Runtime Performance Analysis on Multiple Knowledge Graph Embedding Methods

Gustavo Ribeiro Kremer

1. Introduction

What are Knowledge Graphs

"We define a knowledge graph as a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities." [1]

- Nodes are entities
- Edges are relations between entities

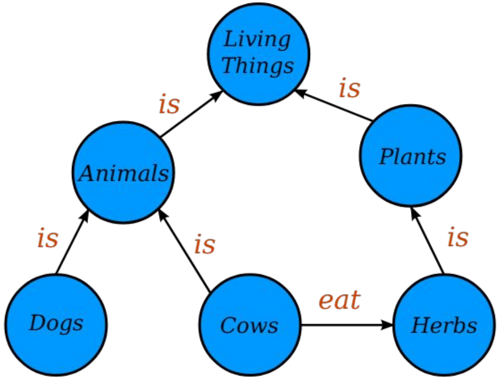


Image source: Wikipedia.

Ok, and what are Knowledge Graphs Embeddings?

Knowledge Graph Embeddings

Embeddings are dense representations of a KG in a continuous, low-dimensional vector space [2]. The KG is transformed in a lower dimensional space, while keeping their semantic meaning [3].

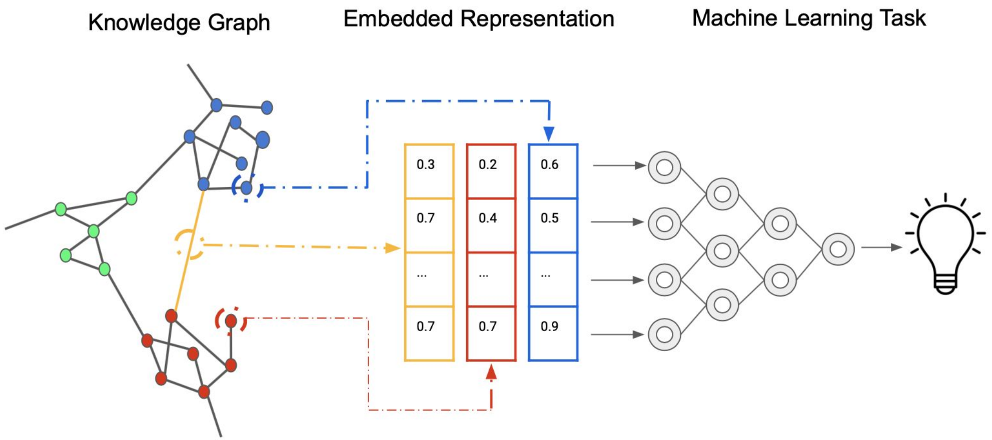


Image source: Wikipedia.

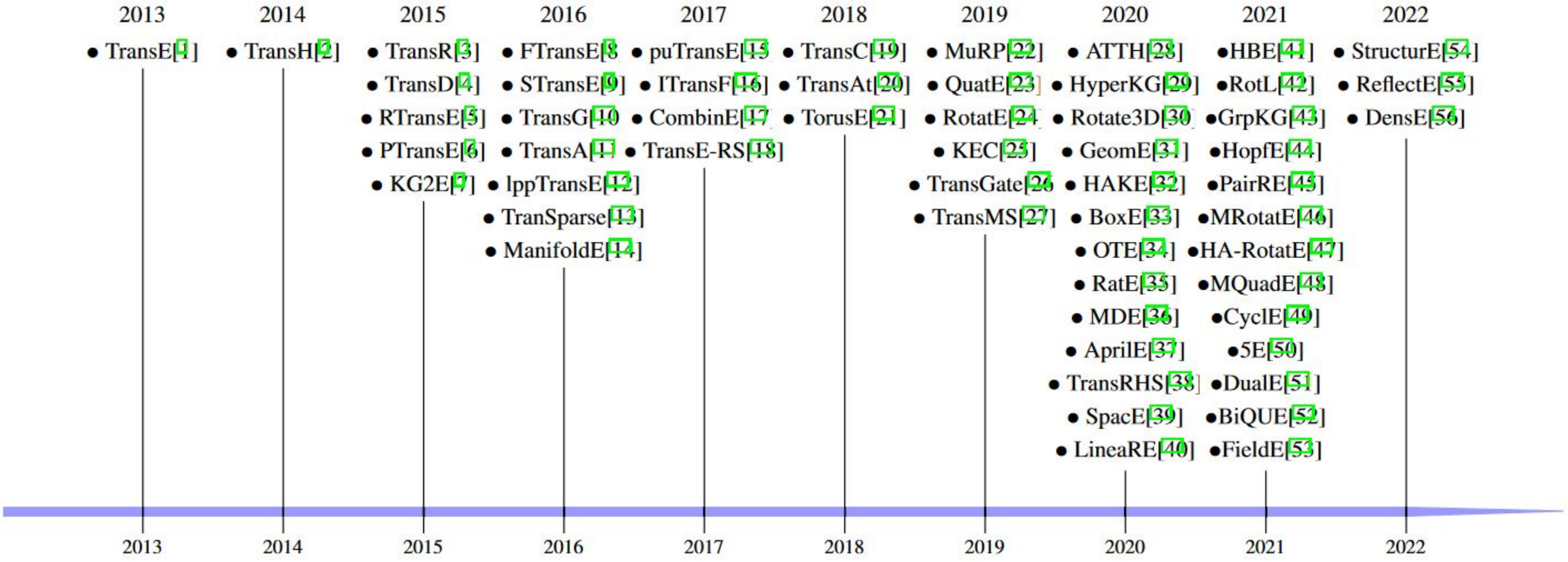
Why?

- Compact and efficient: Lower memory use and faster vector operations than direct graph operations [2].
- ML-friendly: Vector spaces integrate easily with ML and NLP models [2].
- Enables downstream tasks like: Link prediction, multi-hop reasoning, KG alignment, entity classification.
- Can even find missing links: Predict likely but unrecorded relations in KGs [2].

2. Theoretical Background

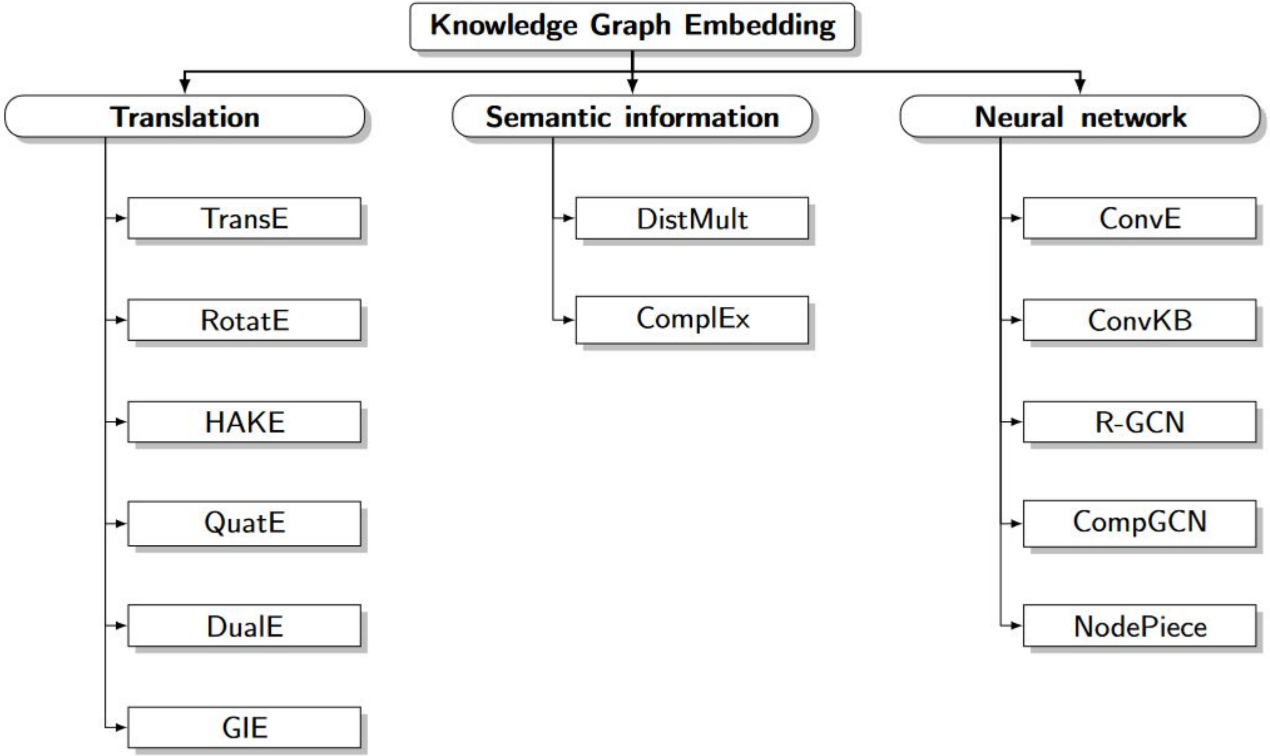
Some techniques

[2]



Types of Models

[4]



Evaluations Metrics

- MRR: Average of the reciprocal ranks of the correct entities across queries. Values ∈ (0,1]; closer to 1 means the model ranks the correct answer near the top.

$$MRR = (1 / N) * \Sigma (1 / rank\_i) \text{ for } i = 1..N.$$

- Hits@K: Fraction of queries where the correct entity appears in the top-K predictions. Typical K values: 1, 3, 10 (report multiple). For example, Hits@10 = 0.72 means 72% of correct answers are inside top 10.

$$Hits@K = (1 / N) * \Sigma\ I(rank\_i \le K), \text{ where } I() \text{ is the indicator function.}$$

3. Proposal

Proposal

Measure training time and inference time of major KGE models, comparing runtime performance across multiple benchmark datasets.

- Training time: report total wall-clock to train with fixed epochs over multiple runs.
- Inference time: report time to perform inference with different inputs, measuring pure forward-pass time.

Metrics to present: total training time, inference latency.

4. Experimental Setup

Frameworks and Tools (subject to modification)

- Python3 as the main language
- Conda for environment management
- Jupyter Notebooks for documenting experiments
- Git + Github for version control management
- PyKEEN for reproducible, facile knowledge graph embeddings
- Seaborn + matplotlib for graphs and visualizations

Models

Name	Model	Citation	Type
ComplEx	`pykeen.models.ComplEx`	Trouillon *et al.*, 2016	Semantic Information
DistMult	`pykeen.models.DistMult`	Yang *et al.*, 2014	Semantic Information
TransE	`pykeen.models.TransE`	Bordes *et al.*, 2013	Translation
RotatE	`pykeen.models.RotatE`	Sun *et al.*, 2019	Translation
ConvE	`pykeen.models.ConvE`	Dettmers *et al.*, 2018	Neural Network
ConvKB	`pykeen.models.ConvKB`	Nguyen *et al.*, 2018	Neural Network
R-GCN	`pykeen.models.RGCN`	Schlichtkrull *et al.*, 2018	Neural Network

Datasets

Name	Documentation	Citation	Entities	Relations	Triplets
Nations	`pykeen.datasets.Nations`	ZhenfengLei/KGDatasets	14	55	1992
DBpedia50	`pykeen.datasets.DBpedia50`	Shi *et al.*, 2017	24624	351	34421
FB15k-237	`pykeen.datasets.FB15k237`	Toutanova *et al.*, 2015	14505	237	310079
WordNet-18	`pykeen.datasets.WN18`	Bordes *et al.*, 2014	40943	18	151442
YAGO3-10	`pykeen.datasets.YAGO310`	Mahdisoltani *et al.*, 2015	123143	37	1089000

5. Planning

Chronogram

Week	Tasks
1, 2, 3	Set up experimental environment (hardware/software, frameworks, scripts).
4	Implement training/inference time measurement (logging, benchmarking functions).
5	Run pilot experiments on 1–2 models to validate measurement pipeline.
6	Full experiments — for all selected models and datasets.
7	Full experiments — for all selected models and datasets.
8	Aggregate results, compute statistics (mean, std), generate preliminary tables/plots.
9	Write analysis and discussion: compare models, highlight trade-offs.
10	Finalize report.

References

[1] Hogan, A., Blomqvist, E., Cochez, M., D’amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A.-C. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2022). Knowledge Graphs. ACM Computing Surveys, 54(4), 1–37. https://doi.org/10.1145/3447772

[2] Ge, X., Wang, Y.-C., Wang, B., & Kuo, C.-C. J. (2023). Knowledge Graph Embedding: An Overview (arXiv:2309.12501). arXiv. https://doi.org/10.48550/arXiv.2309.12501

[3] Ji, S., Pan, S., Cambria, E., Marttinen, P., & Yu, P. S. (2022). A Survey on Knowledge Graphs: Representation, Acquisition and Applications. IEEE Transactions on Neural Networks and Learning Systems, 33(2), 494–514. https://doi.org/10.1109/TNNLS.2021.3070843

[4] Ferrari, I., Frisoni, G., Italiani, P., Moro, G., & Sartori, C. (2022). Comprehensive Analysis of Knowledge Graph Embedding Techniques Benchmarked on Link Prediction. Electronics, 11(23), 3866. https://doi.org/10.3390/electronics11233866

Part 2

What are we measuring?

- Training time (s): Time to train with fixed epochs over multiple runs.
- Evaluation time (s): Time taken to obtain main metrics over the trained model
- Inference time (s): Time to infer with different inputs, measuring pure forward-pass time.
- Power Consumption (w): Total power used by GPU in the training and evaluation.
- GPU usage (%): GPU usage metrics

Training vs Evaluation Time

[4]

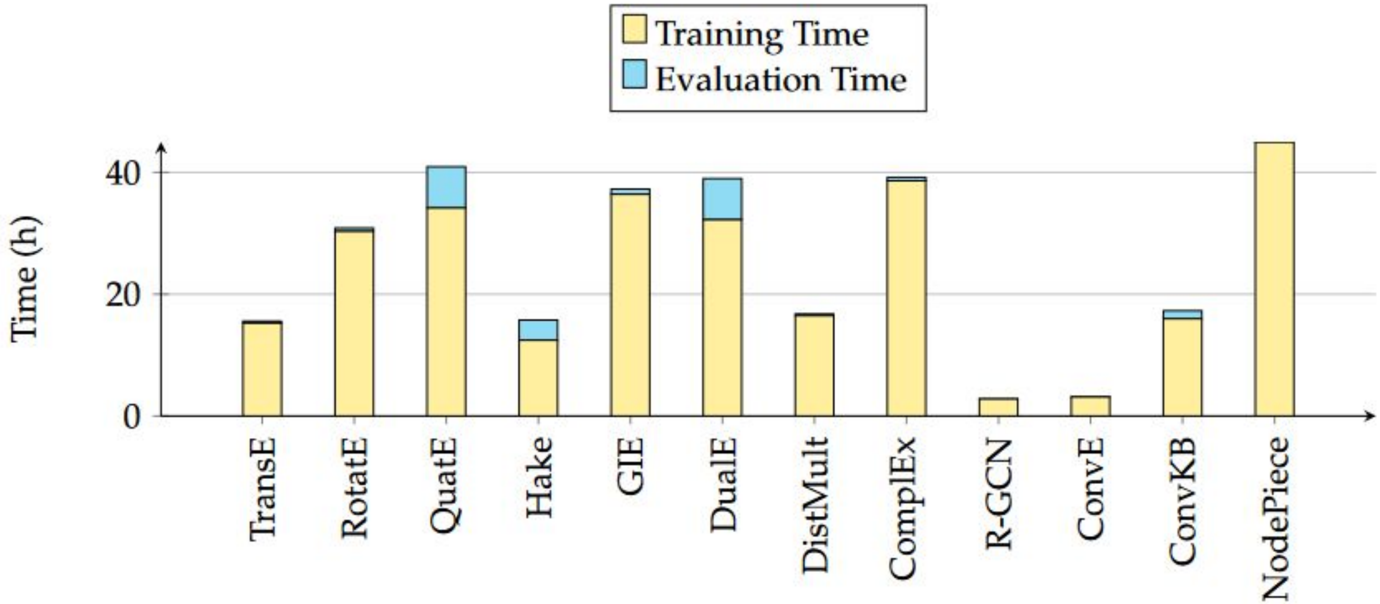


Figure 9. Training and evaluation times (stacked) for each model on OGB-BioKG.

Where are we measuring?

CPU: Intel Core I7 10700 16Mb cache

16 core (8 physical)

2.9GHz (up to 4.8GHz)

Ram: 32Gb DDR4 2933 mt/s

Disk: 512gb NVMe

GPU: RTX 3060 12gb

3584 Cuda Cores

Clock Boost: 1320 MHz (up to 1777 MHz)

How are we measuring?

- Training time (s): PyKEEN default library
- Evaluation time (s): PyKEEN framework
- Inference time (s): Python time framework
- Power Consumption (w): pyNVML (wrapper around the NVML library)
- GPU usage (%): pyNVML

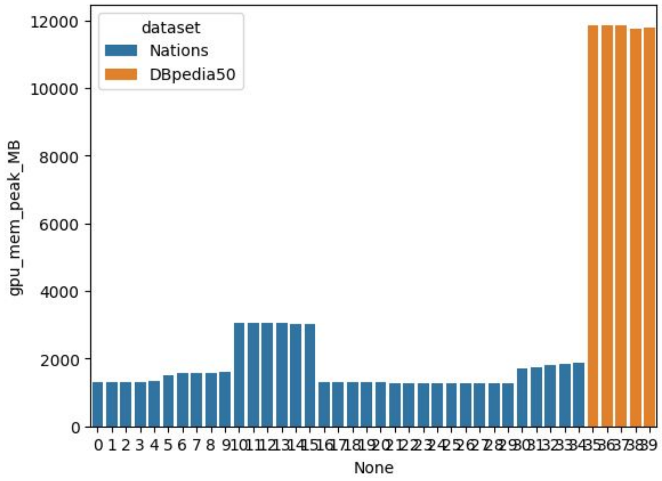
Experimental Setup

Algorithms	["ConvE","DistMult","TransE","RotatE","ComplEx","ConvKB","R-GCN"]
Datasets	[“Nations”, “DBpedia50”]
Epochs	50
Batch Size	1024
Replicata	5
Inference Batch Size	1

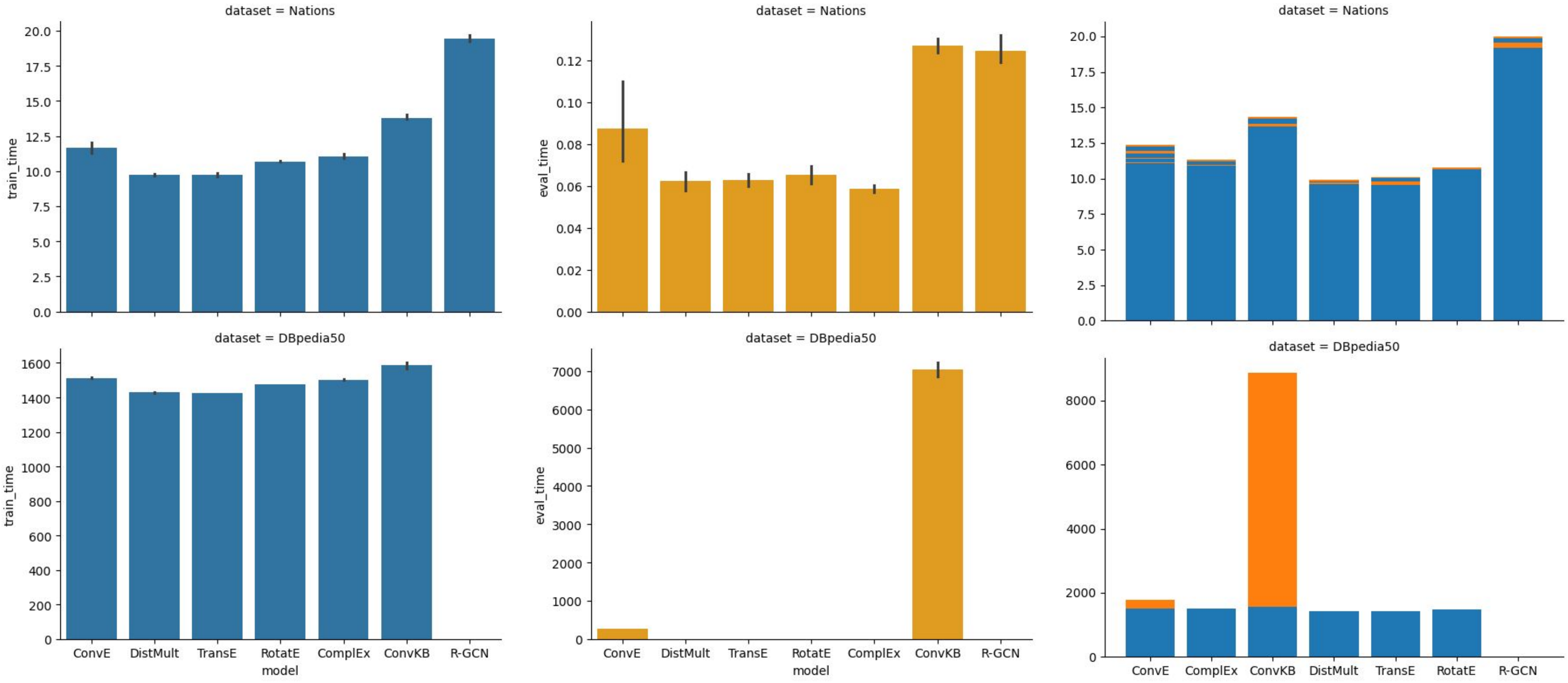
	model	dataset	seed	epochs	train_time	eval_time	inference_time	mrr	hits@1	hits@3	hits@5	hits@10	gpu_mem_avg_MB	gpu_mem_peak_MB
0	ConvE	Nations	1	50	11.742623	0.130367	0.000941	0.550424	0.360697	0.664179	0.808458	0.982587	1362.321849	1623.601562
1	ConvE	Nations	2	50	11.373197	0.066880	0.000994	0.584580	0.402985	0.696517	0.848259	0.972637	1351.931751	1368.039062
2	ConvE	Nations	4	50	11.058975	0.079638	0.000879	0.563294	0.373134	0.694030	0.835821	0.987562	1352.982572	1368.289062
3	ConvE	Nations	7	50	11.846289	0.079801	0.000917	0.579798	0.395522	0.699005	0.838308	0.975124	1351.982459	1366.351562
4	ConvE	Nations	11	50	12.271737	0.079694	0.000955	0.559576	0.378109	0.664179	0.820896	0.990050	1354.726562	1368.164062
5	ComplEx	Nations	1	50	10.881930	0.060670	0.000446	0.387280	0.169154	0.460199	0.674129	0.942786	910.161704	916.789062
6	ComplEx	Nations	2	50	11.238427	0.060916	0.000545	0.380474	0.166667	0.450249	0.669154	0.947761	909.803486	916.789062
7	ComplEx	Nations	4	50	11.054577	0.056204	0.000418	0.409685	0.201493	0.492537	0.674129	0.942786	909.304688	917.976562
8	ComplEx	Nations	7	50	11.271754	0.057398	0.000434	0.366122	0.146766	0.417910	0.674129	0.957711	912.185697	918.101562
9	ComplEx	Nations	11	50	10.911720	0.058351	0.000399	0.394540	0.179104	0.485075	0.676617	0.965174	913.930889	924.289062
10	ConvKB	Nations	1	50	14.208464	0.127992	0.000516	0.602457	0.427861	0.738806	0.853234	0.985075	2433.497789	2618.101562
11	ConvKB	Nations	2	50	13.688159	0.122240	0.000611	0.585398	0.400498	0.711443	0.850746	0.995025	2439.734375	2618.101562
12	ConvKB	Nations	4	50	13.737572	0.130478	0.000517	0.604000	0.430348	0.701493	0.855721	0.987562	2439.008413	2618.039062
13	ConvKB	Nations	7	50	13.723423	0.124082	0.000545	0.600457	0.422886	0.718905	0.870647	0.987562	2436.804688	2616.164062
14	ConvKB	Nations	11	50	13.665044	0.130958	0.000509	0.584981	0.405473	0.703980	0.848259	0.987562	2438.020433	2616.164062
15	DistMult	Nations	1	50	9.587991	0.064867	0.000369	0.444842	0.228856	0.554726	0.773632	0.970149	882.105168	886.164062
16	DistMult	Nations	2	50	9.694942	0.065667	0.000348	0.460729	0.253731	0.562189	0.721393	0.947761	881.754207	886.164062
17	DistMult	Nations	4	50	9.777468	0.067981	0.000362	0.475342	0.286070	0.567164	0.723881	0.962687	881.920072	886.164062
18	DistMult	Nations	7	50	9.861570	0.057729	0.000337	0.412890	0.186567	0.512438	0.791045	0.977612	881.754207	886.164062
19	DistMult	Nations	11	50	9.779886	0.055175	0.000323	0.445091	0.231343	0.537313	0.748756	0.967662	881.483774	886.164062
20	TransE	Nations	1	50	9.580193	0.064155	0.000448	0.296620	0.000000	0.467662	0.713930	0.955224	879.768630	884.164062
21	TransE	Nations	2	50	10.049441	0.063962	0.000436	0.302763	0.000000	0.487562	0.713930	0.952736	879.634014	884.164062
22	TransE	Nations	4	50	9.753514	0.060421	0.000532	0.300569	0.000000	0.492537	0.721393	0.967662	879.601562	884.164062
23	TransE	Nations	7	50	9.556728	0.068186	0.000416	0.299261	0.000000	0.475124	0.701493	0.965174	879.459736	884.164062
24	TransE	Nations	11	50	9.689356	0.057434	0.000476	0.302759	0.000000	0.475124	0.713930	0.965174	879.636418	884.164062

Main Challenges

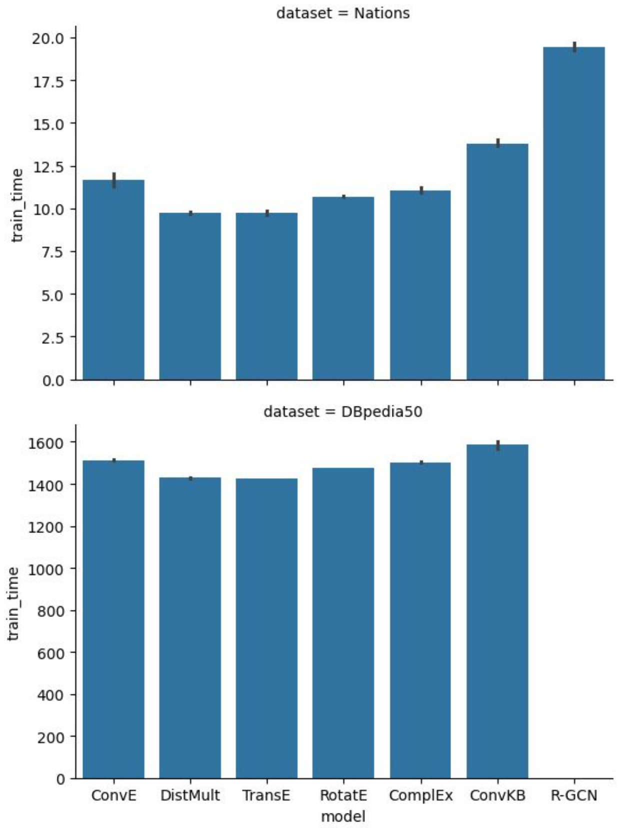
1. Training and evaluation time longer than expected
2. GPU Ram
3. Models taking up ram space

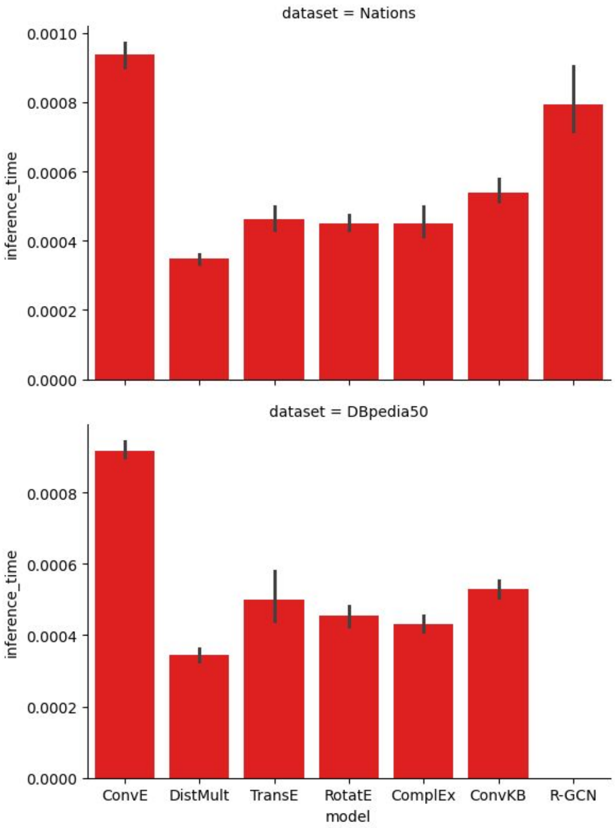


Training and Evaluation

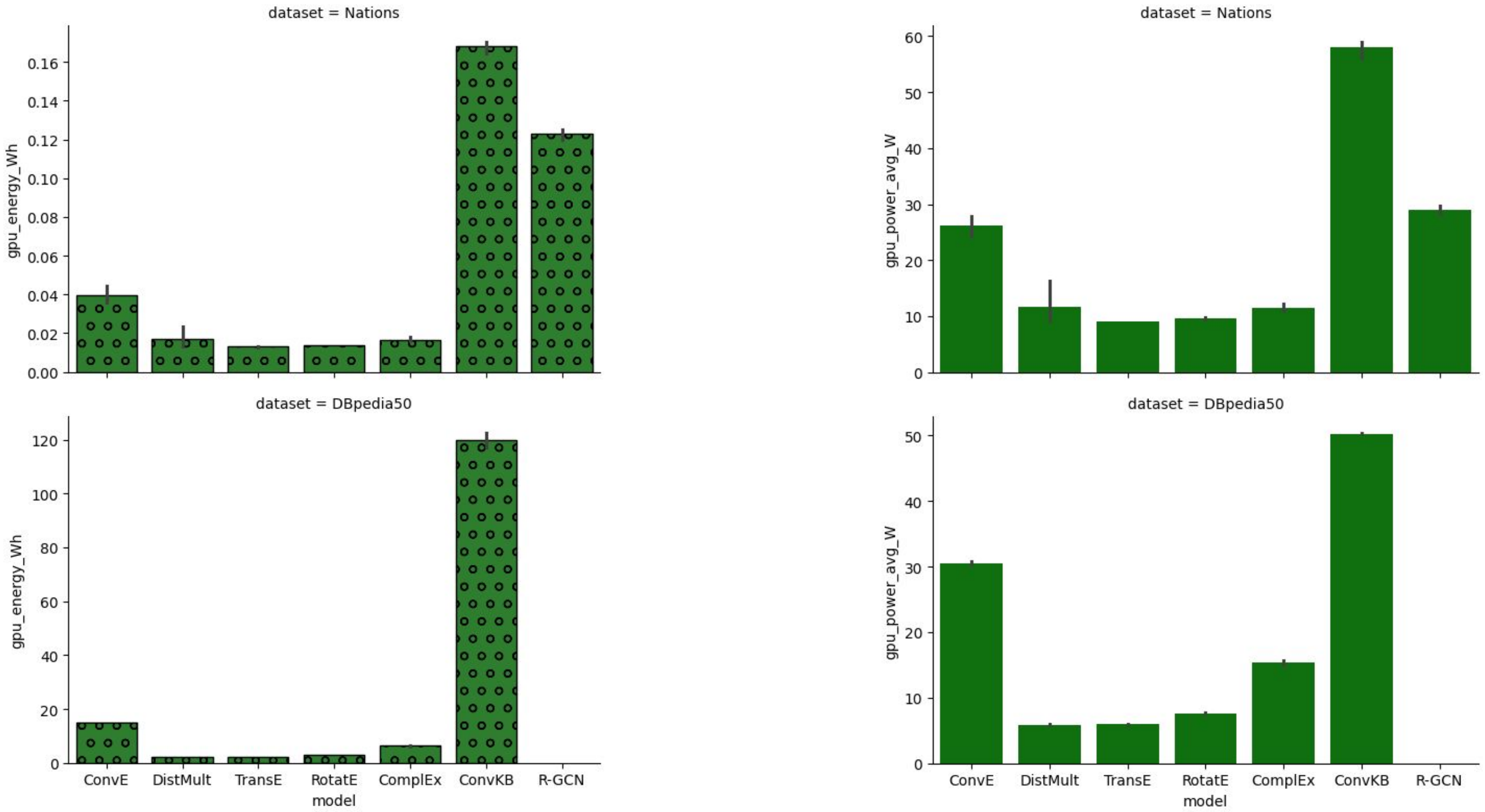


Training and Inference

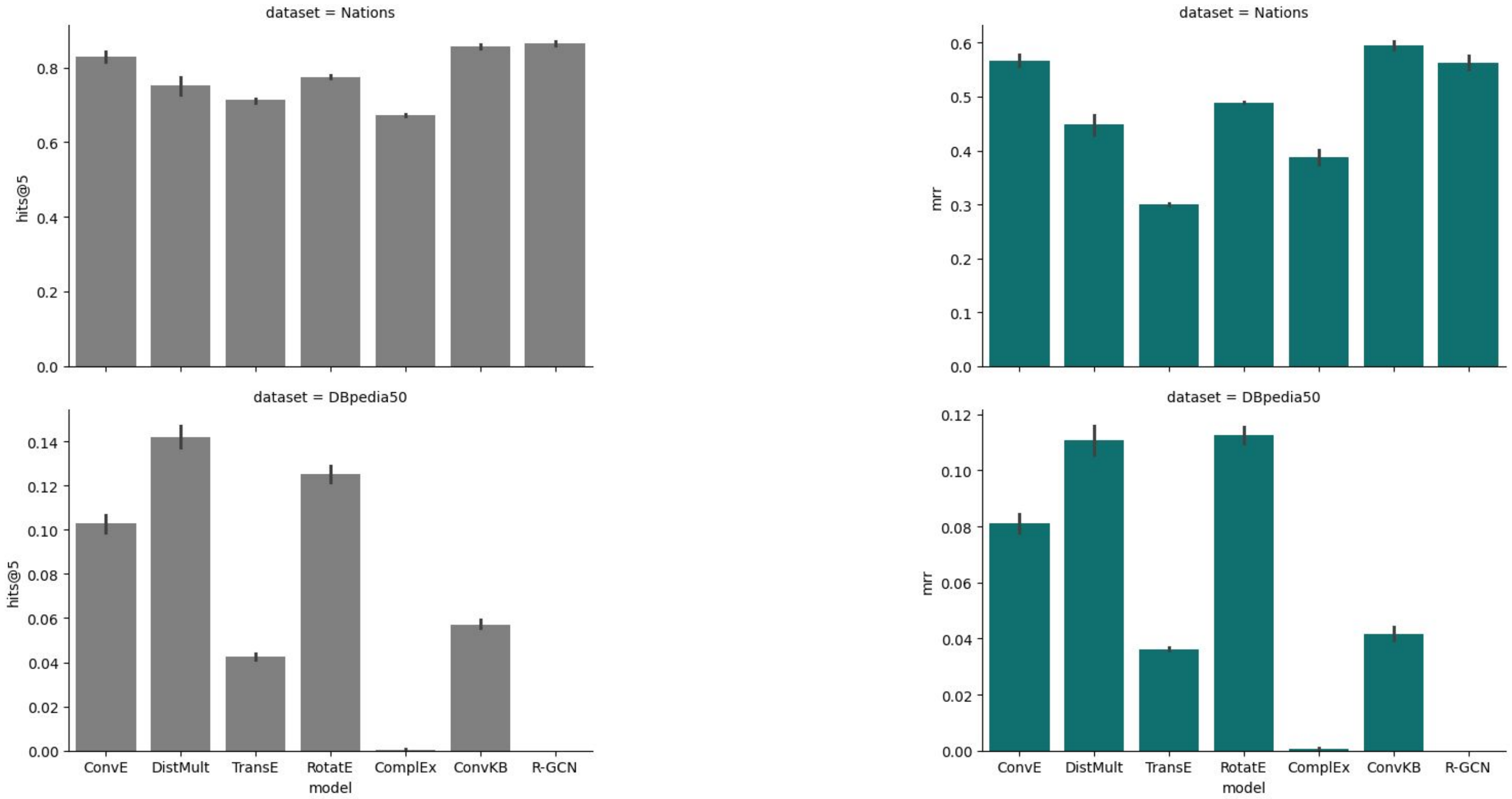




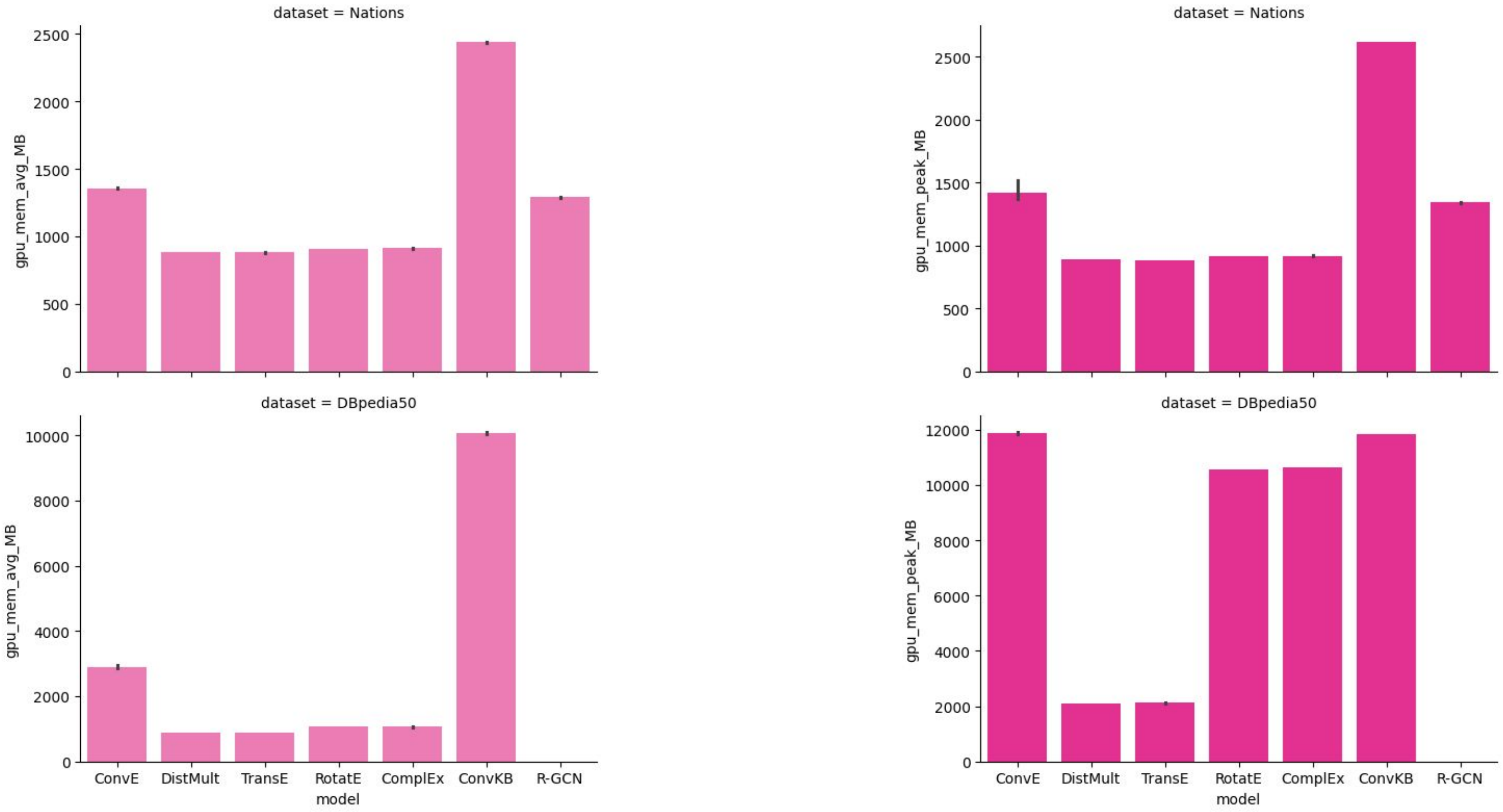
Energy Consumption



Model Performance



Memory Utilization



Next Steps

- Further investigating on ram memory consumption
- Benchmarking the next datasets
- Ranking models based on cost-benefit