RepeatE: repetitive prediction and Center Loss for CNN based knowledge graph embedding

Abstract

Knowledge graph embedding is a very important task, and it is related to many downstream tasks, such as recommendation system. Inspired By ConvE, there are more and more 2D CNN based knwoledge graph embedding methods, such as ConvR, InteractE and AcrE. The methods first reshape the relation embeddings and entity embeddings to 2D matrix and then use CNN to predict the results. Because of the Dropout operation, we find that the 2D CNN based methods will give different predictions through given the same input. To make the model be more robust, we apply repetitive prediction and center loss. For repetitive prediction, we repeatedly use the Convolutional Neural Network to calculates K times for a input and get K different predictions. For center loss, we first calculate the average distance between the predictions and their center. We use InteractE as our baseline. We follow the same evaluation protocol with InteractE and evaluate our model on three datasets. We also discover the impact of repetitive prediction and center loss. Extensive experiments show that our model is very **effective and fast coverage**.

1 Introduction

Knowledge graph embedding is an important topic because it contains the relationship between different objects. Knowledge Graphs(KGs) are the special representation of facts, and it can be represented as a collection of facts (s,r,o). The "r" is the relation between the "subject entity" s and the "object entity" o. There are many tasks for the knowledge graph, such as relation extraction and recommendation system. For the paper, we focus on link prediction. Given one " subject entity " and one "relation", we try to predict the "object entity". There are two steps. One is the embedding part which embeds the object and relation. Another is the score function part, which gives a higher score to the real facts and a lower score to false facts. The existing works mainly focusing on the second part and try to find a better score function. For the score function, there are two main directions. One is translation based method, and another is the machine learning based method. For the translation based method, they map the entities and relations to spaces and calculate the distance. The smaller the distance is, the more possible that the facts are true. However, the translation based method is limited by the expressiveness. To improve it, the only way is to increase the embedding size, which will increase the number of parameters and the calculation cost.

For the machine learning based method, it mainly uses Convolutional Neural Network(CNNs) which has strong expressiveness. ConvE first applies 2D CNNs to the knowledge graph embedding tasks. Then, ConvR points out that we need more interaction for CNN based knowledge graph embedding. InteractE design a special combination method named Checkered Reshaping for "subject entity" embedding and "relation" embedding. First, the InteractE permutate the "subject entity" embedding and "relation" embedding separately. Then it makes sure that the adjacent cell components come from different embedding. Therefore, "subject embedding" and "relation" embedding can interact more.

In the paper, we point out that convolutional based model prediction is not stable because they have to use the Dropout method. That is, given the same subject embedding and relation embedding, the outputs are not the same. For example, the convolution based model processes the input and the output prediction is p_1.Then the model processes the input again and the output prediction is p_2. Because of Dropout operation, the two predictions p_1 and P_2 are not the same. They should be the close to each. If the Dropout operation abandons important points, the relation embedding may just become another relation embedding, which will make the prediction bad because the loss will be much higher than the normal. We can call this an outlier training sample because it drops some important points and gives an extremely high loss. To solve the above problems, we simply give two operations: repetitive prediction and center loss.

To reduce the impact of outlier made by dropping important points, we repeatedly calculate the prediction. We note the repetition number as k. For a normal model, the k is just one, which means we use the model to calculate the prediction once to get the result P_1 and calculate the loss. For our method, the k is an integer and larger than 1, which means we use the model to calculate the prediction more than one time and get the result p_1, p_2... p_k. Then, we use the prediction to calculate the average loss of the k predictions.

To make the prediction close to each other, we design the center loss. For the all prediction p_1, $p_2...p_k$, we first calculate the average value of them and call the result as the center. Then, we calculate the distance between predictions and the center. Finally, we calculate the average. To summarize, we do the following contributions:

1 We point out that convolutional based method output is not stable and may be an outlier because we may drop important points,

2 To reduce the impact out outlier and make the loss smooth, we use the repetitive prediction.

3 To make the predictions from the same entity and relation are close to each other, we apply the center loss.

4 Through extensive experiments, we prove that repeating sample and center loss are all useful.

2 Related Work

Non Neural

The non-neural based methods usually design a score function for triples by simple math operation, including addition, subtraction, dot production and matrix multiplication. The non-neural based methods first embed the one-hot vector to a continuous vector. Then, they design a score function to calculate the score for each triple. They Do Not Use the Dropout method.

TransE(Bordes *et al.* [2013]) first focuses on knowledge graph embedding by distance based method. The score function of TransE is $|e_s + e_r - e_o|$ while e_s, e_r and e_o are subject embedding, relation embedding and object embedding respectively. Then, there are many following work, such as TransH(Wang *et al.* [2014]), TransR(Lin *et al.* [2015]) and TransH(Ji *et al.* [2015]). To better capture the symmetric relation, DistMult(Yang *et al.* [2014]) is proposed, which use multiplication to calculate the score function $|e_s * e_r * e_o|$. ComplexE(Trouillon *et al.* [2016]) points out that one embedding can be divided into two parts, including real part and complex part. Following the idea, there are RotatE(Sun *et al.* [2019]) model.

Neural Network Based

In recent, the neural network is used more and more widely in the knowledge graph. There are two main directions: build score function and use for embedding. For the neural network based method, the models usually have many parameters. Therefore, to avoid the overfitting problem, the models have to add the Dropout operation which drops some points in the feature maps or feature vectors.

For building score function, these are related work. In Neural Tensor Network, a relation specific tensor combine the entity embedding and the relation embedding, and it uses the two embeddings to computer a score. Following the idea, other works consider how to combine and build a better neural network based score function, such as using multiple inception layers.

For embedding by neural network,the related work are the following.

R-GCN(Schlichtkrull *et al.* [2018]) regards the edge as relation and give different different relation the different weight. Then, Conv-TransE(Shang *et al.* [2019]) use graph neural network for embedding and leaverage the TransE for the score function. After that, CompGCN(Vashishth *et al.* [2020b]) points out that a single relation should also be a node in a graph, just similar to an entity. It proposes a method to jointly embed the entity and the relation, which should be the state-of-the-art model in knowledge graph embedding by graph neural network.

ConvE(Dettmers *et al.* [2018]) proposes the 2D Convolution Neural Network on knowledge graph embedding. It reshape and concate the subject and relation embedding. It uses CNNs to embeeding the concatenation and get the tail object projection. ConvR(Jiang *et al.* [2019]) points that the subject and relation embedding needs more interaction. It use different CNNs to process subject embedding and relation embedding, and then it regards the relation embedding as a kernel to process the subject embedding. Then, InteractE(Vashishth *et al.* [2020a]) uses feature permutation and checkered reshaping to increase the interaction between subject embedding and relation embedding.

3 Model

As we said above, the prediction of the neural network based models are not stable and may have outlier prediction because of the Dropout(Srivastava *et al.* [2014]) operation. When training, models use Dropout operation. And when testing, models do not use Dropout operation. Previous works use Dropout to avoid overfitting and get better performance. However, Dropout have to train more epochs. To solve the problem, we add the repetitive prediction and center loss. For the repetitive prediction, given the repetition number K, we repeat K times to use the model to calculate the prediction and get p_1, $p_2...p_k$. All the predictions should have the same labels because the input is totally the same. Then, we calculate the average binary cross entropy loss of the prediction. This repetitive prediction will make the loss smooth, which will accelerate convergence and get better performance. For center loss, we push each prediction close to each other, which adds one more constraint and helps to get better performance.

Notation

Given a triple (s, r, o) while, we have the corresponding embedding (e_s,e_r,e_o) while e_s,e_r,$e_o \in R^d$ while d is the embedding dimension. For each training step, we have $X \in R^{d1xd2}$ combined by [e_s,e_r] while d1 and d2 are the shape of the reshape matrix . Given the Convolutional Neural Network(CNNs) and the input X, we get the prediction $p \in \mathrm{R}^{|\varepsilon|}$ while $|\varepsilon|$ is number of entities.

Repetitive Prediction

The repetitive prediction method can be used for any CNN based models, including ConvE, ConvR, and InteractE, The only requirement is: "Same Input, But different output", such as using Dropout.

ConvE reshapes the (e_s, e_r) to 2D matrix. To avoid overfitting, it uses dropout. Its following work ConvR, InteractE apply the similar process, including reshaping embeddings to 2D and leveraging Dropout.

Though the model can embed and extract features well, it may give a high loss and penalty because we drop down some im-

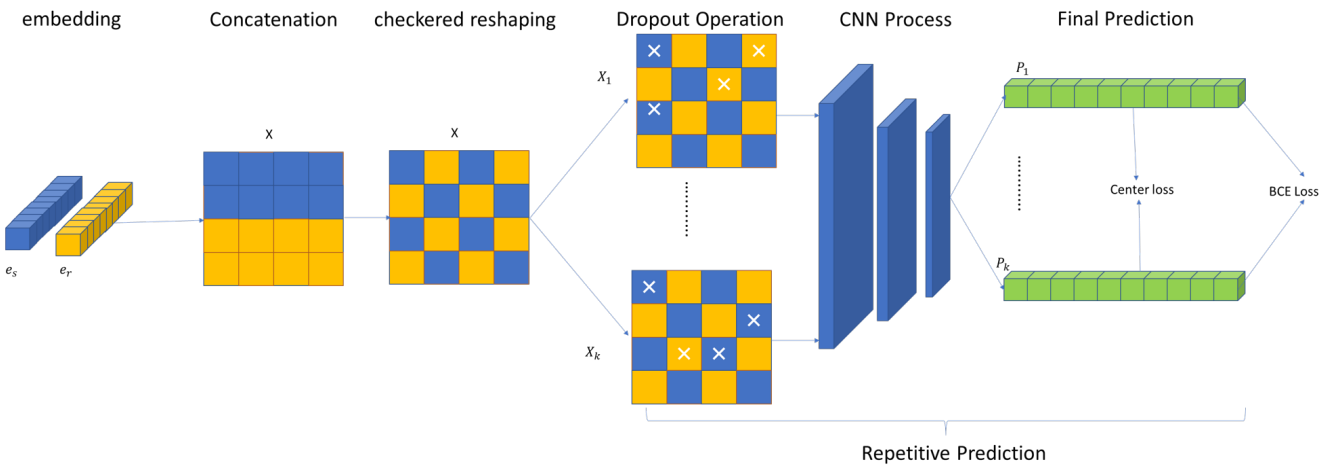

Figure 1: Overview of Repetitive calculation and center loss. **Concatenation**: reshape e_s and e_o to 2D matrix and then concatenate them together. **Checkered Reshapeing**: to be sure that two adjunct pixels come from different embedding. If one comes from the entity embedding, another must comes from the relation embedding (Vashishth *et al.* [2020a]). **Dropout Operation**: repeatedly use Dropout operation on X to get different X_1, X_2...X_K. They are not totally the same because Dropout operation may delete different points. **CNN process**: Use CNNs to process the X_1, X_2...X_K. **Final Prediction**: get the prediction P_1,P_2...P_K which are not totally the same. That is, given the same e_s and e_o, the predictions not totally the same. **Repetitive Prediction**: Repeatedly use Dropout On the input X to get different X_i and use CNN to get the prediction P_i. If the repetition number K is one, our model will just become the baseline InteractE(Vashishth *et al.* [2020a])

portant points. Therefore, the same input will cause different outputs when training.

We can repeatedly predict the results and calculate the average loss to make the loss smooth, and we can reduce the impact of the outlier which has high loss because of dropping down important points.

Assume we the parameter K is the repetition number indicating how many times of repetition, we can calculate using the repetition by the following operations.

Algorithm 1 repetitive prediction

Require:
The Convolutional Neural Network CNN; Subject Embedding e_s; Relation Embedding e_o; The Repetition Number K //For ConvE and InteractE, the K is just 1;

Ensure:
repetitive prediction Prediction P; Center Vector Of repetitive prediction Prediction $Center$;

1: **while** $K \neq 0$ **do**
2: $tempResult \Leftarrow CNN(Dropout[e_s, e_o])$ //same input, different output because of Dropout
3: $P.append(tempResult)$ // Store the prediction to P
4: $Center \Leftarrow Center + tempResult$
5: $K \Leftarrow K - 1$ //minus 1 until K is 0
6: **end while**
7: $Center \Leftarrow Center/K$ //calculate the center vector of predictions
8: **return** P,$Center$;

For repetition, first we uses Dropout operation on the [e_s,e_o] and then give to the CNN. After that, we store the prediction CNN(Dropout[e_s,e_o]) to the prediction array. We repeat the operation for K times. Because CNN based methods use Dropout, each prediction is not the same. After that, we calculate the center vector of all predictions, which will be used for center loss calculation.

Our binary classification loss is a little different from the ConvE and InteractE. Given N triples, we can get $N * K$ predictions after repetitive prediction.

For ConvE and InteractE, the repetition number is just one. Their binary classification loss is the following:

$$\mathcal{L}_{bce-ori} = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(p_i) + (1-y_i) log(1-p_i)) \quad (1)$$

.

N is the batch size. The $p_i \in R^{|\varepsilon|}$ is the prediction of i^{th} input (e_s^i,e_o^i). y_i is a label vector of p_i where $1 <= j <= K$. And $y_i \in R^{|\varepsilon|}$ is the label; the elements of vector y_i are ones for relationships that exists and zero otherwise.

As we use repetitive prediction, our binary classification loss is the following.

$$\mathcal{L}_{bce} = -\frac{1}{N}\frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{k}(y_i log(p_{ij}) + (1-y_i) log(1-p_{ij})) \quad (2)$$

N is the batch size. K is the repetition number. The $p_{ij} \in R^{|\varepsilon|}$ is the j^{th} repetition prediction of the i^{th} embeddings [e_s^i,e_r^i]. As the input is the same i^{th} embeddings (e_s^i,e_r^i), The prediction p_{i1}, p_{i2}...p_{iK} should have the same label $y_i \in R^{|\varepsilon|}$.

Compared to ConvE and InteractE, the average binary cross entropy reduces the instability. Because of Dropout operations, CNN based knowledge graph embedding models are not stable. Some loss may be smaller, and others may be larger because the model drops different points when training. As we calculate the average, the loss will be smooth, it

will increase the robustness of the model.

Center Loss

Given the same $[e_s^i,\ e_o^i]$, the repetitive predictions $p_{i1}, p_{i2}...p_{iK}$ should be close to each other because the input is the same $[e_s^i,\ e_o^i]$. If we only has the label constraint which is Equation 2, it is not sure that each predictions are the same with each other. Therefore, compared to the ConvE and InteractE, we add one more constraint named center loss, which try to push the $p_{i1}, p_{i2}...p_{iK}$ close to each other.

After CNNs process, we can get the repetitive prediction predictions and the corresponding center vectors by repetitive prediction. Then, for the i^{th} triple $[e_s^i,e_o^i]$, we calculate the distance between each simple prediction and the center vector $center_i$. Finally, we calculate the average distance for the total N triples and use it as the center loss. The center loss can measure how repetitive predictions are close to each other. The calculation way is the following.

$$\mathcal{L}_{center} = \frac{1}{N}\frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{k}(|p_{ij} - center_i|) \tag{3}$$

The $center_i \in R^{|\varepsilon|}$ is the average value of $p_{i1}, p_{i2}...p_{iK}$, which is calculated in the repetitive prediction Algorithm. In our paper,to simple implement our ideas, we measure the distance by the Manhattan distance.

The center loss tries to force $p_{i1}, p_{i2}...p_{iK}$ close to each other because the input is the same and the label is the same.

Joint Training

The final target of ConvE and InteractE just tries to minimize the Equation (1).

After adding repetitive prediction and center loss, we try to minimize both Equation (2) and Equation (3) the following.

$$\mathcal{L}_{total} = \mathcal{L}_{bce} + \lambda * \mathcal{L}_{center} \tag{4}$$

We use the λ to control the weight of $\mathcal{L}_{center}$. Usually, setting λ to be 1 is good enough. If the λ value is too high such as ten, the model performance will be very bad.

If given the same input and the model always gives the same output, $p_{i1}, p_{i2}...p_{iK}$ will be totally the same. Therefore, the center loss will just be zero. And Equation (2) will become Equation (1). Therefore, Equation 4 will just be Equation 1.

Therefore, after using our methods, the performance should no worse than the original because our loss function is a general case. The details will be discussed in the Experiment part.

Training and Inference Time

We denote that T_k is the training time when the repetition number is the k. With the increase of repetition number k, the T_k is increase non-linear. For example, T_2 only needs more 30% training time compared to T_1 on a single GPU. If we can parallel calculate each prediction, the training time will be close to T_1 whatever the repetition number k is.

For the inference part, we just calculate the result one time. When inference, the model will not drop out any points. Therefore, when we evaluate the model on the validation set or test dataset, we just calculate once. Also, we do not increase the number of parameters. Given a model M(such as ConvE, InteractE) and an input X, we only repeatedly use the model to process the input X and calculate loss by Equation 4 when training. Therefore, adding repetitive prediction and center loss only increase training time, but not increase inference time and the number of parameters.

Dataset	$\|\varepsilon\|$	$\|R\|$	Train	Valid	Test
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134
YAGO3-10	123,182	37	1,079,040	5,000	5,000

Table 1: Statistics Of three datasets

4 Experiment

In the experiment part, we will try to answer the following questions.

Q1.How does RepeatE perform in comparison to the existing approaches?

Q2.What is the effect of repeating sample and center loss on link prediction performance?

Q3.How does the performance of our model vary with the number of repetitive prediction and center loss weight?

Dataset

In our dataset experiments, we follow the InteractE(Vashishth *et al.* [2020a]) experiments. We evaluate our method on three most commonly datasets: FB15k-237, WN18RR and YAGO3-10. The introduction to the three datasets is the following. We show the corresponding statistics in the Table 1.

* FB15K-237 (**?**) is the subset of the FB15K (Bordes *et al.* [2013]) which deletes the inversion relationship to prevent direct inference of test triplets by removing training triples.
* WN18RR(Dettmers *et al.* [2018]) is the subject of WN18(Bordes *et al.* [2013]) got from WordNet(Miller [1995]), and WN18RR also deletes the inversion relationship.
* YAGO3-10 is the subset of YAGO3(Suchanek *et al.* [2007]. It totally has 37 relations, and each entities with more than 10 relations.

Evaluation protocol

To have a fair comparision, we use the same evaluation protocol with Bordes *et al.* [2013]. Filtered setting is used, which means we filter out all the valid triples in the test triples. The performance is reported on the standard evaluation matrics: Mean Re-ciprocal Rank(MRR), Hits@1 and Hits@10.Similar to the InteractE, we also run five times and report the averages result. As the variance is almost 0, and we hence ignore the variance which is the same with InteractE(Vashishth *et al.* [2020a]).

Baselines In our experiment, we compare our model with many other baselines, and they can be classified as: Non Neural Based: Methods that does not use Convolutional Neural Network, including DistMult(Yang *et al.* [2014]), ComplEx(Trouillon *et al.* [2016]), KBGAN(Cai and Wang [2017]), and Rotate(Sun *et al.* [2019]). The non-neural based methods calculate the score by the vector based operations, such as minus and multiplication. CNN-based: Methods that use Convlutional Neural Network to predict the score. For

Method	FB15k-237			WN18RR			YAGO3-10		
	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
DistMult(Yang *et al.* [2014])	24.1%	15.5%	41.9%	43%	39%	49%	34%	24%	54%
ComplEx(Trouillon *et al.* [2016])	24.7%	15.8%	42.8%	44%	41%	51%	36%	26%	55%
R-GCN(Schlichtkrull *et al.* [2018])	-	15.1%	41.7%	-	-	-	-	-	-
KBGAN(Cai and Wang [2017])	27.8%	-	45.8%	21.4%	-	47.2%	-	-	-
ConvE-TransE(Shang *et al.* [2019])	33%	24%	51%	46%	43%	52%	-	-	-
RotatE(Sun *et al.* [2019])	33.8%	24.1%	53.3%	47.6%	42.8%	**57.1%**	49.5%	40.2%	67%
TuckER(Balažević *et al.* [2019b])	35.8%	26.6%	54.4%	47.0%	44.3%	52.6%	-	-	-
LowFER(Amin *et al.* [2020])	35.9%	26.6%	54.4%	46.5%	43.4%	52.6%	-	-	-
CompGCN(Vashishth *et al.* [2020b])	35.5%	26.4%	53.5%	**47.9%**	44.3%	54.6%	-	-	-
ConvE(Dettmers *et al.* [2018])	32.5%	23.7%	50.1%	43%	40%	52%	52%	45%	66%
HypER(Balažević *et al.* [2019a])	34.1%	25.2%	52.0%	46.5%	43.6%	52.2%	53.3%	45.5%	67.8%
ConvR(Jiang *et al.* [2019])	35%	26.1%	52.8%	47.5%	44.3%	53.7%	-	-	-
AcrE(Ren *et al.* [2020])	35.8%	26.6%	54.5%	46.3%	42.9%	53.4%	-	-	-
InteractE(Vashishth *et al.* [2020a], *baseline*)	35.4%	26.3%	53.5%	46.3%	43%	52.8%	54.1%	46.2%	68.7%
ours	**36.3%**	**26.9%**	**54.9%**	47.8%	**44.6%**	54.0%	**55.8%**	**48.5%**	**69.3%**

Table 2: The link prediction result on the three datasets FB15k-237, WN18RR, and YAGO3-10. The **bold** is the best, and underline is the second. Compared to the baseline InteractE, our model outperform on all the three datasets. Compared to other state-of-the-art model, our model is the best in 7 out of 10

Method	FB15k-237	WN18RR	YAGO3-10
baseline(InteractE)	35.4%	46.3%	54.1%
baseline Remove Dropout	32.0%	45.1%	48.4%
Baseline+R	35.5%	47.5%	**55.8%**
Baseline+R+C	**36.3%**	**47.8%**	**55.8%**

Table 3: The ablation study of repetitive prediction and center loss. "Baseline No Dropout" means that the baseline removes Dropout operation. The evaluation metrics is MRR. "R" is the repetitive prediction. "C" is the center loss.

example, there are R-GCN(Schlichtkrull *et al.* [2018]), ConvTransE(Shang *et al.* [2019]), SACN(Shang *et al.* [2019]), HypER(Balažević *et al.* [2019a]), TuckER(Balažević *et al.* [2019b]), CompGCN(Vashishth *et al.* [2020b]), ConvEDettmers *et al.* [2018], ConvRJiang *et al.* [2019], AcrE(Ren *et al.* [2020]) and InteractEVashishth *et al.* [2020a].

Compared With Other Model

In this part, we will compare with our model with other state-of-the-art models which are list in the above. We show the results in the Table 2. We report the baseline scores from the corresponding paers(Yang *et al.* [2014],Trouillon *et al.* [2016],Cai and Wang [2017],

The following are the different models performance on the three datasets. Sun *et al.* [2019],Schlichtkrull *et al.* [2018],Shang *et al.* [2019],Shang *et al.* [2019],Vashishth *et al.* [2020b],Dettmers *et al.* [2018]).

First, it is easy to find that our model outperforms the baseline InteractE all on the three datasets. On the FB15k-237 dataset, our model increase 0.9%, 0.6% and 1.4% on MRR, Hit@1 and Hit@10 respectively. On the WN18RR, our model increase 1.5%, 1.6% and 1.2%. Also, on YAGO3-10, our model increase 1.7%, 2.3% and 0.6%. Therefore, we have the confidence to say that our model is better than the baseline InteractE.

Then, compared to other models, our model gets the best performance in 7 out of 10. On WN18RR, our performance also gets second place on MRR. Compared to WN18RR and YAGO3-10, the FB15K-237 dataset is more difficult. As we achieve all the best performance on FB15k-237, we believe that our model is suitable for complex datasets.

Effect of Repeating Sampling And Center loss

In this part, we will discuss the effect of repetitive prediction and center loss on link prediction. Compared to InteractE, our model has two additional processes: repetitive calculation and center loss. In the following, we show the effect of repetitive calculation and center loss.

On the difficult dataset FB15k-237, it is easy to find that the center loss contributes the most. The center loss is about 10^{-4}. After adding repetitive prediction, the performance increase about 0.1%. After adding center loss, the performance increase about 0.7%.

Then, on the WN18RR and YAGO3-10 dataset, the repetitive prediction contributes the most. The center loss is about 10^{-5}. On WN18RR, the performance increase 1.4% MRR after adding repetitive prediction, and adding center loss improves 0.3% MRR. On YAGO3-10, adding repetitive prediction increase about 1.7%. And adding center loss does not significantly increase the performance.

To discover why center loss contributes more on FB15k-237 but few on the other two datasets, We show the relation between center loss value and the training epoch by Figure 2. For FB15k-237, it always has the largest center loss. And for

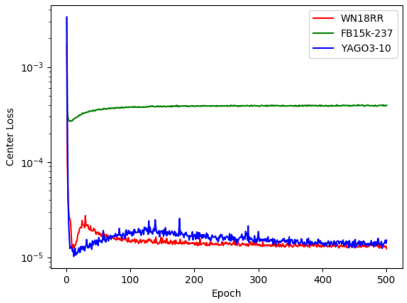

Figure 2: The center loss of each dataset.

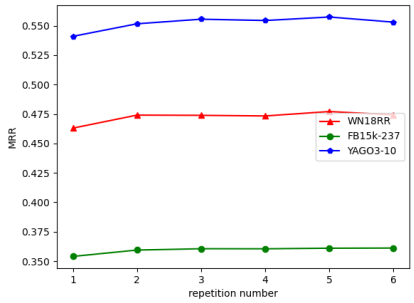

Figure 3: The Impact of repetition number. Center loss weight is 1

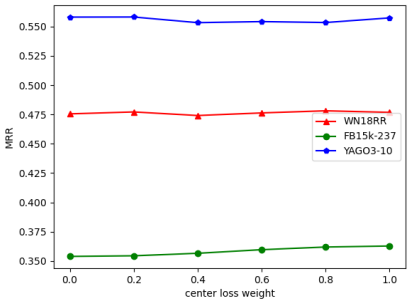

Figure 4: The Impact of center loss weight.

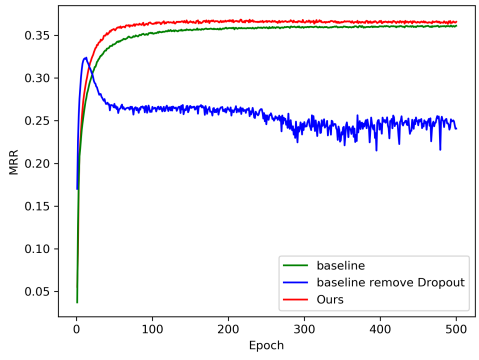

Figure 5: Validation MRR on FB15k-237. Compared to the baseline, our model is effective and fast coverage

WN18RR and YAGO3-10 dataset, the center loss is much smaller.

Is it because the center loss on WN18RR and YAGO3-10 dataset is too small so that center loss does not work so well? We simply change the center loss weight to 10 so that the center loss of WN18RR will also change from 10^{-5} to 10^{-4}. However, we find that the MRR on WN18RR is only 5.2% after running 500 epochs.

Therefore, we can have a summary. When center loss is low, the improvement mainly comes from repetitive predictions. And when the center loss is high, the improvement mainly comes from center loss.

Influence of the repetition number and center loss weight

If the repetition number is 1 and center loss weight is 0, our model will become the baseline InteractE.

The repetition number and the center loss weight have an impact on the performance of our model. To discover how they change the performance, we try different repetition numbers and center loss weight on the WN18RR, Fb15k-237 and YAGO3-10. We keep other parameters being the same with the InteractE but just change the repetition number and the center loss weight. The analysis is the following.

We use Figure 3 to show the relation between MRR and the repetition number. On WN18RR and YAGO3-10, with the increase of repetition number, the model performance first increases. Then, the performance achieves the best when the repetition number is 5. After that, the model performance will decrease a little. On FB15k-237, the model performance does not improve a lot after the repetition number is 2.

We use Figure 4 to show the relation between MRR and center loss weight. On WN18RR and YAGO3-10, we set the repetition number to be 5. On FB15k-237, the repetition number is 16. On the WN18RR dataset, we get the best performance when the center weight is 0.8. The YAGO3-10 dataset gets the best when the weight is 0.2. On Fb15k-237, with the increase of repetition number, the performance gradually increases, and it gets the best performance when the center loss weight is 1. We guess that the performance may be better if the center loss is larger than 1.

According to the two figures, we can find that our model still can get good performance when we just simply set repetition number to be 2 and set center loss weight to be 1. By the simple setting, we can achieve 47.4%, 55.2% and 35.9% MRR on WN18RR, YAGO3-10 and FB15k-237 respectively.

5 Conclusion

In this paper, we point out the discovery of CNN based knowledge graph embedding: "Same Input but Different Output" because of Dropout operation. Compared to the baseline, we add two operations: repetitive prediction and center loss.For repetitive prediction, we use a model to repeatedly process the input and get different predictions because of Dropout operation. Then, we calculate the average binary cross entropy loss. For center loss, we push each repetitive predictions close to each other. In the experiments part, our model outperforms the baseline InteractE on the three datasets. To summarize, to use repetitive prediction and center loss, there is only one requirement: "Same Input but Different Output", such as using Dropout operation.

References

Saadullah Amin, Stalin Varanasi, Katherine Ann Dunfield, and Günter Neumann. Lowfer: Low-rank bilinear pooling for link prediction. In *International Conference on Machine Learning*, pages 257–268. PMLR, 2020.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. Hy-

pernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, 2019.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Empirical Methods in Natural Language Processing*, 2019.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26:2787–2795, 2013.

Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*, 2017.

T Dettmers, P Minervini, P Stenetorp, and S Riedel. Convolutional 2d knowledge graph embeddings. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, volume 32, pages 1811–1818. AAI Publications, 2018.

Alberto Garcia-Duran and Mathias Niepert. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676*, 2017.

Xavier Glorot, Antoine Bordes, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *arXiv preprint arXiv:1301.3485*, 2013.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696, 2015.

Xiaotian Jiang, Quan Wang, and Bin Wang. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 978–987, 2019.

Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295, 2018.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2181–2187, 2015.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Feiliang Ren, Juchen Li, Huihui Zhang, Shilei Liu, Bochao Li, Ruicheng Ming, and Yujia Bai. Knowledge graph embedding with atrous convolution and residual learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1532–1543, 2020.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067, 2019.

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

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, 2016.

Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. Knowledge graph completion via complex tensor factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772, 2017.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha P Talukdar. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *AAAI*, pages 3009–3016, 2020.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer, 2014.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2735–2745, 2019.