Survey on Distributed Neural Networks and GPU Programming Frameworks

Razvan Florian Vasile

razvanflorian.vasile@studio.unibo.it

Department of Computer Science
University of Bologna

May 10, 2025

Outline

The Big Picture: DNNs and Survey Need

- **Importance of DNNs and GPU Programming:**
  - *Powerful applications* in computer vision, NLP, audio processing, etc.
  - *Performance driver*: scaling computational power and datasets.
- **Need for this Survey:**
  - A comprehensive survey bridging *general frameworks* with *practical examples* is currently absent.
  - This survey aims to fill this gap by providing practical, *end-to-end implementations* alongside *theoretical analysis*.

Goals and Target Audience

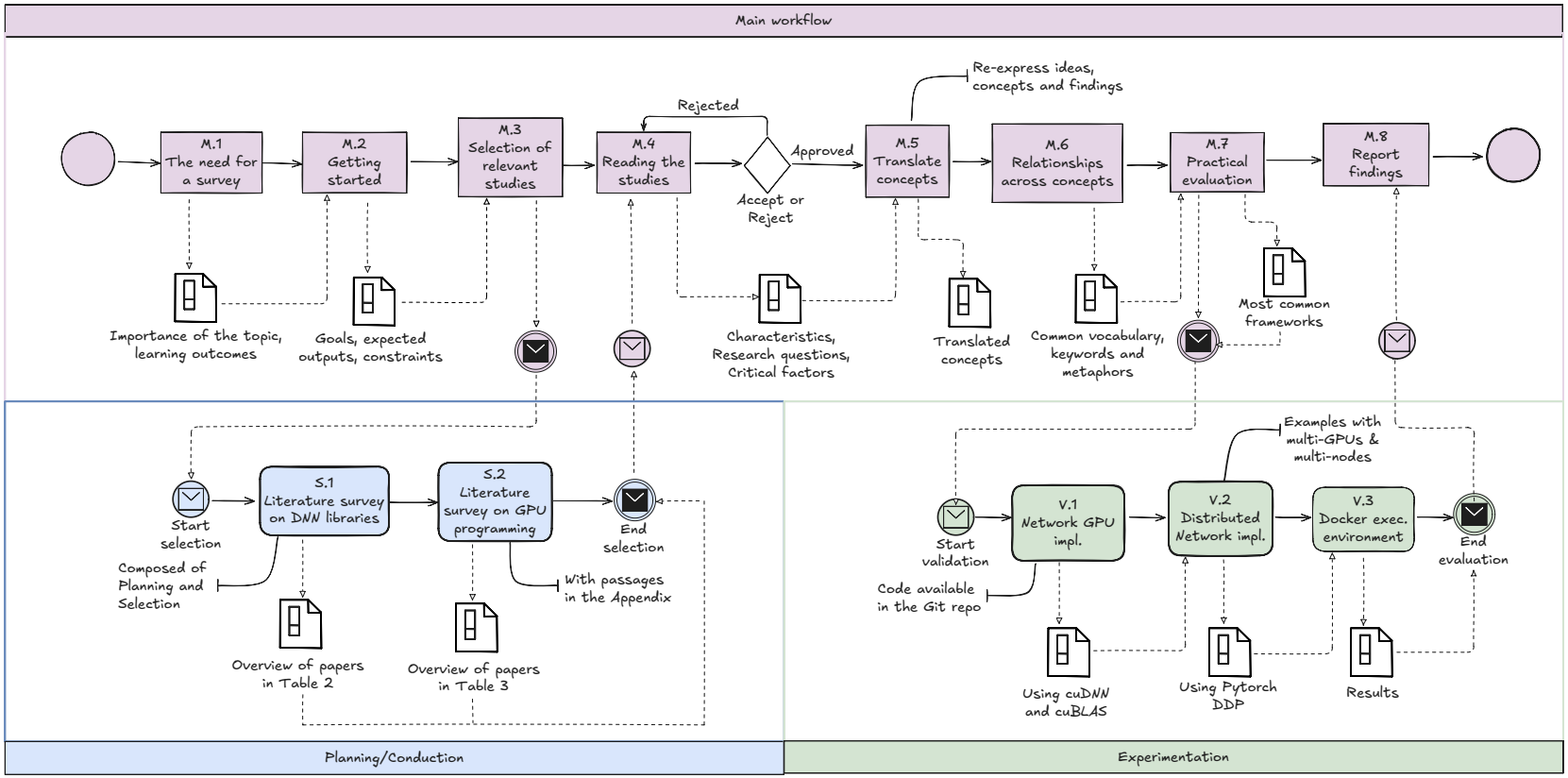
- **Goals of the Survey:**
  - *Analyse strengths and overlaps* of distributed training frameworks for Deep Learning and GPU parallelization libraries.
  - *Gain practical experience* with popular frameworks (PyTorch DDP, cuDNN, cuBLAS).
  - Document the literature *review process*.
- **Target Audience:**
  - Aimed at *students and practitioners*.

Outline

Survey Methodology Overview

- **Main Workflow:**
  - Follows a systematic literature *review protocol*.
  - Key phases: Selection of studies (M.1, M.2, M.3), conducting the review (M.4, M.5, M.6), evaluation (M.7), and reporting (M.8).
- **Minimizing Reviewer Bias:**
  - Systematic reviews should be *transparent* and *replicable* to minimise reviewer bias.
  - The *process is documented* and *artifacts are shown*.

Key Methodological Steps



(Figure 1) Illustration of the review process

Outline

Research Questions (RQ)

Defined using the PICOC criterion.

- **RQ1:** Most commonly cited frameworks for distributed neural network training and community size comparison.
- **RQ2:** Most frequently cited frameworks for GPU programming and user community differences.
- **RQ3:** Overlaps and shared limitations.
- **RQ4:** How these technologies can be applied in practice.

Search Strategy: DNN Frameworks (S.1)

- Manual search in *Scopus, Semantic Scholar, arXiv databases*.
- Inclusion/Exclusion criteria defined (exactly what we are looking for).
- Papers from *2015-2024* considered.
- Tools like *Swift-Review* and *Gemini/NotebookLM* used for semi-automatic selection.
- *Manual inspection* and *snowballing* also used.

Search Strategy: GPU Programming (S.2)

- Not a traditional systematic review due to *proprietary nature* and *focus on documentation/tutorials*.
- Advanced *Github Search* and search engines used for keywords related to AMD, Intel, NVIDIA.
- Inclusion/Exclusion criteria for material like *official documentation, repositories, tutorials*.
- Resources from *2012-2024* considered.

Outline

Distributed Neural Network Frameworks (RQ1)

Details in Table 2 of the report, with citations/stars as community size indicators.

- **Commonly cited:** TensorFlow, PyTorch, MXNet, JAX.
- **Specialized tools:** DeepSpeed, Horovod, Megatron-LM, Transformers, FairScale, Colossal-AI.
- **Cloud platforms:** SageMaker, AzureML, Vertex AI.
- **Characteristics:** Large communities, Python usage, leverage open-source.

GPU Programming Libraries (RQ2)

Details in Table 3 of the report.

- **Algebraic operations:** cuBLAS, rocBLAS.
- **Deep learning primitives:** cuDNN, MIOpen.
- **Communication protocols:** NCCL, RCCL, oneCCL.
- **Cross-platform APIs:** OpenCL, SYCL, Kokkos (cross-hardware).
- **Vendor-specific tools:** HIP, CUTLASS.
- **Language bindings:** CuPy, Numba, Julia packages.

Frameworks Timeline

A timeline of major frameworks in both Distributed DNNs and GPU Programming highlights their evolution.

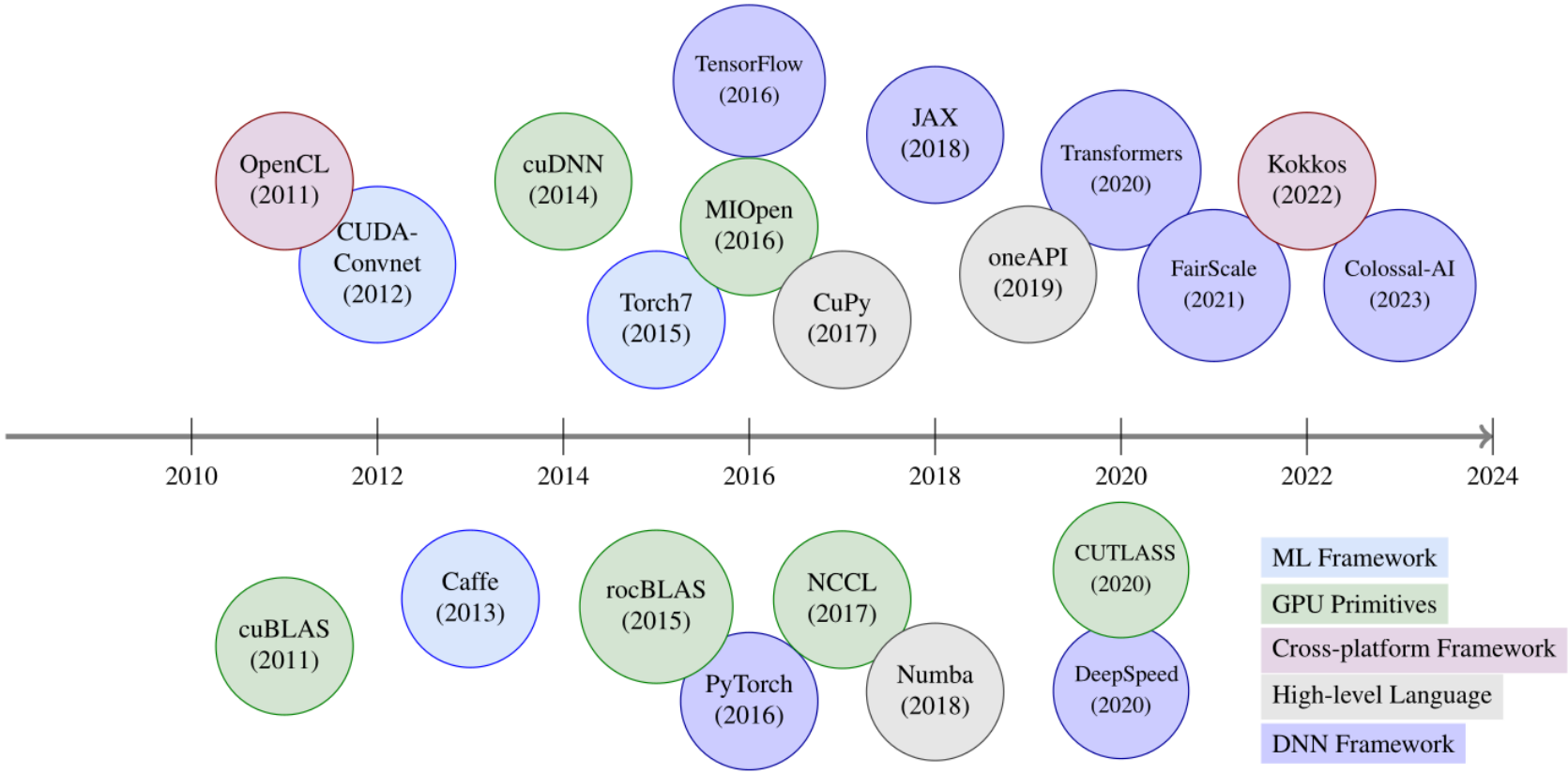


Figure 4. Timeline of Major GPU Programming Libraries and Frameworks

Outline

Shared Motivations

- **Scalability:**
  - *Key motivator* due to increasing *data and model complexity*.
  - *GPU programming* provides the infrastructure for *better performance*.
- **Usability:**
  - Enabling *community access* to *advanced techniques*.
  - *Community vs proprietary* frameworks.

Critical Factors

- **Modularity:**
  - DNNs: abstract *distributed complexity*.
  - GPU programming: used to abstract *hardware details*.
- **Performance Optimization:**
  - DNNs: use *advanced distributed techniques*.
  - GPU frameworks: optimise *specific architectures*.
- **Network Communication:**
  - DNNs: optimise *communication with data, model, pipeline parallelism*.
  - GPU programming: cuDNN lacks multi-GPU support, *user managed*.

Shared Limitations

- **Algorithmic Challenges:**
  - DNNs: related to *memory management across GPUs*.
  - GPU programming: Optimizing for *specific hyperparameter ranges* (e.g. batch size).
- **Community Involvement:**
  - DNNs: *open-source ecosystem*.
  - GPU programming: *proprietary nature* (cuDNN, cuBLAS).

Outline

Purpose and Setup

- **Purpose:** Demonstrate a proof-of-concept implementation of the training process using common frameworks.
- **Frameworks Used:**
  - *cuDNN, cuBLAS* (for GPU programming).
  - *PyTorch DDP* (for distributed training simulation).
- **Setup:**
  - Experiments simulate simple workflows locally using *Docker containers*.
  - Experiments can work on a *single GPU machine*.

GPU Experiment (V.1)

- Implemented a small network with *forward/backward passes, MSELoss, zeroGradients, updateWeights*.
- Used cuDNN and cuBLAS primitives.
- Developed custom GPU *kernels for ReLU* and *MSELoss*.
- **Future work:** Could add more layers (e.g., pooling, dropout, batch normalization).

Distributed Experiment (V.2)

- Used *PyTorch DDP*.
- Simulated distributed training locally using *Docker*.
- **Limitation:**
  - Local simulation on a single GPU does not provide *true multi-GPU/multi-node performance* (due to context switching).
  - However, the code can be reused in a cloud environment.

Outline

Evaluation Metrics and Results

- **Metrics:**
  - Measured *forward and backward pass execution times*, including a warmup phase.
  - Network configurations included *convolutional and fully connected layers*.

Framework	Fwd (ms)	Bwd Input (ms)	Bwd Params (ms)	Total (ms)
cuDNN	0.206760	0.214760	0.028600	0.450120
PyTorch	0.137913	0.112698	0.198538	0.449149

Table: Benchmark results.

Analysis of Results

- *Performance is similar* for both frameworks with similar architectures.
- **Code:**
  - DDP experiment.
  - GPU benchmark.
  - Toy network.

Outline

Summary and Contributions

- **Summary:**
  - The report provides an *introduction to the topic*, while the code provides a *simple implementation of the processes* involved.
- **Future Work:**
  - The *GPU experiments* could be *extended* to more complex primitives.
  - The *distributed experiments* could be *tested* in a *cloud environment*.

Learning Outcomes and Limitations

- **Learning Outcomes:**
  - Gained experience with *literature reviews*.
  - GPU programming familiarity using *cuDNN and cuBLAS*.
  - Practical distributed training experience with *PyTorch DDP*.
  - Practical experience with *Docker*.

Questions?

Thank you for the attention

Thank you for the attention!

Any Questions?