Faculty of Computer Science

International Lab of
Bioinformatics

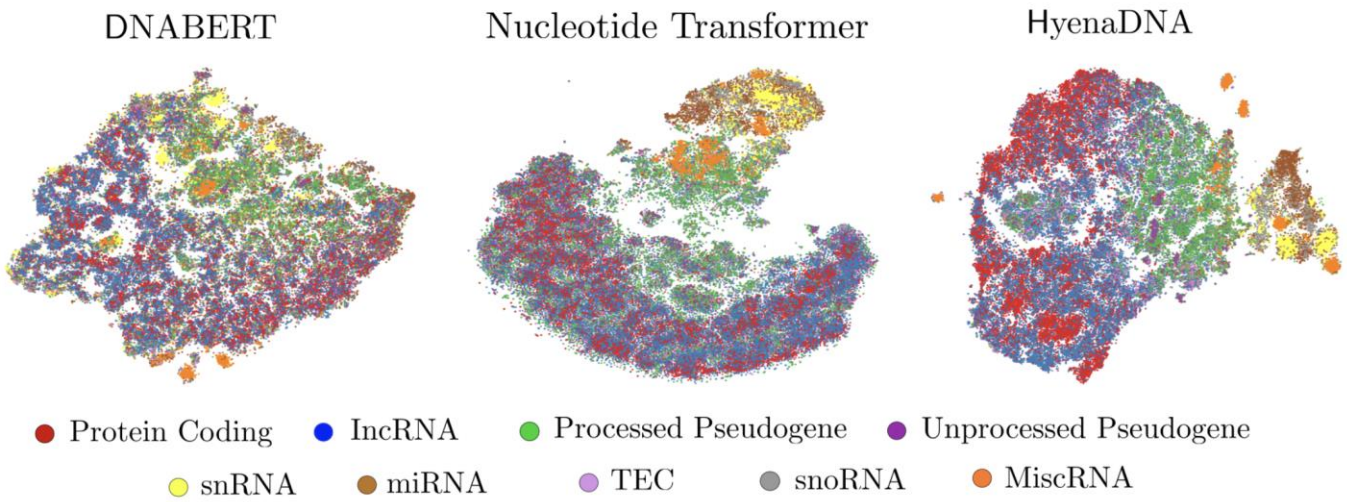
Moscow
PCT - 2025

Optimizing Computational Infrastructure for LLMs in Bioinformatics

Nazar Beknazarov

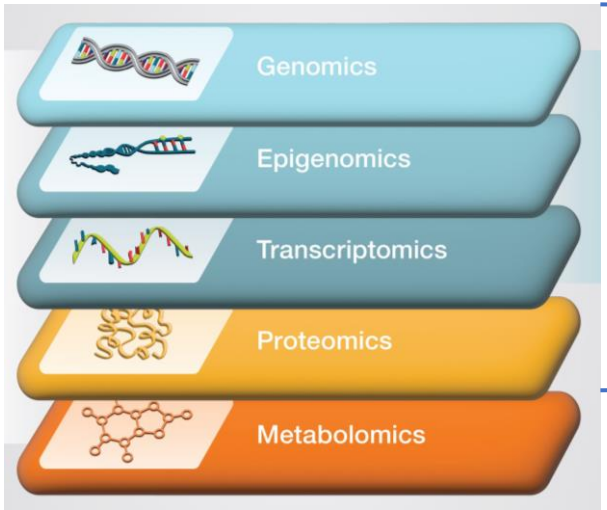
Generalization of models in NLP

Foundational Models in Bioinformatics



Inclusion of OMICS data

Current Models



Proposed Model

Main Task:
Optimization data intensive LLM pipeline for HSE cluster using Pytorch & Lightning

Cluster parameters

Nodes	GPUs per Node	GPU Type
16	4	V100
2	8	A100
1	2	H100

Data parameters

DNA sequence – +3Gb
+1000 BED files - +3Tb

Model parameters

100 M trainable parameters
+65K context length

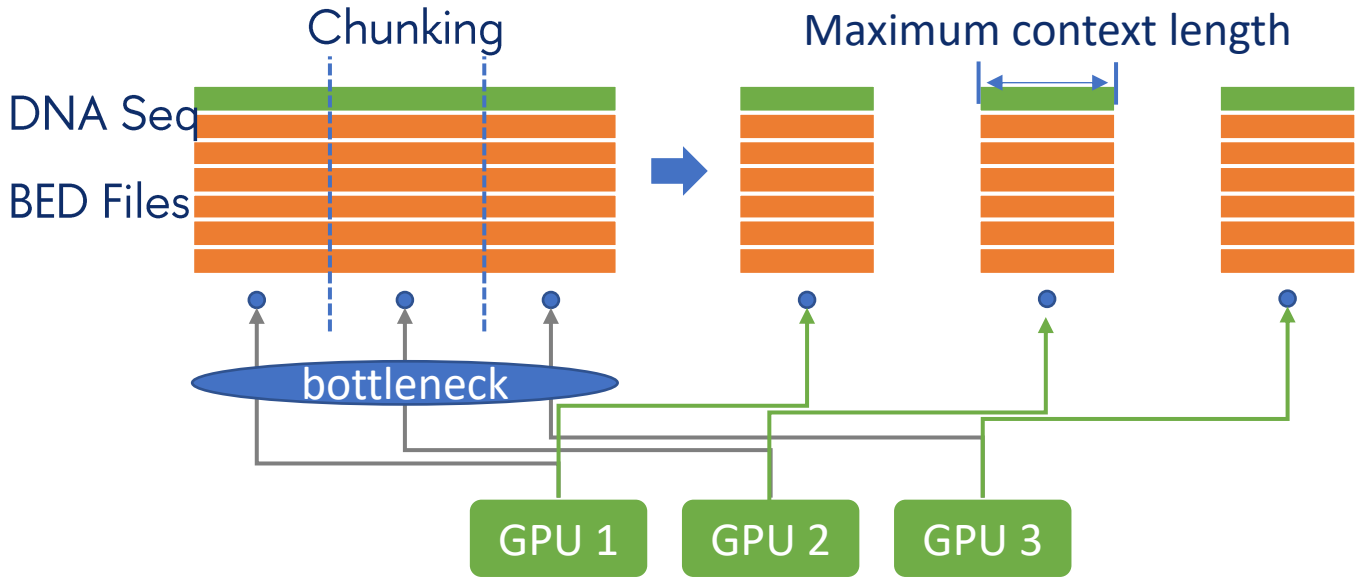
Key Bottlenecks:

1) Shared RAM limitations
2) Disk I/O throughput
3) GPU data transfer latency
4) Context length constraints
5) Model size scalability

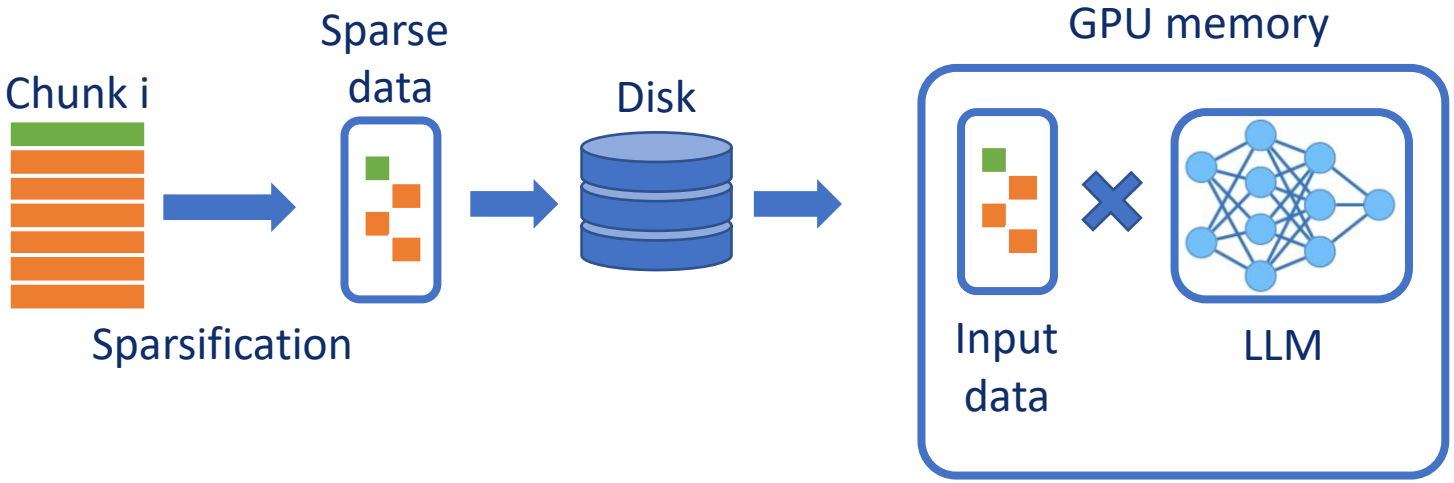
Primary Objectives:

1) Minimize computational overhead
2) Enable parallel processing wherever feasible
3) Maximize GPU utilization

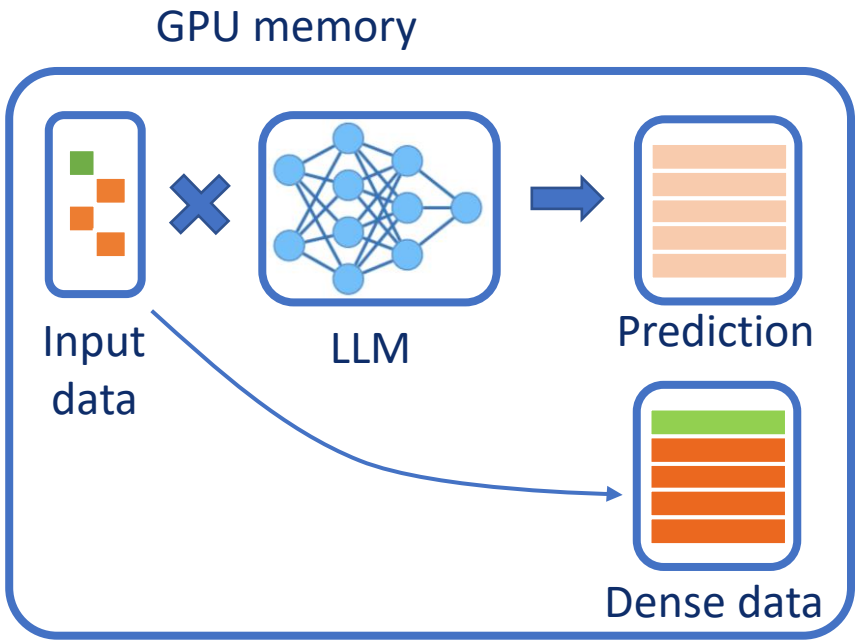
Data chunking



Sparse data pipeline



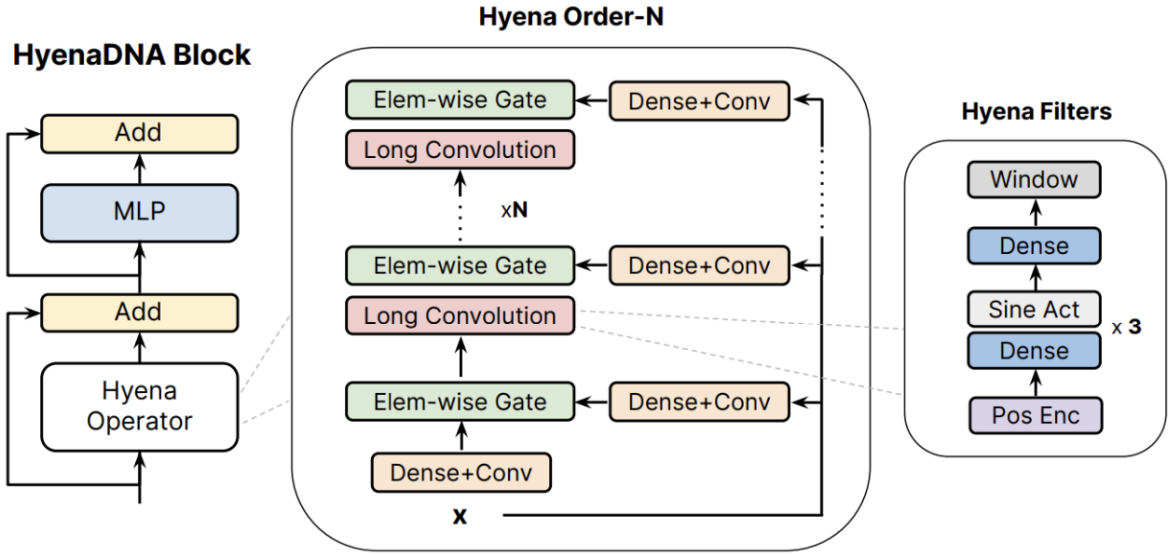
Effective data restorage




```python
def table_to_sparse(table, begin, end, ids_table):
    start_positions = table.Start.values
    end_positions = table.End.values
    values = np.clip(table.Name.values / 1000, 0, 1)
    inds = table.feature_name.apply(ids_table.get).values

    lengths = end_positions - start_positions
    lengths_cumsum = np.concatenate(([0], np.cumsum(lengths))
    )

    repeated_values = np.repeat(values, lengths)
    repeated_inds_y = np.repeat(inds, lengths)
    repeated_inds_x = np.repeat(start_positions - begin,
    lengths) + (np.arange(lengths_cumsum[-1]) - np.repeat(
    lengths_cumsum[:-1], lengths))

    valid_points = (repeated_inds_x >= 0) & (repeated_inds_x
    < end - begin)

    return repeated_inds_x[valid_points], repeated_inds_y[
    valid_points], repeated_values[valid_points]
```

Loop free python code

Model optimization

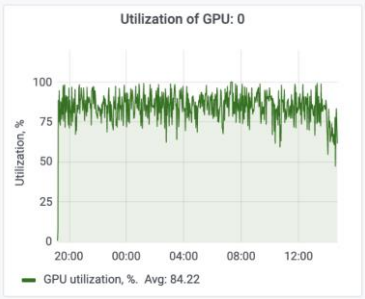


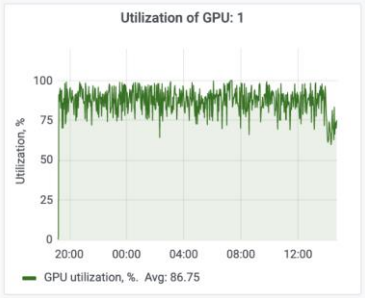
Effective floating-point number formats

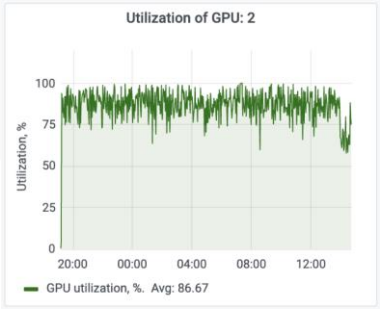
Parameter	**H100 PCIe**	**A100 SXM**	**V100 SXM**
Architecture	Hopper	Ampere	Volta
FP64 (TFlops)	25.6	9.7	7.8
FP64 Tensor Core (TFlops)	51.2	19.5	–
FP32 (TFlops)	51.2	19.5	15.7
TF32 Tensor Core (TFlops)	378	156	–
BF16 (TFlops)	102.4	39	–
BF16 Tensor Core (TFlops)	756	312	–
FP16 (TFlops)	102.4	78	31.4
FP16 Tensor Core (TFlops)	756	312	125
FP8 Tensor Core (TFlops)	1513	–	–
INT32 (TFlops)	25.6	19.5	15.7
INT8 Tensor Core (TOPs)	1513	624	62
INT4 Tensor Core (TOPs)	–	1248	–
Memory (GB)	80	80	32
Memory Bandwidth (GB/s)	2039	1555	900

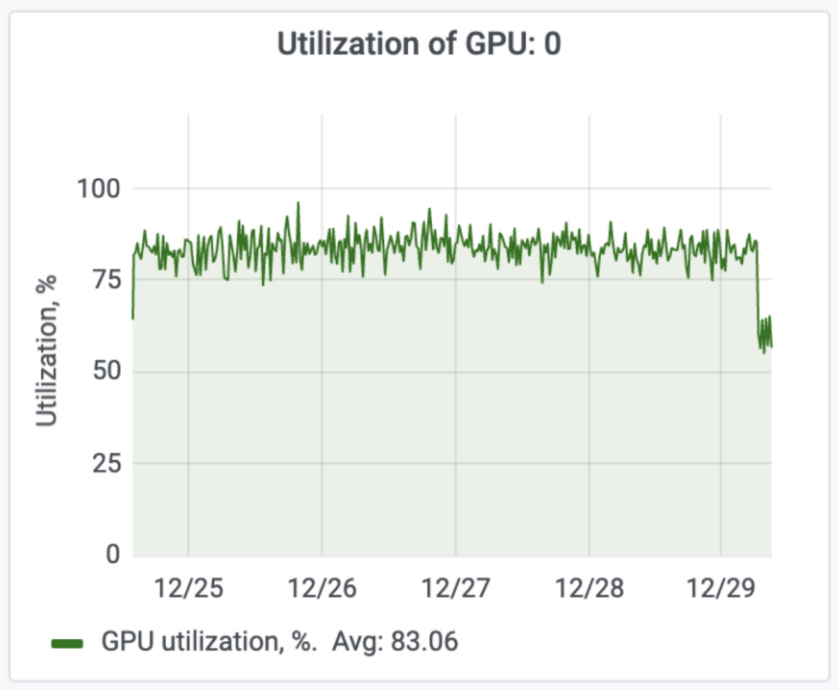
Table 1: Comparison of H100, A100, and V100 GPUs

GPU utilization

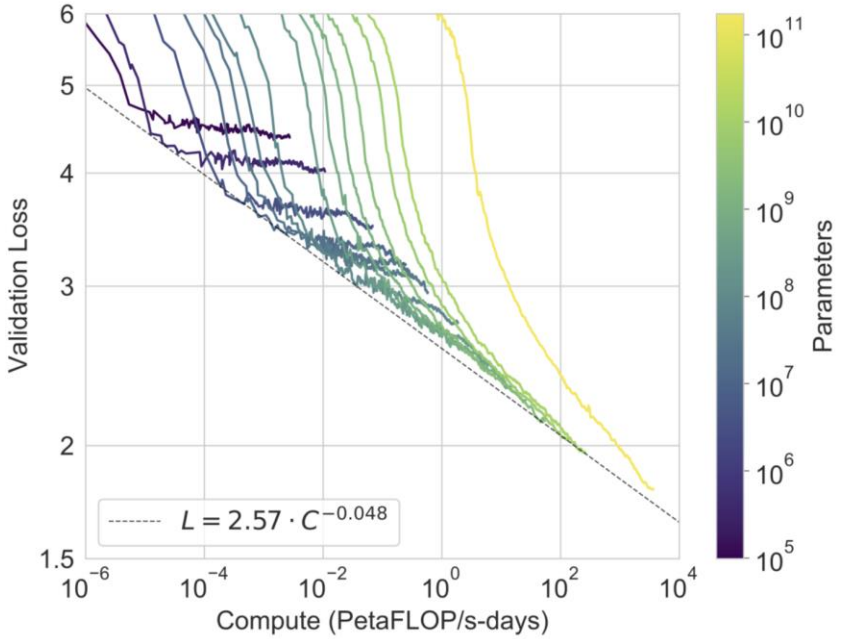








Conclusion:
Bigger model requires more data
More data requires more efficient pipelines
Lightning requires more data for code writing but much more versatile



Contact information

Thank you for your attention