Parameter Averaging in Link Prediction

Rupesh Sapkota
Heinz Nixdorf Institute, Paderborn University
Paderborn, Germany
rupezzz@mail.uni-paderborn.de

Caglar Demir
Heinz Nixdorf Institute, Paderborn University
Paderborn, Germany
caglar.demir@upb.de

Arnab Sharma
Heinz Nixdorf Institute, Paderborn University
Paderborn, Germany
arnab.sharma@uni-paderborn.de

Axel-Cyrille Ngonga Ngomo
Heinz Nixdorf Institute, Paderborn University
Paderborn, Germany
axel.ngonga@upb.de

Abstract

Ensemble methods are widely employed to improve generalization in machine learning. This has also prompted the adoption of ensemble learning for the knowledge graph embedding (KGE) models in performing link prediction. Typical approaches to this end train multiple models as part of the ensemble, and the diverse predictions are then averaged. However, this approach has some significant drawbacks. For instance, the computational overhead of training multiple models increases latency and memory overhead. In contrast, model merging approaches offer a promising alternative that does not require training multiple models. In this work, we introduce model merging, specifically *weighted averaging*, in KGE models. Herein, a running average of model parameters from a training epoch onward is maintained and used for predictions. To address this, we additionally propose an approach that selectively updates the running average of the ensemble model parameters only when the generalization performance improves on a validation dataset. We evaluate these two different weighted averaging approaches on link prediction tasks, comparing the state-of-the-art benchmark ensemble approach. Additionally, we evaluate the weighted averaging approach considering literal-augmented KGE models and multi-hop query answering tasks as well. The results demonstrate that the proposed weighted averaging approach consistently improves performance across diverse evaluation settings.

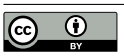
CCS Concepts

• **Computing methodologies** → **Ensemble methods**; **Machine learning**; *Knowledge representation and reasoning*.

Keywords

Knowledge Graphs, Embeddings, Ensemble Learning

ACM Reference Format:
Rupesh Sapkota, Caglar Demir, Arnab Sharma, and Axel-Cyrille Ngonga Ngomo. 2025. Parameter Averaging in Link Prediction. In *Knowledge Capture Conference 2025 (K-CAP '25), December 10–12, 2025, Dayton, OH, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3731443.3771365



K-CAP '25, Dayton, OH, USA

ACM ISBN 979-8-4007-1867-0/25/12
https://doi.org/10.1145/3731443.3771365

1 Introduction

Ensemble learning is one of the most effective techniques to improve the generalization of Machine Learning (ML) algorithms across challenging tasks [9, 13, 22]. In its simplest form, an ensemble model is constructed from a set of K different learners over the same training set [6]. At test time, a new data point is classified by taking a (weighted) average of K the predictions of the K learners [28]. Although prediction averaging often improves predictive accuracy, uncertainty estimation, and out-of-distribution robustness [12, 21], it incurs computational overhead of training K models, increased latency and memory requirements at test time [21].

In recent years, a number of works have considered employing the ensemble approach in knowledge graph embedding models [14, 20, 24, 26, 34, 37]. One of the earliest work to this end is by Krompass et al. [20], where multiple heterogeneous models are combined to get an ensemble of KGE models. The scores of these models are averaged to get the final prediction scores. A similar approach is also followed in [14, 26]. This leads to better accuracy, however, at the cost of training multiple models. There exist other approaches such as subgraph-based ensembling [24, 34], low dimensional ensemble [37], snpashot-based ensemble [17, 29]. However, most of these approaches rely on explicit score-level aggregation or model duplication, which either require significant storage for storing full checkpoint ensembles and manual tuning using relation-aware weight assignment. This incurs additional inference overhead.

Recently, Izmaliov et al. [18] show that Stochastic Weight Averaging (SWA) technique often improves the generalization performance of a neural network by averaging K *snapshots* of parameters at the end of each epoch from a specific epoch onward. Therefore, while a neural network is being trained (called an underlying running model), an ensemble model is constructed through averaging the trajectory of an underlying running model in a parameter space. Therefore, at test time, the memory and running time requirements of using SWA parameter ensemble are identical to the requirements of using a single neural network. Although the idea of averaging parameters of linear models to accelerate Stochastic Gradient Descent (SGD) on convex problems dates back to the works by Polyak et al. [23], Izmailov et al. [18] show that an effective parameter ensemble of ML models can be built by solely maintaining a running average of parameters from a specific epoch onward.

In this work, we introduce the weighted averaging technique for the knowledge graph embedding models. Instead of training individual models, herein we obtain an ensemble of KGE models within

a single training run. Therefore, unlike the existing ensembling approaches that require training and storing multiple independent models, our method maintains a running average of model weights throughout training, capturing diverse solutions in weight space. We found that selecting an unfittingly low number of training epochs leads SWA to suffer more from underfitting than an underlying running model. In other words, a parameter ensemble model is heavily influenced by the early stages of training.

Therefore, striking the right balance in the choice of training epochs is crucial for harnessing the full potential of SWA in enhancing the generalization. To address this, we propose Adaptive Stochastic Weight Averaging (ASWA) technique that extends SWA by building a parameter ensemble based on an adaptive schema governed by the generalization trajectory on the validation dataset.

ASWA can be seen as a combination of SWA with the early stopping technique, as SWA accepts all updates on a parameter ensemble, while the early stopping technique rejects any updates on a running model.

To evaluate the effectiveness of both the weighted averaging techniques in KGE models, we perform an extensive evaluation considering link prediction, literal score prediction, and multi-hop query answering tasks. The results of the evaluations suggest that weighted averaging improves the generalization performance over single and the existing state-of-the-art ensemble approaches. The main contributions of this paper are as follows:

(1) We introduce the weighted averaging approach in the knowledge graph embedding domain.
(2) We extend the stochastic weighted averaging approach by exploiting the early stopping technique.
(3) We perform extensive evaluations considering different link prediction tasks to quantify the effectiveness of the weighted averaging approach.
(4) We compare the effectiveness of the two weighted averaging approaches with diverse KGE ensemble models.
(5) We provide an open-source implementation of our approach.

2 Related Work

Ensemble learning has been extensively studied in machine learning research [22]. At its core, an ensemble model is created by combining the predictions of a set of K learners, often through simple averaging [6]. During inference, the final prediction is obtained by averaging the outputs of the K learners, resulting in a more robust and accurate model. For example, averaging the predictions of K independently trained neural networks of the same architecture has been shown to significantly enhance test set performance [1].

However, in the specific context of link prediction, ensemble-based methods have not been explored extensively [30]. One of the earliest works by Krompass et al. [20] proposed *model-based ensembling* methods, wherein heterogeneous base models (e.g., TransE, RESCAL, neural architectures) are combined via averaging their prediction scores, achieving better accuracy but at the cost of training multiple full-sized models. Later works by Gregucci et al. [14], Rivas-Barragan et al. [26], also follow similar approach. For instance, Building on this framework, Gregucci et al. [14] proposed an attention-based ensembling approach that learns to weight model contributions per query. This yields an ensemble of models that outperforms individual embedding models and adapts flexibly to varied relation types. In a biomedical context, Rivas-Barragan et al. [26] applied ensemble modeling to link-prediction tasks in drug-disease knowledge graphs. They trained ten different Knowledge Graph Embedding Models (KGEMs) and experimented with different ensemble strategies, such as rank-based and distribution-based aggregation for combining predictions.

A different line of work on KGE ensemble focus on subgraph-based ensembling [24, 34]. This approach partitions the KG into subgraphs, trains separate embeddings on each, and aggregates results, however offering mixed evidence on computational efficiency. Xu et al. [37] proposed *low-dimensional ensembling* where several instances of base models with smaller embedding sizes are trained independently and their scores averaged. This method achieves better performance under the same memory budget compared to a larger single model, however, it still requires separate training runs for each instance. Most recently, SnapE [29] introduces a snapshot ensemble technique tailored for KG link prediction. By storing multiple model checkpoints during a single training loop with cyclical learning rates, SnapE obtains diverse base models at the cost of one training run. The authors therein showed the effectiveness and efficiency of this approach over the previous ensemble-based KGE models. Typically, most of the ensemble techniques can lead to substantial improvements in generalization across various learning problems. Yet, as $|\mathcal{E}| + |\mathcal{R}|$ grows, leveraging the prediction average technique becomes computationally prohibitive. Attempts to alleviate the computational overhead of training multiple models have been extensively studied. For instance, Xie et al. [36] show that saving parameters of a neural network periodically during training and composing a final prediction via a voting schema improves the generalization performance. Moreover, the dropout technique can also be seen as a form of ensemble learning [15, 31].

More specifically, preventing the co-adaptation of parameters by stochastically forcing them to be zero can be seen as a *geometric averaging* [5, 13, 35]. Similarly, Monte Carlo Dropout can be seen as a variant of the Dropout that is used to approximate model uncertainty without sacrificing either computational complexity or test accuracy [11]. Garipov et al. [12] show that optima of neural networks are connected by simple pathways having near constant training accuracy [10]. As an alternative to typical ensemble-based approaches, Izmailov et al. [18] proposed a model merging technique called Stochastic Weight Averaging (SWA) that builds a parameter ensemble model with *almost* the same training and exact test time cost as a single model. Note that all these works addressing issues with typical ensemble approaches consider neural network models. To the best of our knowledge, this work is the first to study weighted averaging of KGE model parameters.

3 Background

We start by giving the formal definition of the KGE models, and then we introduce the weighted averaging approach.

Let $\mathcal{E}$ be a set of entities and $\mathcal{R}$ a set of relations. A knowledge graph (KG) is defined as a set of triples:

$$\mathcal{G} := \{(h, r, t) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}, \tag{1}$$

where each triple represents a relation r between entities h and t. To enable learning, knowledge graph embedding (KGE) models map entities and relations into a continuous space $\mathbb{V}^d$ (e.g., $\mathbb{R}, \mathbb{C}, \mathbb{H}, \mathbb{O}$), yielding embeddings $\mathbf{E} \in \mathbb{V}^{|\mathcal{E}|\times d}$ and $\mathbf{R} \in \mathbb{V}^{|\mathcal{R}|\times d}$ [4, 32, 38, 40]. Typically, KGE models are used for link prediction [16], i.e., their scoring function is $\phi_\Theta : \mathcal{E}\times\mathcal{R}\times\mathcal{E} \mapsto \mathbb{R}$, where Θ denotes parameters and often comprise $\mathbf{E}$, $\mathbf{R}$, and additional parameters (e.g., affine transformations, batch normalizations, convolutions). Given an assertion in the form of a triple $(\mathsf{h}, \mathsf{r}, \mathsf{t}) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, a prediction $\hat{y} := \phi_\Theta(\mathsf{h}, \mathsf{r}, \mathsf{t})$ signals the likelihood of $(\mathsf{h}, \mathsf{r}, \mathsf{t})$ being true [8].

Stochastic Weight Averaging Izmailov et al. [18] proposed Stochastic Weight Averaging (SWA) that builds a parameter ensemble model having *almost* the same training and test time cost of a single model. For the KGE models, we define the SWA ensemble parameter (i.e., embedding) update as follows.

$$\Theta_{\text{SWA}} \leftarrow \frac{\Theta_{\text{SWA}} \cdot n_{\text{models}} + \Theta}{n_{\text{models}} + 1} \tag{2}$$

where Θ_{SWA} and Θ denote the embeddings of the ensemble model and the running model, respectively. At each SWA update, n_{models} is incremented by 1. To be effective, finding a *good* start epoch for SWA is important. Selecting an unfittingly low start point and total number of epochs can lead SWA to underfit. Similarly, starting SWA only on the last few epochs does not lead to an improvement, as shown in [18]. Therefore, it is important to perform weighted averaging only when it leads to performance improvement. Using this observation, we propose an *adaptive* version of the weighted averaging approach which we describe next.

4 Adaptive Stochastic Weight Averaging

While SWA averages embeddings uniformly from a fixed start epoch j, its effectiveness is highly sensitive to the choice of j: starting too early can result in underfitting, whereas averaging too late fails to improve generalization [18]. In contrast, our adaptive formulation ASWA constructs the ensemble as a weighted sum of intermediate embeddings of the KGE model, which can be defined as follows.

$$\Theta_{\text{ASWA}} = \sum_{i=1}^{N} \boldsymbol{\alpha}_i \odot \Theta_i, \tag{3}$$

where $\Theta_i \in \mathbb{R}^d$ stands for a embedding vector of a running model at the $i-$th iteration. $\odot$ denotes the scalar vector multiplication. An ensemble coefficient $\boldsymbol{\alpha}_i \in [0, 1]$ is a scalar value denoting the weight of Θ_i in Θ_{ASWA}.Here, N denotes the number of epochs as does in SWA [18]. Note that, Θ_{ASWA} corresponds to Θ_{SWA} starting from the first iteration, if $\boldsymbol{\alpha} = \frac{\mathbf{1}}{\mathbf{N}}$, and Θ_{ASWA} corresponds to Θ_{SWA} starting from the j-th iteration if $\boldsymbol{\alpha}_{0:j} = \mathbf{0}$ and $\boldsymbol{\alpha}_{j+1:N} = \frac{\mathbf{1}}{\mathbf{N-j}}$.

We find that constructing an ensemble embedding model by averaging over multiple nonadjacent epoch intervals (e.g. two nonadjacent epoch intervals i to j and k to m s.t. $m \geq k+1 \geq j+1 \geq i$) may be more advantageous than selecting a single start epoch. With these considerations, we argue that determining $\boldsymbol{\alpha}$ according to an adaptive schema governed by the generalization trajectory on the validation dataset can be more advantageous than using a prefixed schema. By this, we aim to ensure that Θ_{ASWA} does not suffer from underfitting more than the running embedding model Θ and overfitting more than Θ_{SWA}.

Herein, the weights $\boldsymbol{\alpha}_i$ for each individual embedding model can be determined in a fashion akin to the early stopping technique [25]. More specifically, the trajectory of the validation losses can be tracked at the end of each epoch. By this, as the generalization performance degrades, the training can be stopped.

Algorithm 1 Adaptive Stochastic Weight Averaging (ASWA)

Require: Initial model parameters Θ_0, number of iterations N, training dataset $\mathcal{D}_{\text{train}}$, validation dataset $\mathcal{D}_{\text{val}}$
Ensure: Θ_{ASWA}

1: $\Theta_{\text{ASWA}} \leftarrow \Theta_0$ ▷ Initialize Parameter Ensemble
2: $\boldsymbol{\alpha} \leftarrow \mathbf{0.0}$ ▷ Initialize N coefficients
3: $\text{val}_{\text{ASWA}} \leftarrow -1$ ▷ Initialize validation score
4: **for** $i = 0$ to N **do**
5: $\quad\Theta_{i+1} \leftarrow \Theta_i - \alpha\nabla\mathcal{L}(\Theta_i)$
6: $\quad\text{val}_\Theta \leftarrow \text{Eval}(\mathcal{D}_{\text{val}}, \Theta_{i+1})$
7: $\quad$**if** $\text{val}_\Theta > \text{val}_{\text{ASWA}}$ **then**
8: $\quad\quad\Theta_{\text{ASWA}} \leftarrow \Theta_{i+1}$ ▷ Hard Update
9: $\quad\quad\boldsymbol{\alpha} \leftarrow \mathbf{0.0}$
10: $\quad\quad\text{val}_{\text{ASWA}} \leftarrow \text{val}_\Theta$
11: $\quad\quad$**Continue**
12: $\quad$**end if**
13: $\quad\hat{\Theta}_{\text{ASWA}} \leftarrow \frac{\Theta_{\text{ASWA}} \odot (\sum_{j=i}^{N} \boldsymbol{\alpha}_j) + \Theta_{i+1}}{(\sum_{j=i}^{N} \boldsymbol{\alpha}_j) + 1}$ ▷ Look-head
14: $\quad\text{val}_{\hat{\Theta}_{\text{ASWA}}} \leftarrow \text{Eval}(\mathcal{D}_{\text{val}}, \hat{\Theta}_{\text{ASWA}})$
15: $\quad$**if** $\text{val}_{\hat{\Theta}_{\text{ASWA}}} > \text{val}_{\text{ASWA}}$ **then**
16: $\quad\quad\Theta_{\text{ASWA}} \leftarrow \hat{\Theta}_{\text{ASWA}}$ ▷ Soft Update
17: $\quad\quad\boldsymbol{\alpha}_{i+1} \leftarrow 1.0$
18: $\quad\quad\text{val}_{\text{ASWA}} \leftarrow \text{val}_{\hat{\Theta}_{\text{ASWA}}}$
19: $\quad$**else**
20: $\quad\quad$ ▷ Reject Update
21: $\quad$**end if**
22: **end for**

In Algorithm 1, we describe ASWA with hard, soft ensemble updates and rejection according to the trajectories of the validation performances and an embedding ensemble model Θ_{ASWA}. By incorporating the validation performance of the running model, hard ensemble updates can be performed, i.e., if the validation performance of the running ensemble model is greater than the validation performances of the current ensemble model and the look-head (Lines 7-12). The hard ensemble update restarts the process of maintaining the running average of parameters, whereas the soft update updates the current parameter ensemble model based on the running model (Lines 15-18). Note that **the validation performance of Θ_{ASWA} cannot be less than the validation performance of Θ**. Hence, a possible underfitting depending on N is mitigated with the expense of computing the validation performances. Importantly, **the validation performance of Θ_{ASWA} cannot be less than the validation performance of Θ_{SWA} due to the rejection criterion**, i.e., a parameter ensemble model is only updated if the validation performance is increased.

Computational Complexity. At test time, the time and memory requirements of ASWA are identical to the requirements of conventional training as well as SWA. Yet, during training, the time

overhead of ASWA is linear in the size of the validation dataset. This stems from the fact that at each epoch the validation performances are computed. Since ASWA does not introduce any hyperparameter to be tuned, ASWA can be more practical if the overall training and hyperparameter optimization phases are considered.

5 Experimental Setup

Datasets. In our experiments, we used the standard benchmark datasets for link prediction such as UMLS, KINSHIP, Countries S1, NELL-995 h50, NELL-995 h75, NELL-995 h100, FB15K-237, YAGO3-10 benchmark datasets. Apart from simple link predictions, we also consider multi-hop answering tasks where link predictors are being used. Overviews of the datasets and different query types we consider are provided in Table 1 and Table 2, respectively. Additionally, the literal extensions of FB15k-237 and YAGO15K contain 29,220 and 23,520 triples, and 116 and 7 attribute types, respectively.

Table 1: An overview of datasets in terms of the number of entities $\mathcal{E}$, number of relations $\mathcal{R}$, and the number of triples in each split of the knowledge graph datasets.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{G}^{\text{Train}}\rvert$	$\lvert\mathcal{G}^{\text{Validation}}\rvert$	$\lvert\mathcal{G}^{\text{Test}}\rvert$
Countries-S1	271	2	1,111	24	24
UMLS	135	46	5,216	652	661
KINSHIP	104	25	8,544	1,068	1,074
NELL-995-h100	22,411	43	50,314	3,763	3,746
NELL-995-h75	28,085	57	59,135	4,441	4,389
NELL-995-h50	34,667	86	72,767	5,440	5,393
WN18RR	40,943	22	86,835	3,034	3,134
YAGO15K	15,403	32	110,441	13,800	13,815
FB15K-237	14,541	237	272,115	17,535	20,466
YAGO3-10	123,182	37	1,079,040	5,000	5,000

Table 2: Overview of different query types. Query types are taken from [7].

Multi-hop Queries
2p $E_? \,.\, \exists E_1 : r_1(e, E_1) \wedge r_2(E_1, E_?)$
3p $E_? \,.\, \exists E_1 E_2 . r_1(e, E_1) \wedge r_2(E_1, E_2) \wedge r_3(E_2, E_?)$
2i $E_? \,.\, r_1(e_1, E_?) \wedge r_2(e_2, E_?)$
3i $E_? \,.\, r_1(e_1, E_?) \wedge r_2(e_2, E_?) \wedge r_3(e_3, E_?)$
ip $E_? \,.\, \exists E_1 . r_1(e_1, E_1) \wedge r_2(e_2, E_1) \wedge r_3(E_1, E_?)$
pi $E_? \,.\, \exists E_1 . r_1(e_1, E_1) \wedge r_2(E_1, E_?) \wedge r_3(e_2, E_?)$
2u $E_? \,.\, r_1(e_1, E_?) \vee r_2(e_2, E_?)$
up $E_? \,.\, \exists E_1 . [r_1(e_1, E_1) \vee r_2(e_2, E_1)] \wedge r_3(E_1, E_?)$

Training and Optimization. We followed standard experimental setups in our experiments. For the link prediction and multi-hop query answering, we followed the experimental setup used in [2, 27, 33]. We trained DistMult, ComplEx, QMult, and Keci Knowledge Graph Embedding (KGE) models with the following hyperparameter configuration: the number of epochs $N \in \{128, 256, 300\}$, Adam optimizer with $\eta = 0.1$, batch size 1024, and an embedding vector size $d = 128$. Note that $d = 128$ corresponds to 128 real-valued embedding vector size, hence 64 and 32 complex- and quaternion-valued embedding vector sizes, respectively. We ensure that all models have the same number of parameters while exploring various d. Throughout our experiments, we used the KvsAll training strategy. We applied the beam search combinatorial search to apply pre-trained aforementioned KGE models to answer multi-hop queries. More specifically, we compute query scores for entities via the beam search combinatorial optimization procedure, we keep the top 10 most promising variable-to-entity substitutions. Furthermore, we apply SWA from the starting epoch and use its best-found hyperparameters for ASWA to ensure a fair comparison, as ASWA does not rely on selecting a start point.

Similarly, for **SnapE**, we adopt the *deferred CCA* scheduling method for cyclic learning rate adjustments, combined with a *weighted averaging* of scores from an ensemble of models. The weights are assigned based on the inverse of the training loss at the time each snapshot model is captured. While the original work recommends deferring the learning rate scheduling until 80% of the training is complete, we initiate the snapshot ensembling at approximately the halfway point of training. For the **literal-augmented models**, we maintain the same embedding dimension as used in other link prediction tasks ($d = 128$). We perform a hyperparameter search over learning rates $\in \{0.1, 0.01, 0.05, 0.001\}$ and number of epochs $\in \{256, 300, 500\}$, selecting the best configuration for all model variants within each dataset. Our goal is to evaluate model performance under different ensemble strategies rather than to identify the best-performing individual model.

Evaluation. To evaluate the link prediction and multi-hop query answering performances, we used standard metrics filtered mean reciprocal rank (MRR) and Hit@1, Hit@3, and Hit@10 [27]. To evaluate the multi-hop query answering performances, we followed the complex query decomposition framework [2, 7]. Therein, a complex multi-hop query is decomposed into subqueries, where the truth value of each atom is computed by a pretrained KGE model/neural link predictor. Given a query, a prediction is obtained by ranking candidates in descending order of their aggregated scores. For each query type, we generate 500 queries to evaluate the performance. The implementation of the ASWA, along with SWA and SnapE can be found here [1].

Link Prediction Results. Tables 3 to 5 report the link prediction results on WN18RR, FB15K-237, and YAGO3-10, NELL, Countries, UMLS, and KINSHIP datasets, respectively. Overall, experimental results suggest that SWA and ASWA consistently lead to better generalization performance in all metrics than conventional training and the state-of-the-art ensemble approach SnapE in the link prediction.

More specifically, Table 3 shows that SWA and ASWA outperform the conventional training and SnapE on FB15k-237 and YAGO3-10 datasets. On WN18RR, however, we see that SnapE performs the best with DistMult and QMult. This is because during deferred training, models show stable validation performance compared to other approaches while training loss decreases, indicating overfitting. In the ensemble, later snapshots get higher weights due to lower training loss; however, no better generalization, thereby slightly boosting performance through the weighting scheme.

[1] https://github.com/dice-group/dice-embeddings

Table 3: The table presents the link prediction performance of DistMult, ComplEx, QMult, and Keci, alongside their SWA and ASWA variants, across 3 benchmark datasets: WN18RR, FB15K-237, and YAGO3-10. Each model is evaluated using Mean Reciprocal Rank (MRR) and Hits@k metrics (for $k = 1, 3, 10$). Bold values represent the best scores for each metric, indicating the top-performing model for that dataset.

	WN18RR				FB15K-237				YAGO3-10			
	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
ComplEx	0.279	0.249	0.298	0.332	0.095	0.060	0.099	0.162	0.133	0.087	0.149	0.218
+SWA	0.288	0.262	0.303	0.332	0.153	0.109	0.162	0.233	0.459	0.395	0.502	0.571
+SnapE	0.284	0.256	0.302	0.334	0.155	0.102	0.169	0.256	0.403	0.332	0.449	0.531
+ASWA	**0.351**	**0.340**	**0.358**	**0.370**	**0.192**	**0.140**	**0.207**	**0.289**	**0.470**	**0.403**	**0.515**	**0.585**
DistMult	0.281	0.265	0.290	0.309	0.092	0.062	0.095	0.149	0.131	0.097	0.146	0.192
+SWA	0.360	0.352	0.362	0.374	0.150	0.104	0.162	0.235	**0.466**	**0.399**	**0.513**	**0.582**
+SnapE	**0.363**	**0.353**	**0.365**	**0.381**	0.148	0.095	0.159	0.252	0.329	0.247	0.376	0.488
+ASWA	0.354	0.349	0.356	0.363	**0.207**	**0.152**	**0.222**	**0.313**	0.369	0.293	0.408	0.514
QMult	0.085	0.065	0.090	0.121	0.111	0.080	0.117	0.161	0.268	0.211	0.288	0.378
+SWA	0.051	0.033	0.054	0.084	0.174	0.123	0.188	0.269	0.275	0.214	0.302	0.389
+SnapE	**0.104**	**0.076**	**0.115**	**0.159**	0.151	0.108	0.155	0.221	0.304	0.245	0.329	0.411
+ASWA	0.088	0.068	0.094	0.123	**0.241**	**0.176**	**0.262**	**0.366**	**0.358**	**0.284**	**0.395**	**0.497**
Keci	0.305	0.275	0.324	0.359	0.128	0.079	0.137	0.224	0.213	0.148	0.240	0.342
+SWA	0.326	0.315	0.331	0.346	0.156	0.103	0.166	0.259	0.417	0.336	0.464	0.567
+SnapE	0.333	0.317	0.343	0.360	0.224	0.149	0.246	**0.375**	0.397	0.317	0.444	0.546
+ASWA	**0.358**	**0.352**	**0.359**	**0.368**	**0.243**	**0.177**	**0.262**	0.374	**0.459**	**0.375**	**0.511**	**0.607**

Table 4: The table reports the performance of different models, including DistMult, ComplEx, QMult, and Keci, along with their SWA and ASWA variants, on the h100, h75, and h50 of NELL-995 benchmark dataset. Each model is evaluated using Mean Reciprocal Rank (MRR) and Hits@k metrics (for $k = 1, 3, 10$). Bold values represent the best scores for each metric, indicating the top-performing model for that dataset.

	NELL-995-h100				NELL-995-h75				NELL-995-h50			
	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
ComplEx	0.192	0.135	0.212	0.308	0.192	0.137	0.210	0.301	0.195	0.142	0.214	0.300
+SWA	0.212	**0.154**	**0.236**	0.326	0.214	0.156	0.237	0.325	0.212	0.158	0.234	0.317
+SnapE	0.197	0.139	0.218	0.314	0.198	0.144	0.217	0.308	0.199	0.145	0.219	0.307
+ASWA	**0.213**	0.151	0.233	**0.334**	**0.218**	**0.158**	**0.242**	**0.335**	**0.232**	**0.172**	**0.256**	**0.347**
DistMult	0.132	0.094	0.146	0.203	0.110	0.076	0.119	0.177	0.121	0.084	0.137	0.192
+SWA	0.175	0.129	0.193	0.262	0.162	0.122	0.179	0.237	0.158	0.115	0.177	0.242
+SnapE	0.166	0.115	0.187	0.262	0.150	0.108	0.163	0.234	0.154	0.106	0.176	0.243
+ASWA	**0.238**	**0.172**	**0.265**	**0.376**	**0.229**	**0.166**	**0.257**	**0.352**	**0.239**	**0.177**	**0.268**	**0.360**
QMult	0.149	0.099	0.163	0.247	0.160	0.110	0.174	0.260	0.153	0.104	0.168	0.252
+SWA	0.168	0.114	0.186	0.271	0.176	0.124	0.195	0.279	0.162	0.114	0.176	0.260
+SnapE	0.108	0.072	0.117	0.179	0.112	0.079	0.119	0.173	0.118	0.081	0.125	0.196
+ASWA	**0.176**	**0.118**	**0.196**	**0.287**	**0.183**	**0.126**	**0.205**	**0.292**	**0.174**	**0.120**	**0.191**	**0.278**
Keci	0.155	0.109	0.167	0.250	0.150	0.107	0.159	0.237	0.173	0.121	0.188	0.278
+SWA	0.204	**0.152**	0.223	0.310	0.195	0.144	0.212	0.298	0.217	0.161	0.238	0.332
+SnapE	0.164	0.115	0.180	0.264	0.158	0.114	0.168	0.251	0.180	0.129	0.194	0.285
+ASWA	**0.212**	0.150	**0.235**	**0.341**	**0.230**	**0.165**	**0.256**	**0.357**	**0.247**	**0.184**	**0.273**	**0.377**

Table 5: The table presents the link prediction performance of various models, including DistMult, ComplEx, QMult, and Keci, along with their SWA and ASWA variants, on the Countries-S1, UMLS and KINSHIP datsets. The evaluation is conducted using Mean Reciprocal Rank (MRR) and Hits@k (for $k = 1, 3, 10$). Bold values represent the best scores for each metric, indicating the top-performing model for that dataset.

	S1				UMLS				KINSHIP			
	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10
ComplEx	**0.360**	**0.208**	0.396	**0.771**	0.650	0.514	0.741	0.920	0.568	0.398	0.683	0.908
+SWA	0.284	0.146	0.292	0.667	0.782	0.693	0.843	0.948	0.667	0.524	0.772	0.926
+SnapE	0.350	0.188	**0.417**	0.729	0.664	0.530	0.753	0.923	0.566	0.392	0.684	0.909
+ASWA	0.339	**0.208**	0.354	0.646	**0.837**	**0.757**	**0.906**	**0.974**	**0.744**	**0.616**	**0.849**	**0.962**
DistMult	0.379	0.229	0.479	0.667	0.434	0.300	0.477	0.737	0.388	0.239	0.422	0.755
+SWA	0.362	0.188	0.479	0.667	0.600	0.489	0.646	0.833	0.474	0.324	0.520	0.844
+SnapE	0.405	0.250	**0.500**	**0.708**	0.452	0.303	0.505	0.769	0.429	0.273	0.482	0.816
+ASWA	**0.416**	**0.312**	0.438	0.688	**0.727**	**0.626**	**0.784**	**0.927**	**0.501**	**0.345**	**0.563**	**0.865**
QMult	0.365	0.208	0.479	0.625	0.653	0.528	0.728	0.899	0.532	0.365	0.632	0.876
+SWA	**0.457**	**0.333**	**0.562**	0.646	0.767	0.671	0.834	0.937	0.610	0.459	0.704	0.907
+SnapE	0.346	0.167	0.479	0.667	0.664	0.542	0.735	0.904	0.533	0.368	0.628	0.880
+ASWA	0.368	0.188	0.500	**0.729**	**0.824**	**0.734**	**0.897**	**0.970**	**0.688**	**0.553**	**0.786**	**0.937**
Keci	0.268	0.125	0.292	0.604	0.551	0.412	0.613	0.836	0.495	0.313	0.607	0.872
+SWA	0.308	**0.146**	0.375	0.625	0.693	0.589	0.753	0.898	0.686	0.534	0.800	0.954
+SnapE	0.272	0.125	0.292	0.604	0.553	0.413	0.621	0.840	0.505	0.327	0.606	0.874
+ASWA	**0.318**	**0.146**	**0.396**	**0.667**	**0.830**	**0.750**	**0.894**	**0.958**	**0.713**	**0.580**	**0.811**	**0.961**

Apart from these two, for all other models, our approach ASWA shows MRR improvement over SWA in most cases. This is clearly visible on two large datasets, FB15k-237 and YAGO3-10. Specifically, on FB15K-237, ASWA shows a notable advantage, outperforming SWA and SnapE for all models. On YAGO3-10, we see a similar trend—ASWA significantly improves performance on most metrics, especially with ComplEx and Keci models. Table 4 shows results on the NELL h100, h75, and h50 datasets. Again, the weighted averaging approaches outperform the best KGE ensemble approach. Moreover, ASWA consistently leads to further performance gains across all models. Finally, Table 5 shows a similar trend for smaller datasets. Apart from the Countries-S1 dataset with the ComplEx model, all other models show performance improvement with our weighted averaging approach.

Literal Augmented Models. To evaluate the effectiveness of weighted averaging approaches across various types of link prediction tasks, we also assess their performance using literal-augmented KGE models. Among existing methods, we adopt LiteralE [19] to incorporate literal information into KGE models and evaluate them independently, as well as with SWA and ASWA. For a comprehensive evaluation, we include geometric models (ComplEx [33], DistMult [39]), a convolution-based model (ConvE [8]), and a tensor factorization model (TuckER [3]) and report the results in Table 6. The results show that the weighted averaging approaches outperform the existing models. Furthermore, we can see that there is a clear advantage of using our weighted averaging approach ASWA therein since it improves the results of the SWA for all the models.

Multi-hop Query Answering Results Table 7 presents Mean Reciprocal Rank (MRR) results on FB15k-237 for a diverse set of multi-hop logical queries. Overall, ASWA demonstrates consistent improvements across models and query types, particularly improving the performance in queries involving intersection (3i) and union (2u, up). For instance, QMult+ASWA achieves the highest MRR on 3i (0.183) and 2u (0.072), while Keci+ASWA achieves strong gains across all complex queries. While SWA generally boosts performance over the base models—especially on 3i queries—it exhibits inconsistent or even degraded performance on simpler path-based queries like 2p and 3p. This degradation is especially notable in

Table 6: The table presents the link prediction performance of various models augmented with literal values, including DistMult, ComplEx, ConvE, along with their SWA and ASWA variants. Bold values represent the best scores for each metric, indicating the top-performing model for that dataset.

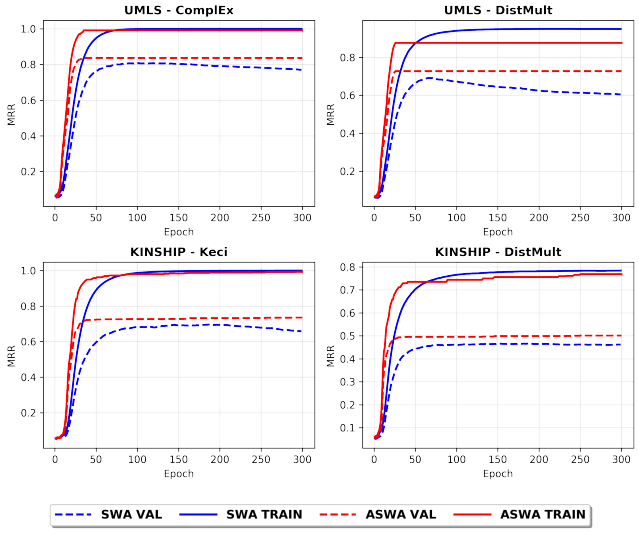
	FB15k-237 + Literal				YAGO15K + Literal			
	MRR	@1	@3	@10	MRR	@1	@3	@10
ComplEx	0.261	0.182	0.284	0.418	0.309	0.235	0.339	0.452
+SWA	0.270	0.189	0.293	0.432	0.263	0.177	0.299	0.429
+ASWA	**0.291**	**0.209**	**0.317**	**0.454**	**0.412**	**0.345**	**0.448**	**0.534**
ConvE	0.290	0.207	0.316	0.456	0.281	0.200	0.309	0.442
+SWA	0.279	0.197	0.305	0.441	0.217	0.141	0.236	0.365
+ASWA	**0.299**	**0.215**	**0.326**	**0.465**	**0.294**	**0.213**	**0.324**	**0.453**
DistMult	0.253	0.176	0.277	0.407	0.304	0.229	0.338	0.449
+SWA	0.264	0.188	0.285	0.419	0.268	0.187	0.302	0.426
+ASWA	**0.285**	**0.206**	**0.307**	**0.445**	**0.398**	**0.331**	**0.432**	**0.526**
TuckER	0.321	0.234	0.350	0.494	0.196	0.126	0.209	0.333
+SWA	0.270	0.193	0.293	0.421	0.139	0.081	0.143	0.249
+ASWA	**0.337**	**0.250**	**0.368**	**0.510**	**0.203**	**0.132**	**0.219**	**0.343**

DistMult and ComplEx, where MRR drops significantly (e.g., 2p drops from 0.007 to 0.002 in ComplEx+SWA). These observations highlight the importance of adaptive or selective model averaging strategies—as implemented in ASWA.

Table 7: Multi-hop query MRR results on FB15k-237.

	2p	3p	3i	ip	pi	2u	up
DistMult	**0.007**	**0.007**	0.044	0.003	0.097	0.020	0.006
+SWA	0.003	0.002	0.106	0.004	**0.098**	0.031	**0.007**
+ASWA	0.002	0.003	**0.175**	**0.005**	0.092	**0.055**	0.002
ComplEx	**0.009**	0.001	0.036	0.003	0.092	0.014	**0.006**
+SWA	0.002	0.002	0.136	**0.005**	**0.105**	0.030	0.005
+ASWA	0.002	**0.003**	**0.155**	**0.005**	0.100	**0.049**	0.001
QMult	0.002	0.003	0.034	0.000	0.099	0.014	0.006
+SWA	0.003	0.002	0.089	0.001	0.092	0.027	**0.007**
+ASWA	**0.004**	**0.005**	**0.183**	**0.002**	**0.117**	**0.072**	0.003
Keci	**0.009**	0.005	0.052	0.002	0.085	0.027	0.013
+SWA	0.007	0.005	0.101	0.002	0.097	0.059	**0.014**
+ASWA	0.004	0.005	**0.165**	**0.007**	**0.106**	**0.064**	0.004

Figure 1: Training and validation MRR scores of SWA and ASWA across epochs for different models on UMLS and KINSHIP datasets.



6 Discussion

Our evaluations indicate that weight averaging approaches perform better than conventional training (e.g. finding model parameters with Adam optimizer); moreover, ASWA consistently generalizes better than SWA across models and datasets. For instance, on YAGO3-10 (see Table 3), SWA finds model parameters leading to higher link prediction performance on the training data in all metrics. Given that KGE models have the same number of parameters (entities and relations are represented with d number of real numbers), this superior performance of SWA over the conventional training can be explained as the mitigation of the noisy parameter updates around minima. More specifically, maintaining a running unweighted average of parameters becomes particularly useful around a minima by means of reducing the noise in the gradients of loss w.r.t. parameters that is caused by the mini-batch training. ASWA, however, renders itself as an effective combination of SWA with early stopping techniques, where the former accepts all parameter updates on a parameter ensemble model based on a running model and the former rejects parameter updates on a running model in the presence of overfitting. Since the ASWA does not update a parameter ensemble model if a running model begins to overfit, it acts as a regularization on a parameter ensemble. This regularization impact leads to a better generalization in all metrics.

Broadly, our experimental results corroborate the findings of [18]: constructing a parameter ensemble by maintaining a running average of parameters at each epoch improves generalization across a wide range of datasets and models. Link prediction and multi-hop query answering results show that SWA and ASWA find better solutions than conventional training based on ADAM and SGD. Our results also show that updating the parameter ensemble uniformly at each epoch leads to suboptimal results as the model begins to overfit. This is clearly seen in Figure 1, which shows the validation MRR comparing SWA and ASWA as training progresses. The results suggest that ASWA effectively mitigates overfitting by rejecting parameter updates once the model begins to overfit. Thus, it serves as an effective combination of SWA and early stopping—SWA accepts all parameter updates, while early stopping rejects updates when overfitting occurs. Importantly, ASWA does not require a starting epoch to begin constructing the parameter ensemble. Instead, it performs a hard update on the ensemble model whenever the running model outperforms it on the validation dataset.

7 Conclusion

In this work, we investigated techniques to construct a high performing ensemble model while alleviating the overhead of training multiple models, and retaining efficient memory and inference requirements at test time. To this end, we introduce the weighted averaging technique, SWA and furthermore propose an adaptive stochastic weight averaging technique that effectively combines the SWA technique with early stopping. Essentially. ASWA extends SWA by building a parameter ensemble according to an adaptive schema governed by the generalization trajectory on the validation dataset. Our extensive experiments across benchmark datasets—spanning link prediction, literal-augmented models, and multi-hop query answering—demonstrate that weighted averaging techniques consistently improve the generalization performance of strong baseline models. Moreover, we observe that ASWA is more effective than SWA in mitigating overfitting. As future work, we aim to explore task-aware scheduling strategies for averaging that dynamically adjust based on query complexity or domain characteristics. Additionally, integrating uncertainty estimation into the averaging process could further enhance robustness and calibration considering noisy or adversarial settings.

Acknowledgments

This work is supported by the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within the project SAIL under the grant no NW21-059D, the project WHALE (LFN 1-04)

funded under the Lamarr Fellow Network Programme by the MKW NRW, the European Union's Horizon Europe research and innovation programme under grant agreement No 101070305, and by the German Federal Ministry of Research, Technology and Space (BMFTR) within the project KI-OWL under the grant no 01IS24057B.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning. In *The Eleventh International Conference on Learning Representations*.

[2] Erik Arakelyan, Daniel Daza, Pasquale Minervini, and Michael Cochez. 2021. Complex Query Answering with Neural Link Predictors. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=Mos9F9kDwkz

[3] Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 5184–5193.

[4] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*. Springer, 553–565.

[5] Pierre Baldi and Peter J Sadowski. 2013. Understanding dropout. *Advances in neural information processing systems* 26 (2013).

[6] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24 (1996), 123–140.

[7] Caglar Demir, Michel Wiebesiek, Renzhong Lu, Axel-Cyrille Ngonga Ngomo, and Stefan Heindorf. 2023. LitCQD: Multi-hop Reasoning in Incomplete Knowledge Graphs with Numeric Literals. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 617–633.

[8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[9] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[10] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. 2018. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*. PMLR, 1309–1318.

[11] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. PMLR, 1050–1059.

[12] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems* 31 (2018).

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[14] Cosimo Gregucci, Mojtaba Nayyeri, Daniel Hernández, and Steffen Staab. 2023. Link Prediction with Attention Applied on Multiple Knowledge Graph Embedding Models. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, 2600–2610. doi:10.1145/3543507.3583358

[15] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[16] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–37.

[17] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot Ensembles: Train 1, Get M for Free. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJYwwY9ll

[18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407* (2018).

[19] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. 2019. Incorporating literals into knowledge graph embeddings. In *International Semantic Web Conference*. Springer, 347–363.

[20] Denis Krompaß and Volker Tresp. 2015. Ensemble Solutions for Link-Prediction in Knowledge Graphs. In *Proceedings of the 2nd Workshop on Linked Data for Knowledge Discovery (in conjunction with ECML/PKDD)*.

[21] Shiwei Liu, Tianlong Chen, Zahra Atashgahi, Xiaohan Chen, Ghada Sokar, Elena Mocanu, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. 2022. Deep Ensembling with No Overhead for either Training or Testing: The All-Round Blessings of Dynamic Sparsity. In *International Conference on Learning Representations*. https://openreview.net/forum?id=RLtqs6pzj1-

[22] Kevin P Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press.

[23] Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization* 30, 4 (1992), 838–855.

[24] Vignesh Prabhakar, Chau Vu, Jennifer Crawford, Joseph Waite, and Kai Liu. 2023. An ensemble learning approach to perform link prediction on large scale biomedical knowledge graphs for drug repurposing and discovery. *bioRxiv* (2023), 2023–03.

[25] Lutz Prechelt. 2002. Early stopping-but when? In *Neural Networks: Tricks of the trade*. Springer, 55–69.

[26] Daniel Rivas-Barragan, Daniel Domingo-Fernández, Yojana Gadiya, and David Healey. 2022. Ensembles of knowledge graph embedding models improve predictions for drug discovery. *Briefings in Bioinformatics* 23, 6 (2022), bbac481.

[27] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You can teach an old dog new tricks! on training knowledge graph embeddings. *International Conference on Learning Representations* (2020). https://openreview.net/forum?id=BkxSmlBFvr

[28] Omer Sagi and Lior Rokach. 2018. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 4 (2018), e1249.

[29] Ali Shaban and Heiko Paulheim. 2024. SnapE - Training Snapshot Ensembles of Link Prediction Models. In *The Semantic Web - ISWC 2024 - 23rd International Semantic Web Conference, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 15231)*, Gianluca Demartini, Katja Hose, Maribel Acosta, Matteo Palmonari, Gong Cheng, Hala Skaf-Molli, Nicolas Ferranti, Daniel Hernández, and Aidan Hogan (Eds.). Springer, 3–22. doi:10.1007/978-3-031-77844-5_1

[30] Arnab Sharma, N'Dah Jean Kouagou, and Axel-Cyrille Ngonga Ngomo. 2024. Resilience in Knowledge Graph Embeddings. *CoRR* abs/2410.21163 (2024). doi:10.48550/ARXIV.2410.21163 arXiv:2410.21163

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[32] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016 (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). JMLR.org, 2071–2080. http://proceedings.mlr.press/v48/trouillon16.html

[33] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*. PMLR, 2071–2080.

[34] Guojia Wan, Bo Du, Shirui Pan, and Jia Wu. 2020. Adaptive knowledge subgraph ensemble for robust and trustworthy knowledge graph completion. *World Wide Web* 23, 1 (2020), 471–490. doi:10.1007/S11280-019-00711-Y

[35] David Warde-Farley, Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. 2013. An empirical analysis of dropout in piecewise linear networks. *arXiv preprint arXiv:1312.6197* (2013).

[36] Jingjing Xie, Bing Xu, and Zhang Chuang. 2013. Horizontal and vertical ensemble with deep representation for classification. *arXiv preprint arXiv:1306.2759* (2013).

[37] Chengjin Xu, Mojtaba Nayyeri, Sahar Vahdati, and Jens Lehmann. 2021. Multiple Run Ensemble Learning with Low-Dimensional Knowledge Graph Embeddings. In *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN) / CEUR Workshop*.

[38] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6575

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

[40] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion Knowledge Graph Embeddings. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 2731–2741. https://proceedings.neurips.cc/paper/2019/hash/d961e9f236177d65d21100592edb0769-Abstract.html