Literature Review

1. **PipeDream-2BW: Memory-Efficient Pipeline-Parallel DNN Training**
   - **Problem to solve:** Out-of-core DNN model, too large to fit into single GPU memory. Moreover, we not only want to solve the problem but also want efficient training of large models, with both high throughput and low memory footprint.
   - **Solution:** PiepeDream-2BW(double-buffered weight updates), a system that performs memory-efficient pipeline-parallel training of DNN models with billions of parameters and is able to achieves high throughput, low memory footprint, and data parallelism-like semantics through a innovative weight update double buffering strategy. (Particularly granted to the its weight gradient coalescing strategy and planning algorithm)
   - **Innovative techniques**:
     - **2BW's weight gradient coalescing strategy**: a technique that reduces the memory footprint of training while avoiding pipeline flushes.
     - **2BW partitioning planning algorithm**: PipeDream-2BW's planner determines how to split a model over the available compute devices by exhaustively searching over the reduced search space of all possible parallel-pipeline configurations.
   - **Result/Impact:** Compared to the hybrid parallelism baseline model, PipeDream-2BW is 6.9x faster for BERT-192, and 5.4x faster for GPT-2 by using 64 GPUs. Compared to the GPipe model, PipeDream-2BW outperforms corresponding GPipe configurations at the same global minibatch size by up to 1.9x due to its lack of periodic pipeline flushes. Compared to PipeDream, PipeDream-2BW is up to 6.2x faster.
   -
2. **ZeRO-Memory: ZeRO: Memory Optimizations Toward Training Trillion Parameter Models**
   - **Problem to solve:** Being able to train large DNN models (billions to trillions of parameters), while retaining high computational granularity, low communication overhead, and eliminate the memory redundancy.
   - **Solution**: Zero Redundancy Optimizer(ZeRO): It can optimize memory and training speed while increasing the model size and eliminates memory redundancies in data- and model-parallel training while retaining low communication overhead and high computational granularity.
   - **Innovative techniques**:
     - **ZeRO-DP**: A type of ZeRO that builds upon DP(Data Parallelism). It aims at reducing the per-device memory footprint of a model while retraining the memory efficiency. In another word, it aims to retain the training efficiency of DP while achieving the memory efficiency of MP, through partitioning them -- optimizer states, gradients and parameters -- across data parallel processes.
     - **ZeRO-R:** Another type of ZeRO that builds upon MP(Model Parallelism). It targets at increasing the memory availability for even larger models by reducing the residual memory consumption/redundancies in MP and the time it takes for the memory allocator to find free contiguous memory.
   - **Result/Impact**:

    - Experimented on ZeRO-100B(a model with up to 170B parameters), enables 8x increase in model sizes, 10x in throughput improvement(41.4 TFlops/GPU), achieves super-linear speedups on modern GPU clusters, and trained the largest model in the world.
    - No model refactoring is necessary, and it is as easy to use as standard data-parallelism.
    - Has the potential to scale beyond 1 trillion parameters by using today's hardware.
    - Declared as a prime candidate for future investigation on large model training.
- **Relevance:** All the papers were recently published at the super computing conference -- SC '20: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis -- which is one of the top conferences in the field of HPC.

3. **GEMS: GPU-Enabled Memory-Aware Model-Parallelism System for Distributed DNN Training**
    - **Problem to solve:** Model parallelism for out-of-core models (more than 1 machine) that has a higher speedup than earlier model parallelism solutions. Currently targeting pathology and large images.
    - **Solution:** GEMS - removes the memory footprint in basic model parallelism by introducing a replica of the model, and switching propagation operations between the two.
    - **Innovative techniques**:
        - **GEMS MAST**: use a replica of the model to fill the memory footprint in MP-basic; synchronize parameters of the models via allreduce operation
        - **GEMS MASTER**: GEMS MAST, but using allreduce half as much (stacking more compute onto the allreduce operation)
        - **GEMS HY**: Can be implemented from GEMS MAST or GEMS MASTER, gives each GPU (model partition) an allreduce operation
    - **Result/Impact:** observed 1.36x speedup in GEMS: MAST and a 1.83x speedup in GEMS: MASTER compared to alternative methods. GEMS introduces more feasible tools to solve overarching problems in digital pathology and other domains that require heavy memory usage.

4. **Training Large Neural Networks with Constant Memory using a New Execution Algorithm**
    - **Problem to solve:** Maintain desired depth of neural networks (NN) while reducing the memory usage during training.
    - **Solution**: Layer-to-layer (L2L): can run very large models independent of depth by applying graph reduction and cross mixed precision to receive results on par with standard NN training.
    - **Innovative techniques**:
        - **Graph Reduction**: abstracts encoder layer architecture so that only one encoder layer is needed, reduces memory footprint from O(n) to O(1) (only 3 layers in model graph needed: one embed layer, one encoder layer, and one class layer).

    - **Cross Mixed Precision:** Keeping floating point (FP) 32 precision on master copy of model while running reduced graph at FP16 precision. Weights updated with FP32 precision, propagations done at FP16 precision
- **Result/Impact**: The depth of models no longer affects the memory available on GPUs as drastically as it has before; this can bring model parallelism problems back to data parallelism potentially, or even reduce the amount of resources needed for model parallelism techniques.

5. **GPipe: Easy Scaling with Micro-Batch Pipeline Parallelism**
    - **Problem to solve:** Finding a way to handle model parallelism with task-indepence and efficiency while supporting any large scale neural network overcoming barriers like memory limitations and communication overhead.
    - **Solution:** GPipe: a flexible pipeline parallelism library which handles scaling of any network expressed as a sequence of layers with efficacy regardless of the network size and architecture.
    - **Innovative techniques:**
        - **Batch-splitting:** GPipe introduces a batch-splitting pipeline algorithm. This algorithm starts by partitioning each layer into cells placed on separate accelerators then splits mini-batches into micro-batches and executes the micro-batches over a cell.
        - **Synchronous Gradient Descent:** Synchronous gradient descent is used for training where gradients are accumulated across micro-batches and then applied at the end of the mini-batch which is consistent regardless of the number of partitions (cells).
    - **Result/Impact:** GPipe provides flexibility and efficiency when used on large neural networks. By scaling up the networks, the model is capable of achieving linear speedup with the number of devices used to train a network. Moreover, GPipe provides flexibility to support any DNN that can be represented as a sequence of layers and reliability by using synchronous gradient descent that is consistent regardless of the number of partitions.
    - **Relevance:** The final version of this paper was published in 2019 by a team of researchers at Google at the gpu technology conference GTC20. This paper provides a great insight on an efficient technique on model parallelism that may inspire other or similar techniques to be implemented in this project.

6. **Channel and Filter Parallelism for Large-Scale CNN Training**
    - **Problem to solve:** Accelerating large-scale training of CNN on complex models and large datasets via model parallelism while keeping low communication overhead and enabling strong scaling. Also, overcoming some limitations of data-parallelism such memory limitations and mini-batch size with wide CNNs.
    - **Solution:** This paper introduces three algorithms that allow for strong scaling, reducing communication overhead with weak scaling, and requiring no hyperparameter tuning. Each algorithm is differentiated by the data movement and computation patterns it uses. The three algorithms are listed in the innovative techniques section below.

- **Innovative techniques:**
    - **Stationary-x**: Avoids communication of input data during forward propagation.
    - **Stationary-y:** Symmetric to Stationary-x swapping forward and backward propagation. It avoids communication of input data during backpropagation as well as the gradient computations.
    - **Stationary-w:** More general algorithm combining the communication pattern of both Stationary-y and Stationary-x algiorhtms. It controls the amount of communication during forward/backward propagation.

  All three algorithms partition the parameters of convolutional layers instead of replicating them for each processor (which is both data and model parallelism).
- **Result/Impact:** The algorithms provided in this paper improve strong and weak training including 4.1x reduction in training time of residual networks and 4x reduction in allreduce overhead. They reduce communication overhead and memory pressure. Moreover, the wider models introduced in this paper provide more accuracy on training/testing the ImageNet dataset.
- **Relevance:** This paper was published in 2019 at the super computer conference SC19. It provides insight on how to accelerate large-scale CNN training using multiple techniques including channel and filter partitioning. The ideas behind these techniques might come in handy in this project especially when dealing with wide models.

Reference:

- **PipeDream-2BW:** Narayanan, Deepak, et al. "Memory-efficient pipeline-parallel dnn training." arXiv preprint arXiv:2006.09503 (2020).
- **ZeRO-Memory:** Rajbhandari, Samyam, et al. "Zero: Memory optimization towards training a trillion parameter models." arXiv preprint arXiv:1910.02054 (2019).
- **GEMS:** A. Jain, *et al*., "GEMS: GPU-Enabled Memory-Aware Model-Parallelism System for Distributed DNN Training," in *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Atlanta, GA, US, 2020 pp. 621-635. doi: 10.1109/SC41405.2020.00049
- **Constant Memory Training:** Pudipeddi, Bharadwaj et al. "Training Large Neural Networks with Constant Memory using a New Execution Algorithm" arXiv preprint arXiv:arXiv:2002.05645 (2020).
- **GPipe:** Huang, Yanping, et al. "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism" arXiv:1811.06965 [cs.CV] (2019).
- **Channel and Filter:** Dryden, Nikoli, et al. "Channel and Filter Parallelism for Large-Scale CNN Training" In The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '19), https://doi.org/10.1145/3295500.3356207 (2019).

Hypothesis

- Use analytical models to estimate execution time for a model split to efficiently split the DNNs across multiple GPUs
- Use PyTorch's Model and Json APIs to understand data flow in DNNs written in PyTorch to implement user transparent model-splitting
- Use CPU offloading mechanism to optimize GEMS-MASTER design

Validation:

- Compare the performance (images per sec) of naive splitting and analytical model splitting
    - Compare the memory consumption
- Validate user transparent model-splitting by splitting complex deep neural network architectures
- Validate whether GEMS design can be applied to larger models that have more parameters using CPU offloading