1. Change of project scope and goal

The goal we stated in the project proposal was to implement a lightweight framework which supports the training and prediction of a Convolutional Neural Network framework which outperforms PyTorch based on C++ and CUDA. We wanted to outperform PyTorch in some corner cases that PyTorch does not perform well. However, after further learning of PyTorch and its source code, we realized that this goal is too aggressive for the following reasons:

(1) PyTorch itself is based on highly optimized Basic Linear Algebra Subprograms on CPU and GPU devices. It is hard to do any optimization without proficient knowledge of hardware;
(2) Implementing a framework which supports the training of a Convolutional Neural Network needs implementing a whole bunch of components, including Linear Layers, Convolutional Layers, Batch Normalization Layers, Activation Functions, Pooling Layers and so on.

Besides the above obstacles, we also realized that what we've proposed is not focusing very much on parallel programming. So we decided to revise our goal and scope while maintaining the topic to be increasing the efficiency of a deep learning model/component. After further study for PyTorch, we found an interesting topic which meets the requirement which is to **implement a basic module (such as LSTM) based on PyTorch and optimize its performance using PyTorch's C++ and CUDA extension**. This revised goal is concentrating on parallel topic and is challenging if we want to outperform PyTorch's default implementation.

With respect to the plan we listed in our proposal, we should learn PyTorch source code and implement the CPU version of the framework before midterm. Although our goal is changed, we still learned what we need from PyTorch source code and implemented a baseline version of our project. So, we are still closely sticking to our original plan.

2. What we have completed so far

Firstly, we found a focus for this project. To make it a meaningful project, we should implement a module which is both meaningful and original. By meaningful, this module should be useful in a deep neural network (LSTM satisfies this requirement). By original, this module should not be found implemented using PyTorch extension in any open source project (LSTM does not satisfy this requirement) we can refer to. After reading several papers, we've decided to implement a second order function described in "PAY LESS ATTENTION
WITH LIGHTWEIGHT AND DYNAMIC CONVOLUTIONS" from ICLR 2019. The high order convolutional layer described in this paper improves the learning effectiveness of the Convolutional Neural Network, and it is not implemented in either PyTorch or any other open source projects using Pytorch extension.

Secondly, we delved into greater detail by reading PyTorch documents and source code and focus on what is more reasonable to implement in this project. As previously mentioned, there exists great obstacles in optimizing low level layers such as TH/THC and BLAS/cuBLAS. Instead we'll be implementing base on the torch::tensor layer, which implements an abstraction based on TH/THC layer. What we would do will be to implement the CPU-based algorithm using C++ and to implement the GPU-based algorithm using C++ & CUDA. Note that although a PyTorch & Python based implementation of the algorithm can run both on CPU and GPU, it is not very well optimized especially for on the GPU side.

Thirdly, we've got preliminary results by implementing a baseline model. We implemented the second order function we are going to optimize using PyTorch's Python extension which takes about 213 seconds to run through (including forward pass and backward propagation) 1000 images from COCO dataset on CPU. Further analysis will be performed after we get access to a machine with GPU and PyTorch.

3. What do you plan to show at the poster session

1. Details for each implementation.
2. Optimizations we use to achieve better performance and why we try these.
3. We will show our results using charts and graphs comparing the performance of different implementations with different metrics.They will include the following contents:

	Python(CPU)	Python(GPU)	C++(CPU)	CUDA(GPU)
Time(Simple Model)				
Memory Utilization(Simple Model)				
Time(Real Model)				
Memory Utilization(Real Model)				

• What about the "nice to have"? In your checkpoint writeup we want a new list of goals that you plan to hit for the poster session.

1. Must Have: Complete all the versions. And experiments results on both simple model and real model should have better performance with our best implementation compared to the Python version with PyTorch's Python extension.
2. Nice to Have: https://pytorch.org/tutorials/advanced/cpp_extension.html Based on the examples in document of PyTorch, we expect our custom GPU implementation to

achieve around 80% more efficient than the Python version with PyTorch's Python extension on GPU devices.

4. List the issues that concern you the most

We read the source code of PyTorch, we find they did a good and mature job at the low level API implementation. Their APIs are written in C and utilized multiple dispatch. So it can select SIMD acceleration according to the environment variables, hardware instruction implementation and other conditions. This may make the strengths of our optimizations less obvious.

5. Make sure your project schedule on your main project page is up to date with work completed so far, and well as with a revised plan of work for the coming weeks. Need a very detailed schedule for the coming weeks. I suggest breaking time down into half-week increments. Each increment should have at least one task, and for each task put a person's name on it.

Date	Task	Assign
Nov.18 ~ Nov.20	Implementation of CPU version	Simeng
Nov.21 ~ Nov.24	Implementation of GPU version	Zhejin
Nov.25 ~ Nov.27	Optimization of GPU version	Simeng, Zhejin
Nov.28 ~ Dec.1	Experiments on simple models	Simeng
Dec.2 ~ Dec.5	Experiments on real models	Zhejin
Dec.5 ~ Dec.8	Final reports and prepare for poster session	Simeng, Zhejin