McKR - Recommendation Systems Final Project

Original Paper: Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation

Or Oxenberg, Sahar Baribi
Github: https://github.com/saharBaribi/RecSyc_Project

July 8, 2021

Abstract

Deep learning has revolutionized many research fields and there is a recent surge of interest in applying it to collaborative filtering (CF). However, in real recommendation scenarios CF often suffers from sparsity and cold start problems. One approach used to solve this problem is the Multi-task feature learning approach for Knowledge graph enhanced Recommendation (MKR), a framework that learns high-order interactions between items in recommender systems and entities in the knowledge graph. The main issue with MKR is the scale problem due to the direct dependency of training time and embedding dimension. In this work we propose a CNN based approach to control this dependency and to dynamically represent the latent features spatial dependency of the entities of the KG. We achieved similar results compared to MKR on three datasets and proved the correctness of using CNN in this architecture, and the potential efficiency over larger embedding dimensions and dataset.

1 Introduction

During the last few decades recommendation systems (RS) have taken more and more place in our lives. Recommendations are everywhere, whether it's content recommendations like YouTube and Netflix, e-commerce or online advertisement. Recommendation systems aims to address the information explosion by suggesting relevant items to users, that meet the users personalized interests. A very popular algorithm in recommendation systems is collaborative filtering, which utilizes users' historical interactions and makes recommendations based on their common preferences [KBV09]. But collaborative filtering often suffers from sparsity and from the cold start problem, and researchers suggest using side information to handle these problems. There are various types of side information that can be incorporated into recommendation systems, such as social connections among users and social networks[JE10], metadata of items, multimedia [WWY15, ZYL+16] and more. There are also many works that suggest using knowledge graphs as side information to recommender systems [WZW+18, WZXG18, YRS+14, ZYL+16, ZYL+17]. In order to incorporate knowledge graphs, the information is usually pre-processed by knowledge graph embedding methods [WWW+18], that embeds the data, entities and relations, into a low dimensional vector space. One work that uses knowledge graph as side information to enhance the recommender system performance is MKR [WZZ+19] - Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In their work, Wang et al. suggest a deep end-to-end framework that utilizes knowledge graph embedding task to assist the recommendation task. They combined the two tasks by using cross&compress units which "automatically share latent features and learn high-order interactions between items in recommender systems and entities in the knowledge graph". In this work, we will follow Wang et al. framework, and try to improve their results by changing the cross&compress units they suggest in their paper. The cross&compress units are structured by two parts - the cross part that we will leave unchanged, and the compress part, that in our work will be replaced by a convolution network (CNN). CNN is able to assign importance to various aspects in the data it receives, and is able to differentiate from

one another. We believe that replacing the compress part in the cross&compress units in the original paper with a convolution network will enhance the performance of the recommender system and it's running time, since the framework will be able to learn which interactions are more meaningful and to improve the performance based on this information.

After a discussion of related work and scientific background in section 2, we describe our method and present the improvement we suggest to the MKR framework [WZZ+19] in section 3. We then present our evaluation methods and the experimental plan in section 4 and our results in section 5. Our results presents a slight improvement over some of the datasets, and similar results as in the MKR paper in others. We will discuss these results in section 6. Finally we conclude with a discussion on our suggested improvement and future work that can be done.

2 Scientific Background and Related Work

In this section we will present some background, the main ideas and the related methods required to understand both the original paper, and the suggested improvement.

2.1 Collaborative Filtering

Collaborative filtering (CF) is one of the most popular methods in RS. CF methods try to predict the opinion the user will have on different items and to be able to recommend the most suitable items to each user based on the user's previous liking and interactions and the opinions of users with the same preferences. Meaning, CF methods are based on past interactions recorded between users and items in order to produce new recommendations. We use the user-item interactions to detect similar users and/or similar items and make predictions based on these estimated proximities. But although CF methods are highly used in RS, we typically have large products sets and user ratings for only a small fraction of them, which means we have a sparsity problem. Moreover, CF often suffers from the cold start problem - how to recommend new items, and what do we recommend to new users. There are various approaches to handle those problems, when one of them is incorporating side information in the recommender system. Side information can vary from images[ZYL+16] and textual data[WWY15], to social connections [JE10]. Knowledge graphs are another side information for RS, which usually contains prolific information and connections between entities.

2.2 Knowledge Graphs

Knowledge graph (KG) is a knowledge base that uses a graph structured data model of topology to integrate data. It is a multi-relational graph composed of entities (nodes) and relations (different types of edges). Each edge is represented as a triple of the form (head entity, relation, tail entity), indicating that two entities are connected by a specific relation [WMWG17]. Graphs allow to postpone the definition of a schema, allowing the data and its scope, to evolve in a more flexible manner than typically possible in a relational setting, particularly for capturing incomplete knowledge [HBC+20]. Although KG are effective in representing structured data, due to their high dimensionality and heterogeneity they are hard to manipulate. Therefore in order to use its information we will use knowledge graph embeddings methods. The key idea is to embed components of a KG including entities and relations into continuous vector spaces, so as to simplify the manipulation while preserving the inherent structure of the KG [WMWG17]. KG are often used as auxiliary information that can be utilized in recommendation systems settings. Many works use Knowledge Graph Embeddings (KGE) to improve the performance of the recommendation systems[WZW+18, WZXG18, YRS+14, ZYL+16, ZYL+17]. But, as the original paper stated, most suffer from different limitations. Some, like Personalized Entity Recommendation (PER) [YRS+14] and Factorization Machine Group lasso (FMG) [ZYL+17], rely on manually designed meta-path and meta graph based latent features to represent the connectivity between users and items along different types of relation paths/graphs. Some need to prepare the entity embeddings in advance or are not able to capture the importance of relations between two entities, like Deep Knowledge aware Network (DKN)[WZXG18] and RippleNet [WZW+18]. And others, like Collaborative Knowledge base Embeddings (CKE) utilizes KGE module that are more suitable for in-graph applications like link predictions.
MKR framework addresses this limitations - they observe the two tasks as correlated tasks since an item

in the RS can be associated with entities in KG. They model the shared features between items and entities in a designed cross&compress units that models high order interactions and controls the cross knowledge transfer for both the recommendation task and the KGE task. We will further elaborate on the MKR framework at the end of this section.

2.3 Multi-task

Multi-task learning (MTL) is a subfield of machine learning in which multiple learning tasks are solved at the same time by a shared model, while leveraging shared properties and differences across tasks. All of the learning tasks are assumed to be related to each other, and it is found that learning these tasks jointly can lead to performance improvement compared with learning them individually. Multi-task learning helps to improve the generalization performance of all tasks and the data efficiency and to reduce over-fitting, by using useful information for all tasks [Cra20, ZY17]. An important motivation of MTL is to alleviate the data sparsity problem where each task has a limited number of labeled data. MTL aggregates the labeled data in all the tasks to obtain a more accurate learner for each task [ZY17]. MTL methods can be partitioned to 3 main directions - architecture design, optimization and task relationship learning. An example for MTL method applied to RS is the multi-domain collaborative filtering [ZCY12]. Zhang et al. consider multiple collaborative filtering tasks in different domains simultaneously and exploit the relationship between domains. They allow the knowledge learned in each domain to be adaptively transferred across different domains by learning the correlations between the domains. MTL can also be combined with other learning paradigms to improve the performance of learning tasks further, including semi-supervised learning, active learning, unsupervised learning,and reinforcement learning [ZY17] . In their paper, Wang et al.[WZZ+19] claims that utilizing KGE in recommender systems can also be modelled as cross-domain recommendation [TWSS12] or transfer learning [PY09], but modeling this as multi-task learning is their main contribution. As mentioned, we follow Wang et al. framework, where they found that the inter-task similarity between the knowledge graph embeddings and the recommender system is not only helpful for the recommendation task, but also to the knowledge graph embedding.

2.4 Deep Recommendation Systems

Neural network-based recommendation models have emerged as an important tool for tackling personalization and recommendation tasks. These networks differ significantly from other deep learning networks due to their need to handle categorical features[NMS+19]. Roughly speaking, deep Recommendation Systems (DRS) can be classified into two categories: (1) Using deep neural networks to process the raw features of users or items, for example auto-encoders, a popular choice of deep learning architecture for RS [ESF18]. The auto-encoder acts as a nonlinear decomposition of the rating matrix replacing the traditional linear inner product. One example is AutoRec [SMSX15] that decomposes the rating matrix with an auto-encoder followed by reconstruction to directly predict ratings obtaining competitive results on numerous benchmark datasets.(2) Using deep neural networks to model the interaction among users and items, like neural Matrix Factorization (NMF) [HLZ+17]. NMF address implicit feedback by jointly learning a matrix factorization and a feed forward neural network. The outputs are then concatenated before the final output to produce an interaction between the latent factors and the nonlinear factors. McKR deploys a multi-task learning framework that utilizes the knowledge from a KG by learning the spatial relationships of the item-entity by using CNN to assist recommendation. This approach combines the two categories mentioned above, where the cross&compress units and the KG corresponds to the second category and the recommendation module correspond to the first category.

2.5 Convolutional Neural Networks

A convolutional Neural Network (CNN,ConvNet) is a deep learning algorithm most commonly applied to analyze visual imagery. CNN can assign importance to various aspects/ objects in the image and be able to differentiate one from the other. CNNs are regularized versions of multi-layer perceptrons, they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. CNN is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. It reduces

the image dimensionality into a form that is easier to process using the filters, without losing features that are important. ConvNets can use the learned knowledge for multiple tasks such as image and video recognition, image analysis and classification, media recreation, recommendation systems, NLP tasks and more. The structure of simple CNN is shown in Figure 1.

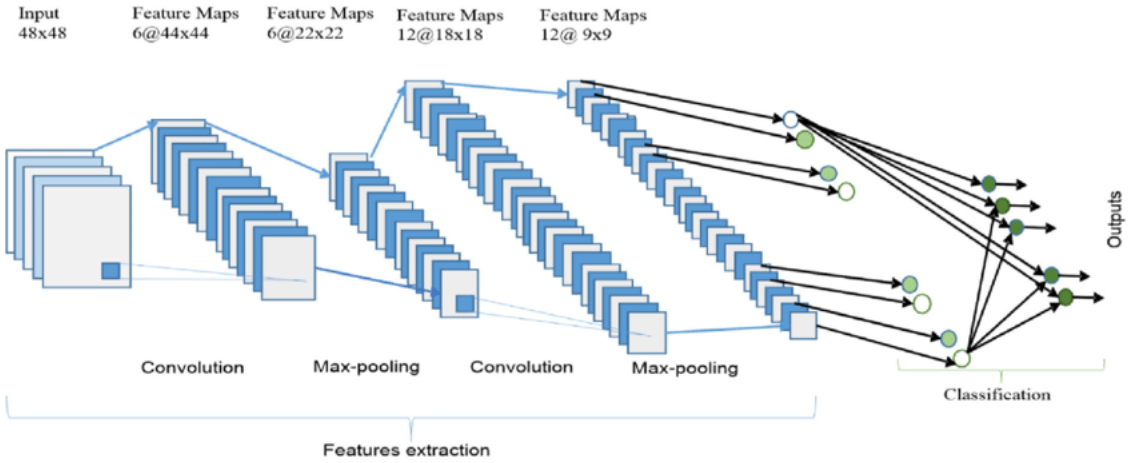

Figure 1: The overall architecture of the Convolutional Neural Network (CNN) includes an input layer, multiple alternating convolution and max-pooling layers, one fully-connected layer and one classification layer.

In our work, we replace the compress part in the cross&compress units with CNN, a detailed explanation is presented in the Method section.

2.6 Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation

In this work, we rely on Wang et al. MKR framework, a Multi-task learning approach for Knowledge graph enhanced Recommendation, described in [WZZ+19] and illustrated in Figure 2.

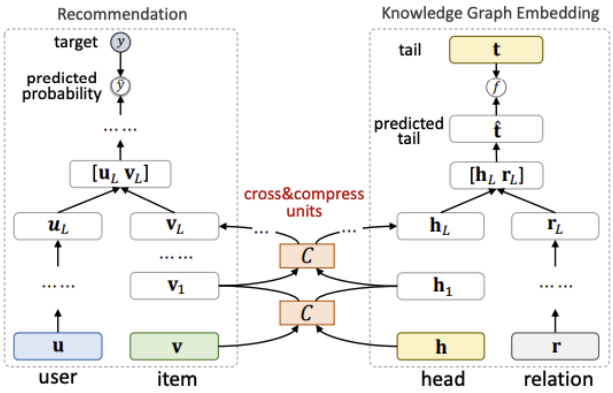

Figure 2: The framework of MKR. The left and right part illustrate the recommendation module and the KGE module, respectively, which are bridged by the cross&compress units

MKR is a generic, end-to-end deep recommendation framework, which aims to utilize knowledge graph embedding task to assist recommendation task, that consists of three components: (1) The recommendation module (2) The knowledge graph embedding module, and (3) cross&compress units. We will elaborate on each of the three components, but a full explanation can be found in the original paper.

2.6.1 Cross&Compress Units

The cross&compress units are designed to model feature interactions between items and entities, see Figure 4. The units consists of two parts: (1) Cross - constructs a matrix of the pairwise interactions of the latent features of entity e and item v, presented in Formula 3. (2) Compress - outputs the feature vectors of items and entities for the next layer by projecting the cross feature matrix into their latent representation spaces, presented in Formula 4 and 5. Through the cross&compress units MKR can adaptively adjust the weights of the knowledge transfer and learn the relevance between the two tasks.

2.6.2 Recommendation Module

The input consists of two raw feature vectors that describe user u and item v. User's raw features are passed through fully connected neural network layer to extract latent features, while the item's features are extracted using the cross&compress units. The output from the cross&compress units is later inserted again into the recommendation module and the latent features of both the user and the item are combined by a prediction function f_{RS}, for example, inner product. The final predicted probability for user u to engage with item v is:

$$\hat{y}_{uv} = \sigma(f_{RS}(u_L, v_L)) \tag{1}$$

2.6.3 Knowledge Graph Embedding Module

As mentioned, the knowledge graph is comprised of entity-relation-entity of triples (h, r, t). h,r and t are denoted as the head, relation, and tail of a knowledge triple, respectively. The input of the KGE module is the head and relation and it predicts the tail using the embedding representation it learns. The module embeds entities and relations into continuous vector spaces while preserving their structure. It utilizes multiple cross&compress units and nonlinear layers to process the raw feature vectors of head h and relation r. Then, it concatenates the latent features and pass them to a K-layer MLP to predict tail t. Finally, The score of the triple (h,r,t) is calculated using a score(similarity) function f_{KG}:

$$score(h, r, t) = f_{KG}(t, \hat{t}) \tag{2}$$

where in the paper they used the normalized inner product as the choice of the score function.

3 Method

In our work we suggest the McKR framework - Multi-Task Feature Learning using CNN for Knowledge Graph Enhanced Recommendation. We rely on the MKR framework by Wang et al.[WZZ+19] as explained above. McKR differentiate from MKR in the cross&compress unit, as illustrated in Figure 3. In the original paper cross&compress units consists of two parts - the Cross and the Compress part. The cross creates a matrix that contains the pairwise interactions of the latent feature $v_l \in \mathbb{R}^n$ and $e_l \in \mathbb{R}^n$ from layer l(Figure 4):

$$C_l = v_l e_l^T = \begin{bmatrix} v_1 e_1 & ... & v_1 e_d \\ v_d e_1 & ... & v_d e_d \end{bmatrix} \tag{3}$$

where $C_l \in \mathbb{R}^{dxd}$ is the cross feature matrix of layer l, and d is the dimension of hidden layers.
The compress part forwards the feature vectors of items and entities to the next layer by projecting the cross feature matrix into their latent representation spaces. The simple forward process is done by weights multiplication:

$$v_{l+1} = C_{lw} W_l^{VV} + C_{lw}^T W_l^{EV} + b_l^V \tag{4}$$

$$e_{l+1} = C_{lw} W_l^{VE} + C_{lw}^T W_l^{EE} + b_l^V \tag{5}$$

where $w_l \in \mathbb{R}^d$ and $b_l \in \mathbb{R}^d$ are trainable weights and bias vectors. We suggest a change in this simple multiplication process to a matrix multiplication with spatial filters by using CNN (Figure 3):

$$\hat{C}_{111} = C * F = \begin{bmatrix} v_1 e_1 & ... & v_1 e_n \\ v_n e_1 & ... & v_n e_n \end{bmatrix} \circ \begin{bmatrix} W_{11} & ... & W_{1n} \\ W_{n1} & ... & W_{nn} \end{bmatrix} + \vec{b}_n \tag{6}$$

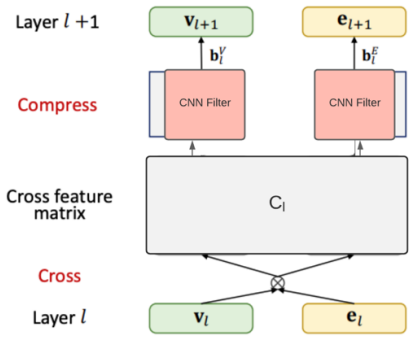

Figure 3: McKR cross&compress

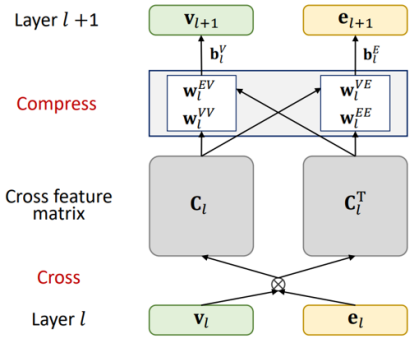

Figure 4: MKR cross&compress

where $F \in \mathbb{R}^{nxn}$is the filter we use Z time (Z is chosen in the hyper-parameters tuning process) and n is the filter dimension. $\hat{C} \in \mathbb{R}^{nxnxf}$ is a tensor that represents the convolution result where f is the amount of filters (also chosen by tuning the parameters).
We repeat this process for the entire matrix C by moving the filter over the matrix. After building $\hat{C}$ we can repeat the process with more convolution layers, flatten the final output and continue to forward process like the original paper does with simple dense layer. CNN incorporate constraints and achieve some degree of shift and deformation in variance using three ideas: (1) local receptive fields (2) shared weights and (3) spatial sub sampling. The use of shared weights also reduces the number of parameters in the system aiding generalization. By using CNN we are able to successfully capture the spatial dependencies in the matrix C without manually transpose it and get even more complex relations between the latent features of the embedding of the item and head.

4 Evaluation

4.1 Datasets

We run our suggested improvement on the following three datasets in our experiments:

- MovieLens-1M[1]: widely used benchmark dataset in movie recommendations, which consists of approximately 1 million explicit ratings (ranging from 1 to 5) on the MovieLens website.
- Book-Crossing[2]: dataset contains 1,149,780 explicit ratings (ranging from 0 to 10) of books in the Book-Crossing community.
- Last.FM[3]: dataset contains musician listening information from a set of two thousand users from Last.FM online music system.

These datasets were chosen due to their use in the original paper. In addition, in the original MKR paper, Wang et al. [WZZ+19] transform these datasets into implicit feedback where each entry is marked with 1 indicating that the user has rated the item positively, and sample an unseen set marked as 0 for each user. The threshold of positive rating is 4 for MovieLens1M, while no threshold is set for Book-Crossing and Last.FM due to their sparsity. We used the data sets after the transformation.

4.2 Evaluation Metrics

To evaluate the result of the McKR model we used the same metrics Wang et al. used in the original paper. This was done in order to simplify the comparison of the two approaches (original paper Vs. our suggested improvement):

[1] https://grouplens.org/datasets/movielens/1m/
[2] http://www2.informatik.uni-freiburg.de/~cziegler/BX/
[3] https://grouplens.org/datasets/hetrec-2011/

- Accuracy (ACC): measures how many observations, both positive and negative, were correctly classified

$$ACC = \frac{tp + tn}{tp + tn + fp + fn} \tag{7}$$

fp - false positive, fn - false negative, tp - true positive, tn - true negative

- Area under the curve (AUC): In order to define AUC we need to understand the receiver operating characteristic (ROC) curve. ROC is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters - (1) True Positive Rate (TPR) and (2) False Positive Rate (FPR).

$$TPR = \frac{tp}{tp + fn} \tag{8}$$

$$FPR = \frac{tn}{tn + fp} \tag{9}$$

Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The AUC is the area under the ROC graph and is calculate by simple integral of this graph:

$$AUC = \int_0^1 ROC \tag{10}$$

4.3 Experimental Plan

We used the same three datasets presented above, with the explicit - implicit transformation as a pre-process step. For us to compare the results of the suggested model to the original paper we need to use the same data in order to remove any bias factor. We extracted the original paper's code from Github [4], and ran the original code to make sure we are able to restore the original paper's results (shown in Table 3 in the Results section). Then, we changed the cross&compress units to add the convolution layers. After implementing our changes, we first run the framework with the same parameters as in the original paper. We witnessed that while in some datasets we receive relatively fair results on the train set, we are not able to generalize well on the validation and test set. This was due to a low dimension size used in the original paper, that forced the use of small filters. Later, we worked on changing the dimensions, the amount of filters and it's sizes.

Parameter	Description
dim	dimension of user and entity embeddings
L	number of low layers
H	number of high layers
b	the size of the batch
k	training interval of KGE task
λ_2	weight of l2 regularization
α_{rs}	learning rate of RS task
α_k	learning rate of KGE task
l_c	convolutional layer number of filters
l_d	dense layer number of filters

Table 1: Hyper parameters list

To improve the results further, we applied a grid search over multiple hyper-parameters (see Table 1) to achieve the best AUC and ACC result over the three datasets. We collected the results and get the best one in the epoch level. All three datasets and their hyper-parameters are given in Table 2, while the training interval of KGE task was always 2.

[4] https://github.com/hwwang55/MKR

Dataset	hyper-parameter values
MovieLens-1M	H = 1, L = 1,b = 1024, $l_c = 8$, $l_d = 16$ dim = 16, λ_2 = 1e-06, $\alpha_{rs} = 0.0009$, $\alpha_k = 0.0002$
Book-Crossing	H = 1, L = 1,b = 32, $l_c = 16$, $l_d = 8$ dim = 8, λ_2 = 1e-06, $\alpha_{rs} = 0.0002$, α_k = 2e-05
Last.FM	H = 1, L = 1,b = 128, $l_c = 16$, $l_d = 16$ dim = 16, λ_2 = 1e-07, $\alpha_{rs} = 0.0002$, $\alpha_k = 0.001$

Table 2: Hyper parameters values

5 Results

We performed two experiments to compare McKR with the original MKR. We compared time, ACC, AUC on all the datasets.

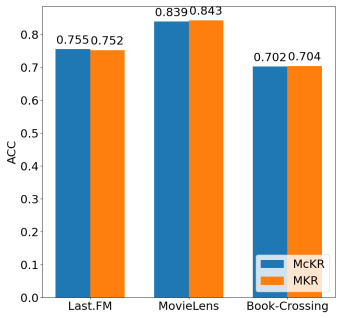

Figure 5: Exp. 1-ACC

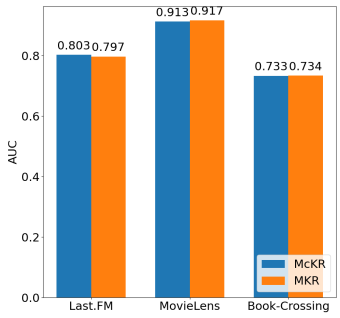

Figure 6: Exp. 1-AUC

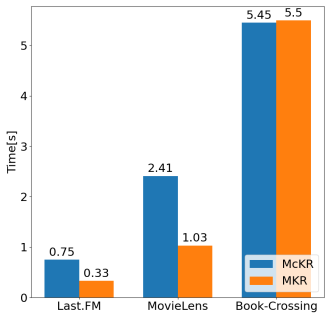

Figure 7: Exp. 2-running time

5.1 Experiment 1: ACC & AUC

In this experiment we tried to compare the same metric that was used in the original paper to see if the McKR convolution process in the cross&compress unit effects the results. If both scores are higher, we can conclude that we are able to determine more precisely that user rated item positively or negatively and make good recommendation. As shown in Figure 5 and 6, the results of our model are almost the same. We improved ACC and AUC on the Last.FM dataset, the smallest dataset in size. ACC was improved from 0.752 in the original dataset to 0.755 with our suggested framework. AUC improved from 0.797 to 0.803. For the BookCrossing and MovieLens datasets, our results were lower by 0.003+-0.001 for both AUC and ACC, compared to the original paper. This marginal is similar to the improvement we had in the Last.FM dataset.

5.2 Experiment 2: Running Time

Here we compared McKR and MKR average running time for each epoch. The motivation was to understand if the CNN architecture can reduce the parameters after the hyper-parameters tuning due to the effect of increased or decreased overall network size. Many systems operate at a scale that stresses the challenges of successful academic and "small data" applications. Furthermore, increased user awareness and choice has led to preference promiscuity meaning users are happier than ever to drop a service that is not personally accurate. This is why we chose to present this experiment - it is important to see the usability of such algorithm in real world problem. Figure 7 shows that in the Last.FM and MovieLens datasets McKR took more time to run on each epoch, more than 2x times for each epoch. For the book-Crossing dataset we see a different result - MKR ran for a longer average time per epoch but the difference is not as big as observed for the two other datasets (the difference is only 0.05 seconds). The book-crossing dataset, the largest among the compared datasets, as can be seen in the running times per epoch, took the most time to run. We believe that McKR could handle

bigger datasets better when comparing running times. This is due to the parameters reduction we achieve when using CNN compared to MLP. Though we didn't test this hypothesis, CNN are effective for image classification as the concept of dimensional reduction suits the huge number of parameters in an image. The image in our case is the latent interaction matrix and we can increase the embedding dimension without worrying about the increased parameters size.

Model	LastFM			MovieLens			BookCrossing		
	ACC	AUC	Time	ACC	AUC	Time	ACC	AUC	Time
MKR - Original	0.752	0.797	-	**0.843**	**0.917**	-	**0.704**	**0.734**	-
MKR - Reconstruction	0.7531	0.7993	**0.33**	0.8407	0.915	**1.03**	**0.704**	0.7303	5.50
McKR	**0.755**	**0.803**	0.75	0.839	0.913	2.41	0.702	0.733	**5.45**

Table 3: The results of AUC, Accuracy, Time in CTR prediction

6 Discussion

The results presented in the previous sections shows that although we could not achieve higher results on all datasets compared to the original paper, we reached similar or slightly higher results, as presented in Table 3. The results show both the original paper published results, the results we got from running the original paper's code, and the results from our experiments. Note that in some of the metrics and datasets, our results were lower than the original results but higher than the restored results (e.g. AUC on the book crossing dataset). Our suggested improvement takes the compress part in the cross&compress units and replaces it with a CNN. The compress part can be viewed as a MLP, meaning it can be viewed as a fully connected (FC) layer. When using CNN, 1x1 convolutions can replace FC layers, and reduce the parameters of the network. Although we did not use 1x1 convolutions in our suggested improvement, this basic idea explains why we were able to achieve almost the same results as the original paper. The difference between the original paper and the suggested improvement is not big, but we believe that we were able to capture the essence of the network and reaching almost the same results, and slightly better on some datasets proves this.

As described in the results section, in our experiments we decided to add an extra metric we compare the performance to - time. Though the original paper did not present the time, we measured the time for the MKR framework when trying to restore the original paper results. By using CNN, we reduce the networks parameters and therefore we would expect our framework to run faster. The results show a different picture - MKR framework was able to defeat our time in two out of the three datasets. With that said, we believe that since CNN does reduce the parameters of the network, it will work better on bigger datasets. This assumption can be evaluated in future work.

In our work we encountered several limitations. First, the code is written in Tensorflow v1, which makes it difficult to change the architecture of the network. We had to change the variables the network receives in order to incorporate our improvement in the network. Moreover, running CNN is preferable on GPU, which we did not have access to. We struggled with using GPU and Tensorflow together, and this could impacted the results we observe. Another limitation was time - to improve our suggested architecture we used hyper-parameters tuning. Our experiments contained only a small subset of hyper-parameter tuning. We focused on the main hyper-parameters we believed will have the most impact, when focusing on the range around the original paper's parameter. Future work can include richer hyper-parameter tuning search on a wider range.

7 Conclusion

This paper proposes McKR, a multi-task learning approach for knowledge graph using convolution neural network for enhanced recommendation. We relied on the framework presented in the original

MKR paper [WZZ+19] while changing the cross&compress units. We left the recommendation module, and the KGE module unchanged, and changed the inner architecture to fit the simple convolution with variety of filters. By doing so, McKR had the ability to develop an internal representation of a two-dimensional latent interaction matrix between item and entities from the KG. This allows the model to learn position and scale in variant structures in the data, which is important when working with large datasets which are more common in production, especially in the field of recommendation systems. For future work, we plan to investigate other types of datasets, with emphasis on large scale datasets to see if we are able to obtain a significant decrease in epoch running time, and to achieve higher precision score with higher embedding dimensions. We will also want to use Residual neural networks (ResNet) [HZRS16] with deeper CNN architecture on the same datasets. We believe that by doing so we will be able to run for more epochs and achieve higher AUC and ACC scores. The skip connection property of the ResNet helps ease the training of the networks that are substantially deeper than before by eliminating the degradation problem. Kaiming et al. [HZRS16] proved that ResNets are easier to optimize and can have high accuracy at considerable depths. We will try to implement skip connections between the CNN layers and the rest of McKR architecture.

References

[Cra20] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[ESF18] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 515–524, 2018.

[HBC+20] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[HLZ+17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JE10] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142, 2010.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[NMS+19] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[PY09] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[SMSX15] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112, 2015.

[TWSS12] Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1285–1293, 2012.

[WMWG17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[WWW+18] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[WWY15] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244, 2015.

[WZW+18] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426, 2018.

[WZXG18] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844, 2018.

[WZZ+19] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*, pages 2000–2010, 2019.

[YRS+14] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292, 2014.

[ZCY12] Yu Zhang, Bin Cao, and Dit-Yan Yeung. Multi-domain collaborative filtering. *arXiv preprint arXiv:1203.3535*, 2012.

[ZY17] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

[ZYL+16] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[ZYL+17] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 635–644, 2017.