Your Knowledge Graph Embeddings are Secretly Circuits and You Should Treat Them as Such

Lorenzo Loconte[1] **Nicola Di Mauro**[1] **Robert Peharz**[2] **Antonio Vergari**[3]

[1]Department of Computer Science, University of Bari, Italy
[2]Institute of Theoretical Computer Science, TU Graz, Austria
[3]School of Informatics, University of Edinburgh, UK

Abstract

Some of the most popular and successful knowledge graph embedding (KGE) models – CP, ComplEx, Rescal and TuckER– encode tensor factorizations that define an energy-based score over subject-relation-object triples. As such, they are not amenable to efficient maximum-likelihood training, and do not easily allow to sample triples nor answering complex queries in a principled probabilistic way. In this paper, we show how all these models can be readily interpreted as constrained computational graphs—*circuits*—and show how, by some minor modifications, one can turn them into tractable generative models of triples. This novel perspective not only fixes many of the aforementioned shortcomings of KGE models, but helps understand why recent learning strategies for KGE are successful while suggesting interesting new ones.

1 FROM KNOWLEDGE GRAPH EMBEDDINGS…

A knowledge graph (KG) $\mathcal{G}$ is a graph-structured knowledge base encoding relationships between entities as triples of the form (s, p, o) where s and o denote the *subject* and *object* entities and p the *predicate*, or relation type, labeling the relationship between the two. More formally, let $\mathcal{E}$ be the set of all entities and $\mathcal{R}$ be the set of all relation types. Then, $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E} = \{(s_i, p_i, o_i)\}_{i=1}^{T}$.

The simplest reasoning query over KGs is to predict missing triples, a task also called *link prediction* [Nickel et al., 2016]. Knowledge graph embedding (KGE) models achieve the current state-of-the-art models for link prediction on KGs [Lacroix et al., 2018, Ruffinelli et al., 2020, Chen et al., 2021]. A KGE model associates a continuous

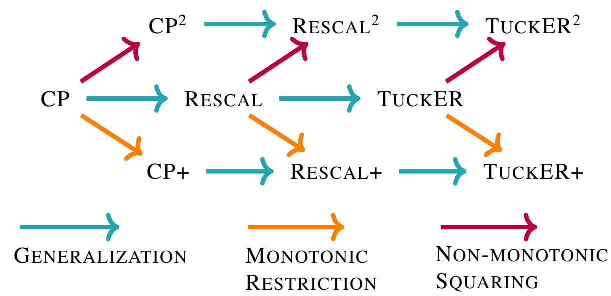

Figure 1: We turn existing KGE models based on tensor factorizations into tractable generative models of KG triples by interpreting their score functions as circuits and restricting their parameters (Sec. 3.1) or squaring them (Sec. 3.2).

vector representation to entities in $\mathcal{E}$ and relation types in $\mathcal{R}$. These embeddings are then used to compute a *score function* $\phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ that outputs the unnormalized log-probability of observing the triple (s, p, o), i.e., $\phi(s, p, o) \propto \log \Pr(s, p, o)$. As such, they are an instance of energy-based models [LeCun et al., 2006].

We denote with $\mathcal{X} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{R}|\times|\mathcal{E}|}$ the three-order tensor of confidence scores for each triple, i.e., $x_{ijk} = \phi(s_i, p_j, o_k)$. In this work we focus on KGE models such as DistMult [Yang et al., 2015], CP [Lacroix et al., 2018], ComplEx [Trouillon et al., 2016], Rescal [Bordes et al., 2013] and TuckER [Balazevic et al., 2019], that define a score function that represents a specific factorization for $\mathcal{X}$. E.g., CP defines the factorization as the trilinear product

$$\phi_{\text{CP}}(s, p, o) = \langle \mathbf{e}_s, \mathbf{w}_p, \mathbf{e}_o \rangle \tag{1}$$

over $\mathbf{e}_s, \mathbf{w}_p, \mathbf{e}_o \in \mathbb{R}^R$, the R-dimensional embedding vectors associated to the subject, relation type and object. DistMult defines the same score function of CP, but does not differentiate between subject and object roles of entities. ComplEx, yielding state-of-the-art performance on several link prediction benchmarks [Ruffinelli et al., 2020, Chen et al., 2021], defines a score $\phi_{\text{ComplEx}}(s, p, o)$ from

Accepted for the 5th Workshop on Tractable Probabilistic Modeling at UAI (TPM 2022).

the application of DistMult over complex embeddings:

$$\langle \mathrm{Re}(\mathbf{e}_s), \mathrm{Re}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o)\rangle + \langle \mathrm{Im}(\mathbf{e}_s), \mathrm{Re}(\mathbf{w}_p), \mathrm{Im}(\mathbf{e}_o)\rangle$$
$$+ \langle \mathrm{Re}(\mathbf{e}_s), \mathrm{Im}(\mathbf{w}_p), \mathrm{Im}(\mathbf{e}_o)\rangle - \langle \mathrm{Im}(\mathbf{e}_s), \mathrm{Im}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o)\rangle$$

where Re and Im define the real and imaginary part of the complex embeddings $\mathbf{e}_s, \mathbf{w}_p, \mathbf{e}_o \in \mathbb{C}^R$. Instead, TuckER and Rescal generalize CP and DistMult as shown in Fig. 1. Fig. 2 illustrates their scoring functions.

To obtain normalized probabilities and exact gradients, one would need to compute the *partition function*, $\mathcal{Z} = \sum_{s\in\mathcal{E}}\sum_{p\in\mathcal{R}}\sum_{o\in\mathcal{E}} \exp\phi(s,p,o)$. As it would require a summation over $|\mathcal{E}\times\mathcal{R}\times\mathcal{E}|$ terms, this can be infeasible for real-world KGs. For instance, for Freebase [Nickel et al., 2016] it would require 10^{19} evaluations of ϕ and in the order of 10^{11} for the much smaller WN18RR and FB15k-237 KGs [Dettmers et al., 2018, Toutanova and Chen, 2015]. Therefore, several training strategies for KGEs models have been devised, involving heuristics and losses that circumvent the computation of $\mathcal{Z}$. E.g., the 1vsALL objective is a *discriminative* objective [Ruffinelli et al., 2020] that does so by computing log-conditional probabilities:

$$\mathcal{L}_{\mathsf{1vsALL}} := \sum_{(s,p,o)\in\mathcal{G}} \log \Pr(s \mid p, o) + \log \Pr(o \mid s, p). \quad (2)$$

While these losses and heuristics can deliver good performance in link prediction tasks [Ruffinelli et al., 2020, Chen et al., 2021], other probabilistic reasoning scenarios are still out of their reach. For example, even sampling from energy-based models is inherently hard [Song and Kingma, 2021] despite some recent heuristics for KGE models [Chauhan et al., 2021]. Answering more complex queries such as *union of conjunctive queries* (UCQ) [Dalvi and Suciu, 2007] exactly and efficiently would require a principled *generative* model [Friedman and Van den Broeck, 2020].

In this paper, we investigate when and how we can devise a generative KGE model for triples that is expressive as a discriminative one and furthermore allows to: **1)** obtain normalized and calibrated probabilities thus facilitating the comparison of triples; **2)** efficiently and exactly marginalize, thus enabling the computation of exact gradients and maximum-likelihood learning; **3)** efficiently sample new triples; **4)** exactly answer UCQs. We do so by first noting how some of the most popular KGE models, whose scores are based on tensor factorizations [Kolda and Bader, 2009], can be naturally interpreted as constrained computational graphs, also known as *circuits* [Vergari et al., 2021]. Then, we devise under which assumptions these circuits can be cast as probabilistic circuits [Choi et al., 2020] and discuss how this enables properties **1-4** listed above.

2 ... TO CIRCUITS...

We start by showing in Thm. 1 that the score functions of tensor-factorization KGE models can be readily represented as parameterized computational graphs with certain structural properties, called *circuits* [Vergari et al., 2021, Choi et al., 2020]. The next definitions introduce the properties of circuits that are relevant for our purposes.

Definition 1 (Circuit)**.** A Circuit $\mathcal{C}$ over variables $\mathbf{X}$ is a parametrized directed acyclic computational graph encoding a function $\mathcal{C}(\mathbf{X})$ and comprising three kinds of computational units: *input functionals*, *product*, and *sum* units. An input functional n represents a base parametric function $\mathcal{C}_n(\delta(n);\ \boldsymbol{\lambda})$ over some variables $\delta(n) \subseteq \mathbf{X}$, called its scope, and it is parameterized by $\boldsymbol{\lambda}$. Sum and product units n elaborate the output of other units, denoted $\mathsf{in}(n)$. Sum units are parameterized by $\boldsymbol{\omega}$ and compute the weighted sum of their inputs $\sum_{i\in\mathsf{in}(n)} \omega_i \mathcal{C}_i(\delta(n))$, while product units compute $\prod_{i\in\mathsf{in}(n)} \mathcal{C}_i(\delta(n))$. The scope of an inner unit (i.e., product or sum) is the union of the scopes of its inputs. The output of the circuit is given by the last unit in the computational graph.

Exact computation of the partition function, and any other marginals, can be done in a *smooth* and *decomposable* circuit in a single graph evaluation (Thm. 3).

Definition 2 (Smoothness and Decomposability)**.** A circuit is *smooth* if for every sum unit, its input units depend all on the same variables. A circuit is *decomposable* if the inputs of every product unit depend on disjoint sets of variables.

Furthermore, *structured-decomposable* circuits can support the exact computations of natural powers, which will be useful in the next section.

Definition 3 (Structured-Decomposability)**.** A decomposable circuit is *structured-decomposable* if all the product units sharing the same scope decompose in the same way.

Theorem 1 (KGE models as Circuits)**.** The score functions of CP, DistMult, ComplEx, Rescal and TuckER can be represented as smooth and structured-decomposable circuits over variables $\mathbf{X} = \{S, P, O\}$ denoting respectively the subject, the relation type and the object, without additional memory requirements.

Proof. We report the complete proof by construction in App. A. In a nutshell, it suffices to i) transform the tensor multiplications into corresponding sum and product units and ii) create an input functional for each i-th embedding component, $i = 1, \ldots, R$, as to implement a lookup function that computes the corresponding value for an entity or predicate, e.g., e_{si} for the subject embedding in CP. Fig. 2 shows this construction for the scoring functions of CP/DistMult, Rescal and TuckER. □

This construction paves new ways to build KGE models by leveraging the literature on how to build circuit structures [Vergari et al., 2020]. More crucially, it helps us devise tractable generative KGE models.

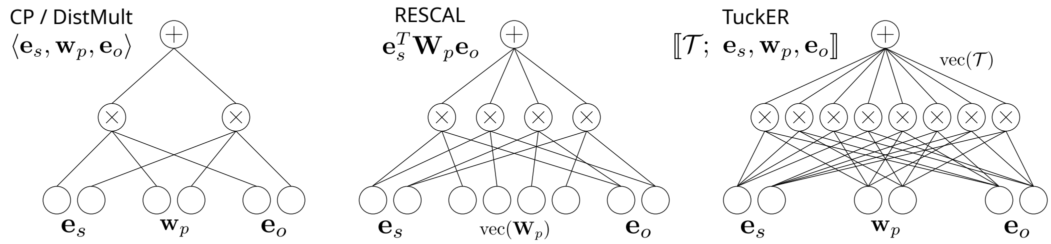

Figure 2: CP, DISTMULT, RESCAL and TUCKER scoring functions over rank-2 embeddings represented as circuits. All sum unit parameters are assumed to be 1 except for TUCKER where they are the vectorization of the core tensor $\mathcal{T}$.

3 ... TO PROBABILISTIC CIRCUITS

Under the light of Thm. 1, the score functions of KGE models *are* circuits that *could* support efficient marginalization, *but* in log-space and not in probability space. As stated in Sec. 1, current KGE models retrieve a probabilistic output by applying a logistic function to the output of their circuits and this clearly hinders marginalization [Vergari et al., 2021, Van den Broeck et al., 2021]. To retrieve both a probabilistic semantic and efficient marginalization in one pass we are looking at smooth and decomposable circuits that encode functions that output positive values, i.e., probabilistic circuits (App. B).

Definition 4 (Tractable probabilistic KGE Circuit)**.** A tractable probabilistic circuit (PC) for KGE models is a smooth and decomposable circuit $\mathcal{C}$ that encodes a triple score function $\phi_{\mathcal{C}}(s,p,o) \propto \mathsf{Pr}(s,p,o)$, i.e., it outputs $\phi_{\mathcal{C}}(s,p,o) \geq 0$ for all triples (s,p,o).

In Def. 4 the score function $\phi_{\mathcal{C}}$ always outputs non-negative values in contrast with the score functions of the KGE models previously cited. To build such a PC, we propose two strategies: restricting its parameter space and squaring it, as reported in the following.

3.1 MONOTONIC RESTRICTION

A sufficient condition for obtaining a PC as in Def. 4 is to restrict the circuit to be *monotonic*, i.e., to allow only for non-negative parameters [de Colnet and Mengel, 2021]. By contrast, circuits encoding KGE score functions, as discussed in Thm. 1 and shown in Fig. 2, are a form of *non-monotonic circuits*—i.e., they contain negative parameters. While sum unit parameters are unitary (hence positive) in CP and RESCAL, their embeddings can take any real values. We denote as CP+, DISTMULT+, RESCAL+ and TUCKER+ the monotonic restrictions of the corresponding KGE models. In these monotonic PCs, we can now interpret the input functionals associated to embedding entries as (unnormalized) *categorical random variables* that can take values in $\mathcal{E}$ or $\mathcal{R}$ if they refer to entities or relations.

However, we cannot simply restrict parameters to be non-negative in COMPLEX to obtain a PC, as its score function explicitly contains a subtraction (Sec. 1). To circumvent this issue, we impose an additional constraint that enforces that the real part of each embedding entry is always greater or equal than the corresponding imaginary part. We discuss this in detail in App. C.

3.2 SQUARING NON-MONOTONIC CIRCUITS

Restricting PCs to have non-negative parameters can be a too strong limitation impacting its expressiveness [Valiant, 1979]. For this reason, we propose to obtain a PC by *squaring* the non-monotonic circuits encoding a KGE score function, i.e., $\phi^2(s,p,o) = \phi(s,p,o) * \phi(s,p,o)$. This will ensure the non-negativity of the score, while allowing parameters to be negative. For example for CP, its squared version CP^2 will encode

$$\phi_{\text{CP}^2}(s,p,o) = \phi^2_{\text{CP}}(s,p,o) = \langle \mathbf{e}_s, \mathbf{w}_p, \mathbf{e}_o \rangle^2 \qquad (3)$$

Since the score functions of CP, COMPLEX, RESCAL and TUCKER can be readily represented as structured-decomposable circuits (Thm. 1), their squared versions can be compactly represented as smooth and decomposable circuits [Vergari et al., 2021].

Theorem 2 (Tractable squaring of KGE Circuits)**.** The marginalization of the score functions of CP^2, DISTMULT^2, COMPLEX^2, RESCAL^2, and TUCKER^2 can be computed in time linear to $|\mathcal{E}|$ and $|\mathcal{R}|$ and quadratic in the size of the original circuits.

Proof. The proof directly comes from the fact that squaring a smooth, decomposable and structured-decomposable KGE circuit $\mathcal{C}$ can be done in time $\mathcal{O}(|\mathcal{C}|^2)$ [Vergari et al., 2021]. Since the resulting probabilistic circuit is smooth and decomposable, marginalization can be performed in time $\mathcal{O}(|\mathcal{E}| \cdot |\mathcal{C}|^2 + |\mathcal{R}| \cdot |\mathcal{C}|^2)$. In App. D we show time and space complexity results regarding the computation of the partition function of squared KGE circuits. $\square$

In the next sections, we discuss how the efficient and sound probabilistic interpretation derived by our monotonic restriction and squaring strategies can enable a number of strategies to train and perform inference on KGE models that were not possible before.

4 THE PERKS OF BEING A TRACTABLE GENERATIVE MODEL

Learning. Our probabilistic KGE circuits can be efficiently trained by directly maximising the log-likelihood

$$\mathcal{L}_{\mathsf{MLE}} = \sum\nolimits_{(s,p,o)\in\mathcal{G}} \log \phi_{\mathcal{C}}(s,p,o) - \log \mathcal{Z}_{\mathcal{C}}; \quad (4)$$

as we can exactly compute the partition function $\mathcal{Z}_{\mathcal{C}} = \sum_{s'\in\mathcal{E}}\sum_{p'\in\mathcal{R}}\sum_{o'\in\mathcal{E}}\phi_{\mathcal{C}}(s',p',o')$ in a single pass. This enables us to also speed-up the computation of the discriminative 1vsALL objective (Eq. (2)) while marginalizing over subjects s and objects o. By applying this idea to other pseudo-likelihood like objectives [Chen et al., 2021] and composing them we can train our models by novel *composite (log-)likelihood objectives* [Varin et al., 2011], e.g., by optimizing $\mathcal{L}_{\mathsf{MLE}} + \mathcal{L}_{\mathsf{1vsALL}}$ which retrieves a generative-discriminative objective and $\mathcal{L}_{\mathsf{MLE}}$ can be interpreted as a regularizer [Peharz et al., 2019, 2020].

Sampling. While sampling triples from KGE models is generally intractable, one can sample from KGE circuits obtained through monotonic restriction (Sec. 3.1) easily via ancestral sampling [Vergari et al., 2019]. For non-monotonic squared circuits (Sec. 3.2) we can use *inverse transform sampling*, since they support tractable conditioning and the computation of the cumulative distribution function (CDF). This can be done in an autoregressive fashion: one can sample every variable by mapping some uniform noise through the inverse CDF conditioned on some variable ordering [Novikov et al., 2021].

Complex query answering. Answering queries such as UCQs on KGs has been addressed via several *heuristics* such as training additional neural network classifiers or using continuous relaxations of logic operators [Hamilton et al., 2018, Ren et al., 2020, Arakelyan et al., 2021].

To answer all UCQs *exactly*, instead, we follow the assumption of Friedman and Van den Broeck [2020] and factorize each relation $R(x, y)$ appearing in a UCQ Q into a conjunction of unary atoms $E(x) \wedge T(R) \wedge E(y)$. This implies we now need a probability distribution defined over a larger set of random variables: $\mathcal{E}\cup\mathcal{R}$, i.e., one for each entity and one for each relation type. This is different from considering only three random variables S, P, O as we assumed so far. In order to fill this conceptual gap, we can view any of the proposed KGE circuit $\mathcal{C}$ as the result of the marginalization of another PC $\mathcal{B}$ encoding a joint probability distribution over independent binary variables $\mathcal{E}\cup\mathcal{R}$:

$$\phi_{\mathcal{C}}(s,p,o) = \sum_{\mathbf{x}\in\{0,1\}^m} \phi_{\mathcal{B}}(s=1, p=1, o=1, \mathcal{E}'\cup\mathcal{R}' = \mathbf{x})$$

where $\mathcal{E}' = \mathcal{E}\setminus\{s,o\}$, $\mathcal{R}' = \mathcal{R}\setminus\{p\}$ and $m = |\mathcal{E}'\cup\mathcal{R}'|$. This can be realized by substituting the categorical input distribution for S or O (resp. P) in $\mathcal{C}$ by a product over the binary variables in $\mathcal{E}$ (resp. $\mathcal{R}$). If we do so for our DISTMULT+, we retrieve TRACTOR [Friedman and Van den Broeck, 2020].

Therefore, we are able to answer any UCQ Q with a circuit $\mathcal{B}$ exactly and efficiently by i) preprocessing them and factorizing each binary atom as in Friedman and Van den Broeck [2020], ii) compiling the logic query into a smooth and decomposable propositional logic circuit as in Van den Broeck et al. [2011], and iii) computing the expectation of Q w.r.t. $\mathcal{B}$ which can be done efficiently by multiplying the resulting logic circuit with $\mathcal{B}$ [Vergari et al., 2021].

Dataset	Model	1vsALL		1vsALL+MLE	
		MRR	Hits@1	MRR	Hits@1
Nations	CP	0.792	0.676	—	—
	CP+	0.804	0.700	0.786	0.677
	CP^2	0.797	0.699	**0.805**	**0.705**
UMLS	CP	**0.943**	**0.897**	—	—
	CP+	0.855	0.759	0.854	0.759
	CP^2	0.920	0.873	0.896	0.817
Kinship	CP	0.855	0.769	—	—
	CP+	0.722	0.598	0.734	0.612
	CP^2	0.868	0.796	**0.889**	**0.827**

Table 1: Best MRR and Hits@1 on the test sets of small multi-relational knowledge graphs with CP as a baseline.

5 EMPIRICAL EVALUATION

Here we provide some preliminary experiments to support the use of tractable KGE circuits. Specifically, we investigate how expressive are our monotonic restriction and squared circuits when compared to unrestricted KGE models. To this end, we compare CP against our alternatives CP+ and CP^2 on link prediction datasets: Nations, UMLS, Kinship, WN18RR and FB15k-237. App. E reports the experimental setting details. We use the 1vsALL objective in Eq. (2) and, for CP+ and CP^2, also the composite likelihood combining the 1vsALL and MLE objectives.

Table 1 shows the results in terms of the test mean reciprocal rank (MRR) and Hits@1 after a grid search over hyperparameters. The metrics are averaged over 5 independent trials with different seeds. CP+ and CP^2 achieve competitive performance with respect to CP, and perform better in Nations and Kinship using the composite objective. On WN18RR and FB15k-237 instead, we performed experiments using the 1vsALL objective. On WN18RR and FB15k-237, CP achieves MRRs of 0.440 and 0.340, while CP^2 achieves MRRs of 0.392 and 0.273 respectively. Furthermore, CP^2 always performs better than CP+ confirming that squared circuits can be more expressive than the ones obtained by monotonic restriction.

References

Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. Complex query answering with neural link predictors. In *ICLR*. OpenReview.net, 2021.

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*, pages 5184–5193. Association for Computational Linguistics, 2019.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

Jatin Chauhan, Priyanshu Gupta, and Pasquale Minervini. A probabilistic framework for knowledge graph data augmentation. *arXiv preprint arXiv:2110.13205*, 2021.

Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. *CoRR*, abs/2110.02834, 2021.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic modeling. Technical report, 2020.

Nilesh Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, 2007.

Alexis de Colnet and Stefan Mengel. A compilation of succinctness results for arithmetic circuits. *arXiv preprint arXiv:2110.13014*, 2021.

Aaron W. Dennis. Algorithms for learning the structure of monotone and nonmonotone sum-product networks, 2016.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818. AAAI Press, 2018.

John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

Tal Friedman and Guy Van den Broeck. Symbolic querying of vector spaces: Probabilistic databases meets relational embeddings. In *UAI*, volume 124, pages 1268–1277. AUAI Press, 2020.

William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *NeurIPS*, pages 2030–2041, 2018.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM*, 51(3):455–500, 2009.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR, 2018.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *IEEE*, 104(1):11–33, 2016.

Georgii S. Novikov, Maxim E. Panov, and Ivan V. Oseledets. Tensor-train density estimation. In *UAI*, volume 161 of *Proceedings of Machine Learning Research*, pages 1321–1331. AUAI Press, 2021.

Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Xiaoting Shao, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In Amir Globerson and Ricardo Silva, editors, *UAI*, volume 115 of *Proceedings of Machine Learning Research*, pages 334–344. AUAI Press, 2019.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference on Machine Learning*, pages 7563–7574. PMLR, 2020.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *ICLR*. OpenReview.net, 2020.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *ICLR*. OpenReview.net, 2020.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality*. ACL, 2015.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

Leslie G Valiant. Negation can be exponentially powerful. In *Proceedings of the eleventh annual ACM symposium on theory of computing*, pages 189–196, 1979.

Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In *IJCAI*, pages 2178–2185. IJCAI/AAAI, 2011.

Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of shap explanations. In *Proceedings of the 35th Conference on Artificial Intelligence (AAAI)*, 2021.

Cristiano Varin, Nancy Reid, and David Firth. An overview of composite likelihood methods. *Statistica Sinica*, 21, 01 2011.

Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Visualizing and understanding sum-product networks. *Machine Learning*, 108(4):551–573, 2019.

Antonio Vergari, YooJung Choi, Robert Peharz, and Guy Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications. In *Tutorial at the The 34th AAAI Conference on Artificial Intelligence*, 2020.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations: From simple transformations to complex information-theoretic queries. *arXiv preprint arXiv:2102.06137*, 2021.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR (Poster)*, 2015.

A FROM KGES TO CIRCUITS

Proof. For the extended proof of Thm. 1, we prove it for TuckER, since the other KGE models are based on specializations of the Tucker tensor factorization [Balazevic et al., 2019]. For instance, the DistMult score function can be written as TuckER's where $\mathcal{T}$ is a three-order tensor having ones on the superdiagonal and zeros elsewhere.

The presented proof constructs a circuit that compute the TuckER score function in a bottom-up way, i.e., by creating the input functionals of the circuit and by transforming the tensor multiplications into corresponding sum and product units. Given $(s,p,o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, the TuckER score function computes:

$$\phi_{\text{TuckER}}(s,p,o) = [\![\mathcal{T};\ \mathbf{e}_s, \mathbf{w}_p, \mathbf{e}_o]\!] \tag{5}$$

$$= \mathcal{T} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_p \times_3 \mathbf{e}_o \tag{6}$$

$$= \sum_{i=1}^{R_e} \sum_{j=1}^{R_p} \sum_{k=1}^{R_e} \tau_{ijk} e_{si} w_{pj} e_{ok} \tag{7}$$

where $\times_n$ denotes the tensor product along the n-th mode, and R_e, R_p are the dimensions of the embeddings of entities and relation types respectively. For subjects, relation types and objects we introduce input functionals as parametric mappers such that:

$$\mathcal{I}_i^S := \mathcal{C}_i^S(\{S\};\ \mathbf{E}) \qquad \mathcal{I}_i^S(s) = e_{si} \tag{8}$$

$$\mathcal{I}_i^P := \mathcal{C}_i^P(\{P\};\ \mathbf{W}) \qquad \mathcal{I}_i^P(p) = w_{pi} \tag{9}$$

$$\mathcal{I}_i^O := \mathcal{C}_i^O(\{O\};\ \mathbf{E}) \qquad \mathcal{I}_i^O(o) = e_{oi} \tag{10}$$

where $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times R_e}, \mathbf{W} \in \mathbb{R}^{|\mathcal{R}| \times R_p}$ are the parameters. We introduce $R_e^2 R_p$ product units that compute products of the input functionals:

$$\mathcal{P}_{ijk} := \mathcal{C}_{ijk}(\{S, P, O\}) \tag{11}$$

$$\mathcal{P}_{ijk}(s,p,o) = \mathcal{I}_i^S(s) \cdot \mathcal{I}_j^P(p) \cdot \mathcal{I}_k^O(o) \tag{12}$$

Finally, we introduce a sum unit that computes a weighted summation of the results given by the product units:

$$\mathcal{S} := \mathcal{C}(\{S, P, O\};\ \mathcal{T}) \tag{13}$$

$$\mathcal{S}(s,p,o) = \sum_{\substack{(i,j,k) \in \\ [R_e] \times [R_p] \times [R_e]}} \tau_{ijk} \cdot \mathcal{P}_{ijk}(s,p,o) \tag{14}$$

where $[n]$ denotes the set $\{1, \ldots, n\}$. It is straightforward to see that $\mathcal{S}(s,p,o) = \phi_{\text{TuckER}}(s,p,o)$ for any triple by construction.

Notice that each product unit $\mathcal{P}_{ijk}$ fully factorizes the scope $\{S, P, O\}$. For this reason the resulting circuit is decomposable and structured-decomposable. The inputs of the sum unit $\mathcal{S}$ share the same scope $\{S, P, O\}$, hence the circuit is also smooth. □

B PROBABILISTIC CIRCUITS

Definition 5 (Probabilistic Circuit)**.** A probabilistic circuit (PC) over variables $\mathbf{X}$ is a circuit $\mathcal{C}$ encoding a function that is non-negative for all values of $\mathbf{X}$, i.e., $\forall \mathbf{x} \in \mathsf{val}(\mathbf{X}) : \mathcal{C}(\mathbf{x}) \geq 0$.

Theorem 3 (Tractable integration)**.** Let $\mathcal{C}$ be a smooth and decomposable circuit over variables $\mathbf{X}$ with input functions that can be tractably integrated. For any $\mathbf{Y} \subseteq \mathbf{X}$, $\mathbf{y} \in \mathsf{val}(\mathbf{Y})$, $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$, the following integral can be computed in time $\Theta(|\mathcal{C}|)$, where $|\mathcal{C}|$ denotes the size of the circuit [Choi et al., 2020].

$$\int_{\mathbf{z} \in \mathsf{val}(\mathbf{Z})} \mathcal{C}(\mathbf{y}, \mathbf{z}) d\mathbf{Z} \tag{15}$$

Here the integral symbol denotes the usual integration for continuous variables, while summation over states for discrete variables.

Given a smooth and decomposable PC $\mathcal{C}$, Thm. 3 asserts that we can perform marginalization in linear time w.r.t. the size of $\mathcal{C}$. Therefore, we can answer full evidence ($\Pr(\mathbf{X})$), marginal ($\Pr(\mathbf{Y})$ with $\mathbf{Y} \subset \mathbf{X}$) and conditional ($\Pr(\mathbf{Y} \mid \mathbf{Z})$ with $\mathbf{Y} \subset \mathbf{X}$ and $\mathbf{Z} \subset \mathbf{X} \setminus \mathbf{Y}$) probabilistic queries exactly and efficiently by evaluating the circuit in a single forward pass [Choi et al., 2020].

C REALIZING COMPLEX+

The score function of ComplEx explicitly contains a subtraction, as showed below.

$$\phi_{\text{ComplEx}}(s,p,o) = \mathrm{Re}(\langle \mathbf{e}_s, \mathbf{w}_p, \overline{\mathbf{e}_o} \rangle) \tag{16}$$

$$\begin{aligned} &= \langle \mathrm{Re}(\mathbf{e}_s), \mathrm{Re}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o) \rangle \\ &+ \langle \mathrm{Im}(\mathbf{e}_s), \mathrm{Re}(\mathbf{w}_p), \mathrm{Im}(\mathbf{e}_o) \rangle \\ &+ \langle \mathrm{Re}(\mathbf{e}_s), \mathrm{Im}(\mathbf{w}_p), \mathrm{Im}(\mathbf{e}_o) \rangle \\ &- \langle \mathrm{Im}(\mathbf{e}_s), \mathrm{Im}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o) \rangle \end{aligned} \tag{17}$$

Restricting the real and imaginary parts to be non-negative using monotonic restriction as described in Sec. 3.1 is not sufficient to obtain a PC, since it could output negative values for some inputs. Under monotonic restriction, we ensure that $\phi_{\text{ComplEx}}(s,p,o) \geq 0$ for any $(s,p,o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ by enforcing the following constraint.

$$\langle \mathrm{Re}(\mathbf{e}_s), \mathrm{Re}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o) \rangle \geq \langle \mathrm{Im}(\mathbf{e}_s), \mathrm{Im}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o) \rangle \tag{18}$$

The constraint can be simplified to two inequalities:

$$\forall u \in \mathcal{E} \quad \mathrm{Re}(e_{ui}) \geq \mathrm{Im}(e_{ui}) \tag{19}$$

$$\forall p \in \mathcal{R} \quad \mathrm{Re}(w_{pi}) \geq \mathrm{Im}(w_{pi}) \tag{20}$$

In other words, we assume that the real part of each parameter is always greater or equal than the corresponding imaginary part. Practically, we can *parameterize* the imaginary part in function of the real part:

$$\forall u \in \mathcal{E} \quad \mathrm{Im}(e_{ui}) := \mathrm{Re}(e_{ui}) \cdot \sigma(\theta_{ui}) \tag{21}$$
$$\forall p \in \mathcal{R} \quad \mathrm{Im}(w_{pi}) := \mathrm{Re}(w_{pi}) \cdot \sigma(\gamma_{pi}) \tag{22}$$

where σ denotes the logistic function and $\theta_{ui}, \gamma_{pi} \in \mathbb{R}$ are additional parameters for entities and relation types respectively. The parametrization of the imaginary parts using Eqs. (21) and (22) is sufficient for the satisfaction of the constraint showed in Eq. (18), and also maintains the same number of parameters of ComplEx.

We denote as ComplEx+ the PC corresponding to ComplEx encoding the function:

$$\phi_{\text{ComplEx+}} := \mathcal{C}_1 + \mathcal{C}_2 + \mathcal{C}_3 \tag{23}$$
$$\mathcal{C}_1(s,p,o) = \langle \mathrm{Im}(\mathbf{e}_s), \mathrm{Re}(\mathbf{w}_p), \mathrm{Im}(\mathbf{e}_o)\rangle \tag{24}$$
$$\mathcal{C}_2(s,p,o) = \langle \mathrm{Re}(\mathbf{e}_s), \mathrm{Im}(\mathbf{w}_p), \mathrm{Im}(\mathbf{e}_o)\rangle \tag{25}$$
$$\begin{aligned}\mathcal{C}_3(s,p,o) = &\langle \mathrm{Re}(\mathbf{e}_s), \mathrm{Re}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o)\rangle \\ &- \langle \mathrm{Im}(\mathbf{e}_s), \mathrm{Im}(\mathbf{w}_p), \mathrm{Re}(\mathbf{e}_o)\rangle\end{aligned} \tag{26}$$

where $\mathcal{C}_1, \mathcal{C}_2$ are PCs and $\mathcal{C}_3$ is a Twin Sum-Product Network (TwinSPN) [Dennis, 2016], since it is a subtraction of two PCs with pairwise constrained parameters that ensure that $\mathcal{C}_3$ always outputs non-negative values.

D PARTITION FUNCTION OF SQUARED KGE CIRCUITS

KGE Circuit	Time	Space
CP^2	$\mathcal{O}(\lvert\mathcal{E}\rvert \cdot R^2 + \lvert\mathcal{R}\rvert \cdot R^2)$	$\mathcal{O}(R^2)$
DistMult^2	$\mathcal{O}(\lvert\mathcal{E}\rvert \cdot R^2 + \lvert\mathcal{R}\rvert \cdot R^2)$	$\mathcal{O}(R^2)$
ComplEx^2	$\mathcal{O}(\lvert\mathcal{E}\rvert \cdot R^2 + \lvert\mathcal{R}\rvert \cdot R^2)$	$\mathcal{O}(R^2)$
Rescal^2	$\mathcal{O}(\lvert\mathcal{E}\rvert \cdot R^2 + \lvert\mathcal{R}\rvert \cdot R^4)$	$\mathcal{O}(R^4)$
TuckER^2	$\mathcal{O}(\lvert\mathcal{E}\rvert \cdot R_e^2 + \lvert\mathcal{R}\rvert \cdot R_p^2 + R_e^4 R_p^2)$	$\mathcal{O}(R_e^4 R_p^2)$

Table 2: Time and additional space complexity of computing the partition function of squared KGE circuits.

D.1 CP^2, DISTMULT^2, COMPLEX^2

Here we derive the partition function of CP^2. For the scoring functions of DistMult^2 and ComplEx^2 the derivation is similar, since they share the same computational graph. The squared CP scoring function ϕ_{CP^2} can be written as:

$$\begin{aligned}\phi_{\mathrm{CP}^2}(s,p,o) &= \phi^2_{\mathrm{CP}}(s,p,o) &(27)\\ &= \langle \mathbf{e}_s, \mathbf{w}_p, \mathbf{e}_o\rangle^2 &(28)\\ &= \sum_{i=1}^R \sum_{j=1}^R e_{si}e_{sj}w_{pi}w_{pj}e_{oi}e_{oj} &(29)\end{aligned}$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^R$ and $\mathbf{w}_p \in \mathbb{R}^R$ are rows of matrices $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{|\mathcal{E}|\times R}$ and $\mathbf{W} \in \mathbb{R}^{|\mathcal{R}|\times R}$ respectively. The partition function $\mathcal{Z}$ can be computed as:

$$\begin{aligned}\mathcal{Z} &= \sum_{s\in\mathcal{E}}\sum_{p\in\mathcal{R}}\sum_{o\in\mathcal{E}} \phi_{\mathrm{CP}^2}(s,p,o) &(30)\\ &= \sum_{i=1}^R\sum_{j=1}^R \left(\sum_{s\in\mathcal{E}} e_{si}e_{sj}\right)\left(\sum_{p\in\mathcal{R}} w_{pi}w_{pj}\right)\left(\sum_{o\in\mathcal{E}} e_{oi}e_{oj}\right) &(31)\\ &= \langle \mathrm{vec}(\mathbf{U}^T\mathbf{U}), \mathrm{vec}(\mathbf{W}^T\mathbf{W}), \mathrm{vec}(\mathbf{V}^T\mathbf{V})\rangle &(32)\end{aligned}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization operator. With the simplest algorithm for matrix multiplication, we recover that computing the partition function of CP^2 requires time $\mathcal{O}(|\mathcal{E}| \cdot R^2 + |\mathcal{R}| \cdot R^2)$ and additional space $\mathcal{O}(R^2)$.

D.2 RESCAL^2

The squared Rescal scoring function ϕ_{RESCAL^2} can be written as:

$$\begin{aligned}\phi_{\mathrm{RESCAL}^2}(s,p,o) &= \phi^2_{\mathrm{RESCAL}}(s,p,o) &(33)\\ &= \left(\mathbf{e}_s^T\mathbf{W}_p\mathbf{e}_o\right)^2 &(34)\\ &= \sum_{(i,j,k,l)\in[R]^4} e_{si}e_{sk}w_{pij}w_{pkl}e_{oj}e_{ol} &(35)\end{aligned}$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^R$ are rows of matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}|\times R}$ and $\mathbf{W}_p \in \mathbb{R}^{R\times R}$ are slices of tensor $\mathcal{W} \in \mathbb{R}^{|\mathcal{R}|\times R\times R}$ along the first mode. As computing the partition function of CP^2 requires operating on 2-dimensional tensors (i.e., matrices), we operate on a 4-dimensional tensor for Rescal^2. We do so compactly by using the *einsum* notation. The partition function $\mathcal{Z}$ can be computed as:

$$\begin{aligned}\mathcal{Z} &= \sum_{s\in\mathcal{E}}\sum_{p\in\mathcal{R}}\sum_{o\in\mathcal{E}} \phi_{\mathrm{RESCAL}^2}(s,p,o) &(36)\\ &= \sum_{(i,j,k,l)\in[R]^4} \left(\sum_{s\in\mathcal{E}} e_{si}e_{sk}\right)\left(\sum_{p\in\mathcal{R}} w_{pij}w_{pkl}\right)\left(\sum_{o\in\mathcal{E}} e_{oj}e_{ol}\right) &(37)\\ &= \mathbf{u}^T\mathbf{V}\mathbf{u} &(38)\end{aligned}$$

where

$$\begin{aligned}\mathbf{u} &= \mathrm{vec}(\mathbf{E}^T\mathbf{E}) &(39)\\ \mathbf{V} &= \mathrm{reshape}(\widehat{\mathcal{W}}, R^2 \times R^2) &(40)\\ \widehat{\mathcal{W}} &= \mathrm{einsum}(\text{“}nij, nkl \to ikjl\text{”}, \mathcal{W}, \mathcal{W}) &(41)\end{aligned}$$

and $\mathrm{reshape}(\cdot,\cdot)$ denotes the reshape operator. We recover that computing the partition function of Rescal^2 requires time $\mathcal{O}(|\mathcal{E}| \cdot R^2 + |\mathcal{R}| \cdot R^4)$ and additional space $\mathcal{O}(R^4)$.

D.3 TUCKER^2

The derivation of the partition function of TUCKER^2 is similar to the one for RESCAL^2. The squared TUCKER scoring function ϕ_{TUCKER^2} can be written as:

$$\phi_{\text{TUCKER}^2}(s,p,o) = \phi^2_{\text{TUCKER}}(s,p,o) \tag{42}$$
$$= (\mathcal{T} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_p \times_3 \mathbf{e}_o)^2 \tag{43}$$
$$= \left(\sum_{i=1}^{R_e} \sum_{j=1}^{R_p} \sum_{k=1}^{R_e} \tau_{ijk} e_{si} w_{pj} e_{ok} \right)^2 \tag{44}$$

where $\mathbf{e}_s, \mathbf{e}_o \in \mathbb{R}^{R_e}$ are rows of matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times R_e}$, $\mathbf{w}_p$ is a row of matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{R}| \times R_p}$, and $\mathcal{T} \in \mathbb{R}^{R_e \times R_p \times R_e}$ denotes the core tensor. The partition function $\mathcal{Z}$ can be computed as:

$$\mathcal{Z} = \sum_{s \in \mathcal{E}} \sum_{p \in \mathcal{R}} \sum_{o \in \mathcal{E}} \phi_{\text{TUCKER}^2}(s,p,o) \tag{45}$$
$$= \mathcal{V} \times_1 \mathbf{u} \times_2 \mathbf{w} \times_3 \mathbf{u} \tag{46}$$

where

$$\mathbf{u} = \text{vec}(\mathbf{E}^T \mathbf{E}) \tag{47}$$
$$\mathbf{w} = \text{vec}(\mathbf{W}^T \mathbf{W}) \tag{48}$$
$$\mathcal{V} = \text{reshape}(\widehat{\mathcal{T}}, R_e^2 \times R_p^2 \times R_e^2) \tag{49}$$
$$\widehat{\mathcal{T}} = \text{einsum}(\text{“}ijk, pqr \to ipjqkr\text{”}, \mathcal{T}, \mathcal{T}) \tag{50}$$

Therefore, for TUCKER^2 computing the partition function requires time $\mathcal{O}(|\mathcal{E}| \cdot R_e^2 + |\mathcal{R}| \cdot R_p^2 + R_e^4 R_p^2)$ and additional space $\mathcal{O}(R_e^4 R_p^2)$.

E EXPERIMENTAL SETTING

Table 3 shows some statistics about the considered datasets.

Table 4 shows the hyperparameters search for CP, CP+ and CP^2 on *small* datasets: Nations, UMLS and Kinship. Moreover, Table 5 shows the hyperparameters search on *large* datasets: FB15k-237 and WN18RR. All the models are trained by SGD with the Adagrad optimizer [Duchi et al., 2011] for 200 epochs, and by augmenting the training data with reciprocal triples [Lacroix et al., 2018].

Following Chen et al. [2021], we intialize the parameters of CP by sampling from a normal distribution $\mathcal{N}(0, 10^{-3})$. In order to ensure non-negative parameters in CP+, we re-parameterize them with their logarithm, and perform computations in log-space. We initialize the parameters of CP+ directly in log-space by sampling from a normal distribution $\mathcal{N}(0, 10^{-2})$. In CP^2, we initialize the parameters by sampling from a log-normal distribution $\text{LogNormal}(0, 10^{-2})$, and allow them to become negative during training. The reason of using a log-normal distribution is that by doing so we ensure that the scores in log-space are approximately normally distributed in the initial optimization steps. Empirically this resulted in CP^2 converging to a better local minimum.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	# Train	# Valid	# Test
Nations	14	55	1,592	100	301
UMLS	135	46	5,216	652	661
Kinship	104	25	8,544	1,068	1,074
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Table 3: Statistics of Nations, UMLS, Kinship, FB15k-237 and WN18RR showing the number of entities $|\mathcal{E}|$ and relation types $|\mathcal{R}|$, and the number of triples in training, validation and test splits.

Model	Rank	Learning Rate	Batch Size
CP	$[200, 500]$	$[0.01, 0.1]$	$[100, 500]$
CP+	$[200, 500]$	$[0.1, 1.0]$	$[100, 500]$
CP^2	$[200, 500]$	$[0.1, 1.0]$	$[100, 500]$

Table 4: Hyperparameters search for CP, CP+ and CP^2 on Nations, UMLS and Kinship.

Model	Rank	Learning Rate	Batch Size
CP	2000	$[0.01, 0.1]$	500
CP+	2000	$[0.1, 1.0]$	500
CP^2	2000	$[0.1, 1.0]$	500

Table 5: Hyperparameters search for CP, CP+ and CP^2 on FB15k-237 and WN18RR.