particle Transformer - DeepSoCFlow

Project Specification

**Abijit Jayachandran, Andrew Masek, Juan Yin**

**CSE 145 / 237D Spring 2024**

# 1. Project Charter

## 1.1 Project Overview

The Large Hadron Collider (LHC) at CERN works by colliding particles at high speeds and measuring what particles are created by the collisions. One issue with the current process is that the data the LHC creates is in the form of particle jets, which are large datasets that can be processed to find out exactly what particles are created in the collisions. To avoid the need to store all of the particle jet data itself, machine learning models such as the Particle Transformer (ParT) are used to process the particle jet data on the edge so only the data regarding which particles were created needs to be stored. Currently, these machine learning algorithms run on GPUs which are costly and power-inefficient, so our project looks to move the models onto FPGAs which are cheaper and more efficient.

## 1.2 Project Approach

To move the ParT onto an FPGA, there’s a lot of work that needs to be done on the software side. First, we need to build up an understanding of the model and make sure that we can reproduce the results presented in the model paper. Next, we need to port the model to QKeras to quantize the model. In particular, we are going to try to lower the number of precision bits used by the floating point calculations that the model does and record the loss in accuracy at different bit counts. This quantization should make deploying this large transformer model onto a modest FPGA much more realistic and should improve inference speed drastically. Once we have our quantized model, deploy the model onto an FPGA using DeepSoCFlow. Finally, we will validate our implementation of the ParT on the FPGA.

## 1.3 Minimum Viable Product

Our minimum viable product (MVP) is to reimplement the [NanoGPT model](https://www.youtube.com/watch?v=kCc8FmEb1nY&t=2286s) in QKeras. Since our full project is to reimplement the ParT model from Pytorch to QKeras, reimplementing a smaller network seems like a worthwhile stepping stone to familiarize ourselves with the relevant technical skills. We are not too concerned with maintaining high accuracy for the MVP.

Implementing this MVP will involve multiple steps, each of which will help us in eventually reimplementing the ParT model. First, we will have to create the nanoGPT model in Pytorch, this will help us to understand the existing ParT model implementation. Next, we will have to reimplement the model in QKeras. This will likely be the hardest part of creating the MVP, but also the most important towards our final goal. This step will involve learning QKeras and translating the different layers and functions that make up NanoGPT.

Finally, we will evaluate our NanoGPT model by comparing the accuracy and inference speed metrics of the original implementation versus the QKeras implementation. We will also evaluate the metrics when the model is quantized at different levels using QKeras. This should help us to understand the tradeoffs of different types/ levels of quantization when using Transformer models. The graphs demonstrating these tradeoffs will also be useful for a mid-quarter project presentation.

Once we have our MVP complete, we can iterate on it by using the codebase as a template as well as the skills we learned towards reimplementing the ParT model. We will follow the same steps in generally the same order: understanding the Pytorch implementation, reimplementing the layers and functions into QKeras, and finally evaluating the final implementation.

We would iterate towards this long-term goal by focusing on accuracy or efficiency depending on our MVP's appearance. To evaluate the MVP, we plan on graphing its accuracy and efficiency at varying levels of precision and comparing the results to the original model. This will show us our current status and what steps we need to take next. For example, if we find that our accuracy is much lower than the original model, in our next few milestones we need to focus on raising this value. However, if we find that our model is still very inefficient and compute-intensive, our next steps would be to focus on increasing performance instead.

## 1.4 Constraints, Risk, and Feasibility

| Risk | Risk Description | Risk Mitigation |
| --- | --- | --- |
| Trouble Understanding ParT Implementation | This is somewhat guaranteed as the ParT implementation unfortunately has very poor documentation. An understanding of the model and in particular its implementation is important to properly port the model to QKeras. | Reach out to Aba and Zhenghua for help as they have more experience with the model and ML in general.  Reach out to authors via Github issues if needed. |
| QKeras Migration Through H5 File Fails | The ParT is an extremely complex model, and porting the model directly to QKeras would likely take a team of PHDs multiple quarters. Our plan involves exporting the ParT as an h5 file, and importing it in Keras. From there we should be able to move it QKeras without much difficulty.  The risk here is in our ability to export and import the model in Keras without breaking anything or reducing accuracy significantly. | Follow an exporting and importing tutorial first  Reach out to Aba and Zhenghua for help |
| Quantization has a more drastic effect on model performance than expected | There's a possibility that moving the model to QKeras and then lowering the bit precision hurts the model performance more than we expect. While this doesn’t seem too likely, it would have a large impact on the project. We would have to take extra time to explore other options to make the model smaller/ faster, and then implement those instead of just simple quantization. | Discuss expectations about model performance with Aba  Go through a simple QKeras quantization tutorial to make sure we know how to do bit quantization properly. |
| Issues with DeepSoCFlow | DeepSoCFlow is a relatively new and untested library since it is still under development. We may run into issues with compatibility or some bugs with DeepSoCFlow which prevent us from using the library to port the QKeras model to the FPGA. | We will have to rely on Aba and Zhenghua largely in this case, as they are the developers of DeepSoCFlow, and our team doesn’t have too much relevant experience |

# 2. Group Management

## 2.1 Major Roles

Documentation Lead - Juan

In charge of documenting the code in our repository, making sure the folder structures are organized, and creating our web presence through the ReadMe file

Manager - Abijit

In charge of making sure that each team member is progressing in their milestones at the pace assigned in this project specification. If not, the Manager will work with the team to catch back up.

Presentation Lead - Andrew

In charge of creating the slides and graphics for the presentations and videos for the class itself.

First off, we've got the Documentation Lead, who's in charge of managing our web presence and making sure all our project documentation is up to snuff. Then there's the Manager, who keeps us on track with deadlines, milestones, and deliverables, making sure we're hitting our goals on time. Lastly, we have the Hardware Lead, who's diving deep into the technical side of things, especially with DeepSoCFlow. With these roles in place, we've got all our bases covered and can work together effectively to bring our project to life.

## 2.2 Decision Making

In our project to build a transformer, decisions will likely be made by all of us together, rather than just one person taking charge. Since we're all still learning and might not have a ton of experience, it makes sense to talk things out and come to an agreement that we all feel good about. We can share what we know and figure out the best path forward as a team. Plus, since we're new to this, it's smart to ask for advice from Aba and Zhengua who originally pitched this project. Getting their input can help us avoid mistakes and make sure we're on the right track.

## 2.3 Communication

We have three main methods of communication: Discord, weekly physical meetings, and GitHub. We use Discord for daily updates and to message each other about blockers. We also plan on having at least one in-person meeting every week after Thursday’s class to discuss tasks and progress. Finally, we will also use GitHub (specifically the issue tracking and road mapping features) as a management and communication tool for specific tasks and the timeline in general.

## 2.4 Time Management

We'll stay on top of our schedule by keeping an eye on our timelines and the Gantt chart. These tools help us see if we're falling behind or getting off track. If we fall behind a deadline, we need to address it quickly and adjust our next milestones accordingly. The key will be to identify any issues early which should not be a problem if we communicate as explained in section 2.3.

## 2.5 Assignments

Each milestone has members specifically assigned to them which can be found in section 4. You can see our [Gantt chart](https://github.com/users/abijitj/projects/1/views/1) for a visual representation.

# 3. Project Development

## 3.1 Development Roles

Although each of us has a role (as described in section 2.1), these roles simply mean that that person will take charge or do a little bit more of the work for that category. For example, even though Andrew is the presentation lead, it does not mean the other members will not contribute significantly to the presentations. Similarly, this also means that all the members of the group will be heavily involved in the technical work related to the project.

## 3.2 Hardware/ Software Availability

We’ll be using a variety of open-source Python libraries including Pytorch, Keras, QKeras, and DeepSoCFlow. These are free and easily accessible so accessing the required software shouldn’t be a problem. On the hardware side, we will need an FPGA to test our model which will be provided by Aba.

We also need compute to run and test our models so Zhengua will provide us remote access to a computer that has 200GB of storage and 16GB of RAM to train the model. We need at least 150GB of storage because the training data requires that much space.

## 3.3 Testing

We will be testing our models by comparing the accuracy that we measure on a testing dataset provided in the [ParT paper](https://drive.google.com/file/d/1BSYUc_oQBkVTKmFi03gnsrRHo9N-_UHD/view?usp=sharing) against the reference accuracy of the model from the paper where it was published. We will be doing this testing with the reference implementation, the QKeras implementation, and hopefully with the FPGA implementation.

## 3.4 Documentation

Most of our documentation will be done in our [Github](https://github.com/abijitj/cse145-particle-transformer) repository. We have dedicated project milestones where we update our ReadMe to detail the progress that we’ve made up until that point and how to reproduce our results/processes. In particular, we will have accuracy metrics/ charts as well as all of the code that we create uploaded to Github.

# 4. Project Milestones and Schedule

Our up-to-date Gantt chart that contains milestones and deliverables can be viewed [here](https://github.com/users/abijitj/projects/1).

## 4.1 Training and Evaluation of ParT model

Our first deliverable is to train and evaluate the existing ParT’s performance. This will require us to get familiar with their repository, download their dataset, and run their PyTorch implementation of the model. Our goal here is to check that we can get the same or at least similar accuracy reported in their [paper](https://arxiv.org/abs/2202.03772). We’ll put the result of this accuracy benchmark in our repo in a folder titled deliverable1.

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Hours Expected (per person) |
| --- | --- | --- | --- | --- |
| Create a GitHub project and assign task and timeline | P0 | Everyone | 3 | 1.5 |
| Download dataset and do the model training | P0 | Andrew, Juan | 3,4 | 8 |
| Record the accuracy and evaluation result of the transformer model | P0 | Juan | 4 | 3 |

## 4.2 Initial translation from PyTorch to QKeras

The goal for this deliverable is to check the feasibility of automated translation from PyTorch to QKeras. If it is not feasible, we will spend most of our time learning how to implement QKeras layers from scratch while developing a small transformer architecture in the form of NanoGPT (shown in this [tutorial](https://www.youtube.com/watch?v=kCc8FmEb1nY&t=2286s)).

Andrew will be checking the feasibility of automated translation since he has some experience with it. In the meantime, Abijit and Juan will get familiar with QKeras and transformer architectures. In the case that automatic translation works, the deliverable for this section will be a model file (such as onnx, pt, or h5) storing the QKeras model which will be found in the repo inside a folder titled deliverable2. Otherwise, the deliverable will be a version of NanoGPT implemented in QKeras stored in the same place.

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Hours Expected (per person) |
| --- | --- | --- | --- | --- |
| Export the Transformer model from Pytorch into an h5 file, import in Keras, and evaluate feasibility of automated translation | P0 | Andrew | 4 | 4 |
| If automated translation **is feasible**, complete automated translation | P0 | Andrew | 5 | 8 |
| Create NanoGPT clone from [Aba’s tutorial video](https://www.youtube.com/watch?v=kCc8FmEb1nY&t=2286s) | P0 | Abijit, Juan | 4 | 6 |
| Learn QKeras by following this [tutorial](https://github.com/google/qkeras/blob/v0.9.0/notebook/QKerasTutorial.ipynb) | P0 | Everyone | 5 (Abijit, Juan), 6 (Andrew) | 6 |
| If automated translation is feasible, the following milestones will not need to be completed. | | | | |
| If automated translation **is not feasible**, Andrew catches up by creating NanoGPT clone | P0 | Andrew | 5 | 6 |
| Recreate NanoGPT in QKeras in preparation for the larger model. | P0 | Everyone | 4, 5, 6 | 10 |
| Work on oral project update | P0 | Everyone | 6 | 4 |
| Write a short report comparing our quantized NanoGPT in QKeras with the original NanoGPT. | P0 | Everyone | 5, 6 | 6 |

## 4.3 Further translation from PyTorch to QKeras

By the end of this deliverable, we will have a functioning ParT model in QKeras that is significantly more efficient with (hopefully) similar levels of accuracy. A file holding the model will be stored in a deliverable3 folder in the repo.

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Hours Expected (per person) |
| --- | --- | --- | --- | --- |
| Work on implementing the 1st particle attention block of the model (see Fig.1)\* | P0 | Abijit | 6, 7 | 15 |
| Work on implementing the 2nd particle attention block of the model (see Fig.1)\* | P0 | Andrew | 6, 7 | 15 |
| Work on implementing the 3rd particle attention block of the model (see Fig.1)\* | P0 | Juan | 6, 7 | 15 |
| Work on the Class Attention Blocks (see Fig.1)\* | P0 | Everyone | 7, 8 | 15 |
| Finish up the rest of the model (see Fig.1)\* | P0 | Everyone | 8, 9 | 15 |
| Record accuracy and evaluation results of the QKeras ParT model. Compare to the original. | P0 | Everyone | 9, 10 | 6 |

\* Do not need to do these milestones if automated translation is not available.

## 4.4 DeepSoCFlow implementation and Understanding hardware program language configuration

This deliverable is **an optional extension**, which assumes that we have already created a working ParT model implemented in QKeras and that we have at least a week or two left in the quarter. This deliverable involves understanding DeepSoCFlow and then using it to put the QKeras ParT model onto an FPGA and run it. The deliverable consists of an evaluation of model inference speed and accuracy in the form of text or a graph stored in the repo in a folder called deliverable4.

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Hours Expected (per person) |
| --- | --- | --- | --- | --- |
| Read DeepSoCFlow documentation and speak to Aba about using it | P2 | Everyone | 8 | 5 |
| Migrate QKeras model to FPGA using DeepSoCFlow | P2 | Everyone | 9, 10 | 10 |
| Update the README file in GitHub(add update “How to use DeepSoCFlow to port the QKeras model to an FPGA”) | P2 | Juan | 9, 10 | 2 |
| Measure and evaluate model speed and accuracy. Compare results to existing GPU results. | P2 | Everyone | 9, 10 | 6 |

## 4.5 Complete final documentation and presentation

This deliverable involves creating the final documentation and presentation for the course. This deliverable involves the README file and other documentation files in our repository and the final project video.

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Hours Expected (per person) |
| --- | --- | --- | --- | --- |
| Clean up the repository and last-minute changes. | P2 | Abijit | 10, 11 | 6 |
| Prepare the PPT or web page for [Project Webpresence](https://kastner.ucsd.edu/ryan/cse145/project-webpage/)  (Due week 10 Tuesday, **Jun 4** ) | P2 | Juan | 9,10 | 8 |
| Prepare [Final Project Video](https://canvas.ucsd.edu/courses/54613/assignments/779682)  (Due Week 11 Tuesday, **Jun 10**) | P2 | Everyone | 10,11 | 10 |