**Particle Transformer**

**Milestone Report**

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

**CSE 145 / 237D Spring 2024**

# **1.** Addressing Feedback

In our project specification, we were vague in explaining our project charter. So, we will take this opportunity to clarify some of the questions that were posed about this section in the project specification and explain how we have addressed any other feedback received this quarter so far.

Feedback:

*“Where are you porting the model from? Is there currently an out-of-the-box PyTorch/Tensorflow implementation? If so, why aren't you experimenting with the existing quantization/pruning libraries that are available for those libraries?”*

We are taking an existing PyTorch implementation of the Particle Transformer ([Paper repo](https://github.com/jet-universe/particle_transformer/blob/main/networks/example_ParticleTransformer.py), [weaver repo](https://github.com/hqucms/weaver-core/blob/main/weaver/nn/model/ParticleTransformer.py#L460)) and implementing it in QKeras. This will involve rebuilding the entire model from scratch because, after doing some research, there seem to be no effective methods to automatically port the model from PyTorch to QKeras.

We need to use QKeras because the DeepSoCFlow library currently only supports QKeras. Additionally, QKeras has some extra flexibility in terms of quantization capabilities compared to PyTorch alternatives. For example, QKeras allows for custom bit quantization, unlike PyTorch libraries which are limited to a few set values.

Feedback:

*“I would advise looking into some backup plans if quantization doesn't produce the desired results. I know that pruning and knowledge distillation are techniques commonly used in the field to produce smaller models with minimal drop in accuracy.”*

This is a good point and we briefly discussed these options with Aba. For now, quantization is our best approach because it is the simplest to implement using QKeras. After we attempt quantization (successfully or not), we can then proceed to look at other more sophisticated methods of producing efficient models such as pruning or knowledge distillation. We will be keeping these options open if quantization does not produce any results.

Feedback:

*MVP is a little detached from the final product. “Have you mapped out any fundamental differences between NanoGPT and ParT?” How does this help build out the ParT model?*

We understand that the NanoGPT model is a somewhat different model compared to ParT. However, we still feel it is a good MVP because it will essentially act as the base for the rest of our work. By implementing NanoGPT in QKeras, we believe that our work building ParT will be greatly simplified. NanoGPT will allow us to essentially copy-paste a lot of the transformer portions of the ParT model which is a major component within the model.



Above is the full diagram of the Particle Transformer from the paper that introduced it. The two main components are different types of attention blocks, Particle Attention Blocks and Class Attention Blocks.

The Particle Attention Blocks are the same thing as normal attention blocks except that “U” is added before the softmax. U is an embedding created to represent pairwise interactions between the different particles in the Jetstreams and since we have the data for all the pairwise interactions recreating this shouldn’t pose a major challenge.

The second type of attention block we need to create is the Class Attention Block which is the same thing as a normal attention block except that it has a class token passed in to a few places (the concat, the query for the MHA, and added after the post attention layer norm). This should also be quite doable as it's very similar to our normal attention block and we can look to their PyTorch implementation as a reference for how to generate the class token and pass it through the model.

Feedback:

*During the oral update, the guests asked us if we had some form of performance metric to measure our QKeras implementation against.*

This was a very good question brought up by the guests during our oral update. After the presentation, we reached out to Aba and Zhengua to ask if there were any performance metric requirements that DeepSoCFlow had for it to successfully implement the model onto an FPGA. As of the writing of this paper, Aba has informed us that the CGRA generated by DeepSoCFlow can support about 307 GOP/s or 307 billion multiply-accumulates (MACs) per second. If we can calculate the number of MACs in the ParT transformer architecture, we can use this metric to estimate the time required to run inference on a given input.

We also asked if there was a limit to the number of parameters our model could have and Aba said that the number of parameters wouldn’t be an issue. The inference time will proportionally get slower with larger models, but DeepSoCFlow should still be able to successfully run it on an FPGA. We plan on further discussing the details of the performance metrics with Aba.

# 2. Group Management

To remind the reader, we had assigned roles to each of our team members in the project specification. These roles highlight one member who will focus on that aspect of the project. However, all members will still contribute to all areas as required.

As documentation lead, Juan has been keeping track of the evidence for our deliverables and also worked on our web presence in our [GitHub repository](https://github.com/abijitj/cse145-particle-transformer).

As the manager, Abijit has been responsible for the milestones and assignment deadlines. This involves managing our [Gantt chart](https://github.com/users/abijitj/projects/1/views/1?layout=roadmap&sortedBy%5Bdirection%5D=asc&sortedBy%5BcolumnId%5D=90145146) and adjusting tasks/dates as necessary.

As the presentation lead, Andrew developed a large portion of our oral update presentation slides and also created the initial outline for the presentation.

Additionally, it is important to note that all of the members (to varying capacities) have also contributed to the technical aspects of the project. The specific tasks each member completed will be described in the upcoming sections.

# 3. MVP Completion

Our MVP for this project is to implement [NanoGPT](https://github.com/karpathy/ng-video-lecture) in QKeras by converting it from PyTorch. This is also our 2nd deliverable (as explained in the project specification). Overall, even with a few delays, we have made good progress on the MVP and plan on completing it in the next two days.

First, we all set out to work on the PyTorch to Keras conversion of the NanoGPT model. This conversion took longer than expected because we ran into various issues with the differences between PyTorch and Keras functions such as when calculating loss. However, in the end, we were able to successfully convert the model to Keras and achieve similar loss and accuracy results compared to the PyTorch implementation. Please see this [video](https://drive.google.com/file/d/15SEx0Ov1WwXrLdE0PLzDlPpzZFuaLJH5/view?resourcekey) for training and output results. 

Next, we started the process of converting to QKeras from Keras. As of right now, we have converted all the Keras Dense layers to QKeras QDense layers (please see the results of training this model in Fig.2). The model is achieving similar levels of loss as the original PyTorch NanoGPT model. Next, we will evaluate the performance of this model at different bit precisions and compare it to the original NanoGPT in PyTorch. Our current work on this can be found [here](https://github.com/abijitj/nanoGPT/blob/main/gpt_keras.py).

# 4. Milestone Completion

## 4.1. Deliverable One: Training and Evaluation of ParT model

Our first deliverable was to train and evaluate the existing ParT’s performance.

Completed Delayed completion In-progress New Milestone

| Milestones | Priority(0-2) | Member(s) Involved | Weeks Worked on | Approx. Hours Spent (per person) |
| --- | --- | --- | --- | --- |
| Create a GitHub project and assign tasks | P0 | Everyone | 3 | 2 |
| Download the training 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 |

For our first deliverable, we set out to train and evaluate the existing Particle Transformer model. The goal was to check that we could replicate the results that the paper achieved. In the paper, their model had an accuracy of 86% on their test dataset. After training for 5 epochs, we achieved a validation accuracy of 84.9% (see screenshot below).



Although this is less than 86% and it is on the validation dataset, after discussing with Aba and Zhengua we determined that this value was enough to conclude that the ParT model is functional. We can now proceed with the QKeras translation.

You can also find our current main repository [here](https://github.com/abijitj/cse145-particle-transformer).

## 4.2. Deliverable Two: NanoGPT

The second deliverable that we aimed to complete by this point in time was our MVP. This deliverable also involved some time spent evaluating the feasibility of automated PyTorch to Keras translation and also getting more familiar with the literature around this topic (not included as an explicit milestone in this deliverable).

Completed Delayed completion In-progress New Milestone

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Approx. Hours Spent (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 |
| Individually create a NanoGPT clone in PyTorch from [Karpathy’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 not feasible**, Andrew catches up by creating NanoGPT clone | P0 | Andrew | 5 | 6 |
| Recreate NanoGPT in Keras | P0 | Andrew, Abijit | 5, 6 | 12 |
| Work on oral project update | P0 | Everyone | 6 | 5 |
| Switch Keras layers to QKeras | P0 | Everyone | 5, 6 | 8 |
| Write a short report comparing our quantized NanoGPT in QKeras with the original NanoGPT in PyTorch | P0 | Abijit | 5, 6 | 6 |

Initially, during this period, Abijit and Juan focused on recreating [Andrej Karpathy’s nanoGPT](https://www.youtube.com/watch?v=kCc8FmEb1nY&t=2286s). We created an individual repository so that each person could learn as much as possible individually before coming together and comparing thoughts ([Abijit’s repo](https://github.com/abijitj/nanoGPT), [Juan’s repo](https://github.com/JuanYin1/smallGPT_keras)). In parallel, Andrew worked on figuring out if automated translation from PyTorch to Keras is feasible. In the end, we determined that it was not possible because some layers in the ParT model are simply not supported by the existing tools ([see GitHub issue](https://github.com/abijitj/cse145-particle-transformer/issues/13)). Andrew later caught up by make his own NanoGPT clone [here](https://github.com/portoaj/NanoGPT-Fork).

The next step of this deliverable was working on our MVP. This has been described in Section 3.

# 5. Future Milestones

Our upcoming milestones are showcased below. Please note that we have modified and added some milestones (compared to our project specification) based on our current progress and the feedback received.

## 5.1. Deliverable Three: Particle Transformer in 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.

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Hours Expected (per person) |
| --- | --- | --- | --- | --- |
| Calculate the number of multiply-accumulates (MACs) in the Particle Transformer to estimate inference time with DeepSoCFlow | P0 | Abijit | 7, 8 | 5 |
| Work on developing the particle attention block in Keras (see Fig.1)\* | P0 | Andrew | 7, 8 | 15 |
| Work on implementing the Class Attention Block in Keras (see Fig.1)\* | P0 | Abijit | 7, 8 | 15 |
| Understand and figure out how to represent the input embeddings in Keras (see Fig.1\*) | P0 | Juan | 7, 8 | 10 |
| Connect all the blocks and add the remaining layers (see Fig.1)\* | P0 | Everyone | 8, 9 | 15 |
| Convert the entire model to QKeras | P0 | Everyone | 8, 9 | 10 |
| Experiment with different bit precisions | P0 | Everyone | 9, 10 | 10 |
| Record accuracy and evaluation results of the QKeras ParT model. Compare to the original. | P0 | Everyone | 9, 10 | 6 |

## 5.2 Deliverable Four: DeepSoCFlow implementation

This is **an optional extension** which assumes that we have already created a working ParT model in QKeras and that we have at least a week or two left in the quarter.

| 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 | 5 |
| Measure and evaluate model speed and accuracy. Compare results to existing GPU results. | P2 | Everyone | 9, 10 | 6 |

## 5.3 Deliverable Five: Final documentation, presentation, and report

This deliverable involves writing the final documentation, report, and presentation for the course.

| Milestone Tasks | Priority(0-2) | Member(s) Involved | Weeks Worked on | Hours Expected (per person) |
| --- | --- | --- | --- | --- |
| Write the literature review portion of the report | P0 | Abijit, Andrew | 9, 10, 11 | 10 |
| 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** ) | P1 | Juan | 9,10 | 8 |
| Prepare [Final Project Video](https://canvas.ucsd.edu/courses/54613/assignments/779682)  (Due Week 11 Tuesday, **Jun 10**) | P0 | Everyone | 10,11 | 10 |
| Prepare Final Report  (Due Week 11 Friday, Jun 13) | P0 | Everyone | 10, 11 | 10 |

# 6. Rest of Quarter Timeline

Please see our Gantt chart below which visualizes all our upcoming milestones mentioned in the previous section. Please see the up-to-date interactive GitHub project [here](https://github.com/users/abijitj/projects/1/views/1?layout=roadmap&sortedBy%5Bdirection%5D=asc&sortedBy%5BcolumnId%5D=90145146&sortedBy%5Bdirection%5D=asc&sortedBy%5BcolumnId%5D=90145133) for a clearer picture.

