An Analysis of Convolutional 2D Knowledge Graph Embeddings

Gerrit Krot, Alejandro Yaber Llanos, and Calvin Nau

I. Task Definition, Evaluation Protocol, and Data.

Knowledge graphs are graphical data models, "…that capture information in applied scenarios that entail the integration, extraction, and management of data at a large scale from different diverse sources…" [1], [2]. A graph is understood to be a set of nodes with edges describing relationships between nodes [3]. A knowledge graph can typically be described by the triple (s, r, o) where s and o are the subject and object (both nodes), and r is a relationship, or an edge [4]. It should be noted that subjects and objects are the same set of nodes in a graph, but a relationship between nodes defines whether a node is a subject or an object in a relationship. A knowledge graph can be as simple as one (s, r, o) tuple, but commonly complex knowledge graphs are a set of (s, r, o) triples that describe an entire system. Knowledge graphs contain a limited subset of knowledge of what is true in the world [5].

A challenge in using knowledge graphs is that links between subjects and objects are often missing. Link prediction, "...aims to achieve novel links for an established knowledge graph and existing links with other entities" [5]. Link prediction determines the edges in a graph when information is incomplete. Link prediction's utility is dependent on the graph being described. In common applications, such as recommendation systems, spam mail detection, and privacy control in social networks, prediction of relationships between a subject and an object before they interact [6], [7]. In applications, like network reconstruction or source identification in published works, missing links are filled in and extraneous links are pruned to construct a more accurate graph than the input [6]. An example of link prediction is predicting the relationship between a subject (i.e., Lionel Messi) and an object (i.e., Argentina) (see Fig. 1.).

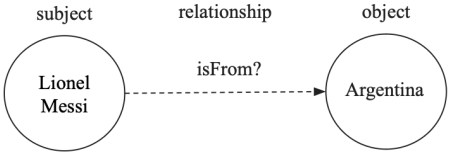

Fig. 1. An example of link prediction is presented. In this example the relationship between the subject (i.e., Lionel Messi) and the object (i.e., Argentina) is unknown and being predicted using link prediction.

Link prediction assessment metrics for knowledge graphs include mean rank (MR), mean reciprocal rank (MRR), and hits-at-k (hits @ k). MR is the average rank of the ground truth triples when ranked based on predicted value [8]. MRR is calculated by finding the reciprocal index of the highest rank correctly predicted data point and averaging the results [8]. To find the reciprocal index, the solutions proposed by the link prediction algorithm are ranked according to probability and the index of the first correct answer is determined. The reciprocal of the index is taken. This metric measures how far the first correct answer is from the highest rank. Values closer to one are considered more optimal. Regarding the metric hits @ k, various "k" values are selected, where k is an integer. The k highest probability predictions are then retrieved as a subset. This subset is analyzed to determine the percentage of correct k-link predictions. This metric accounts for a balanced analysis of performance by analyzing k-predictions, rather than analyzing the single most probable prediction, like MRR.

As previously noted, link prediction datasets describe, at a minimum, a single (s, r, o) relationship. The datasets used by Dettmers et al. have several relationships ("r") of different types relating subjects and objects, common to many canonical datasets. The link prediction datasets used for evaluation include WN18 [9], a subset of the WordNet consisting of word relationships, FB15k [9], a subset of freebase containing knowledge base relations, YAGO3-10, a subset of YAGO3 [10] containing personal attribute relations, and Countries [11], which tests for long-range dependencies on geospatial relationships.

Dettmers et al. [4] note that the data sets WN18 and FB15k suffer from test leakage through inverse relations, an issue that contaminates training instances with test instances. To avoid this issue Toutanova and Chen [12] present a subset of FB15k, called FB15k-237 where inverse relations are withdrawn. Dettmers et al. also discovered similar leakage in WN18. The authors establish WN18RR as a subset of WN18, with 93,003 triples, 40,943 entities, and 11 relations, to resolve this issue [4].

II. Neural Network Machine Learning Model

The authors propose ConvE, a multi-layer convolutional network model for link prediction for knowledge graphs [4]. Interactions among entities and relationships are modeled through convolutional and fully connected layers. A significant contribution of ConvE is the use of 2D convolutions over embedding to achieve the prediction of the missing links. ConvE is composed of three layers: a single convolution layer over 2D-shaped embedding, the projection layer onto the embedding dimension, and an inner product layer (see Fig. 2) [4].

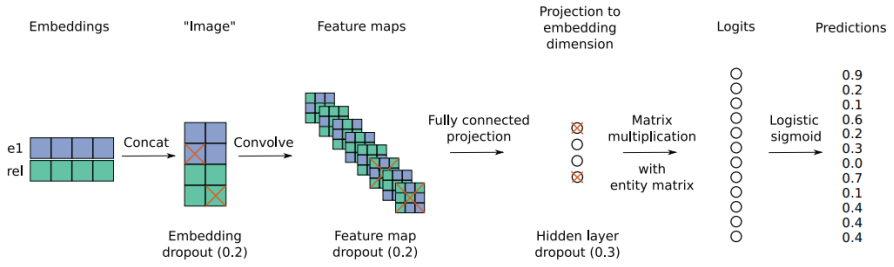

Fig. 2. Architecture of ConvE Model [4]

At a high level, ConvE first converts the triple describing a link to a token, or a numeric representation of the (s, r, o) triple. These factors are the entities "s" and "o" $\epsilon\ \varepsilon$, that refer to a subject and object, where r $\in\ R$ is the relationship amongst them [4]. Following tokenization, the model embeds and then concatenates the s and r components. It then performs batch normalization and a dropout operation to prevent overfitting. The output is then sent through a convolutional layer with multiple output channels. The convolutional layer applies a linear function shifted over the embedded inputs [13], [14]. Including 2D convolution allows for the extraction of more feature interactions between the embeddings than a 1D-convolution would be able to achieve [4]. Additionally, the inclusion of multiple convolution output channels allows for the model to learn more "expressive" features without resulting in the exponential growth in memory requirements associated with shallow neural networks.

Following convolution, the outputs are batch normalized and a ReLU activation function is applied; the authors note this is intended to speed up training. Then, another dropout layer is applied, and the output is manipulated into a 1-D tensor. This output is passed through a fully connected linear layer which is followed by another dropout layer and batch normalized. Again, a ReLU activation function is applied. The output

undergoes matrix multiplication with the embedding weights of o, or e_o, and is summed with the bias. Following the application of a sigmoid function the predictions are determined. The model is trained to minimize binary cross-entropy loss using stochastic gradient descent. The model results in a scoring function which can be summarized as $\psi(s, r, o) = \psi_r(e_s, e_o) \in R$ determined by the relationships between the embedded entities e_s and e_o (see Eqn. 1) [4]. In this formula, $f(vec(f([\overline{e_s}; \overline{r_r}] * \omega))W)$ is the learned function.

Equation 1

$$\psi_r(e_s, e_o) = f(vec(f([\overline{e_s}; \overline{r_r}] * \omega))W)e_o \quad [1]$$

In Eqn. 1., $r_r \in R^k$ represents a relation parameter that is dependent on r. Meanwhile, $\overline{e_s}$ and $\overline{r_r}$ represent the 2D reshaping of e_s and r_s.

Minimizing the number of convolution operations allows for a speed-up in computation time [4]. Convolution in Dettmers et. al.'s architecture [4] utilizes between 75 and 90% of the entire computational time in model evaluation. Increasing batch size to accelerate the evaluation speed is a commonly used tactic in link prediction models [15]. However, this is not feasible in convolutional models, as the required memory will rapidly grow larger than the GPU memory capacity when increasing the batch size [4].

Another novel contribution of the ConvE is a scoring procedure that uses a pair (s, r) and scores it against all entities $o \in \varepsilon$ at the same time. This implies a scoring of (1-N), unlike traditional link prediction models that take the entity pair and the relation as a triple of (s, r, o) and score it in a 1-1 system. This alternative (1-N) scoring procedure trains 3 times faster and evaluates 300 times faster than previous methods [4].

A faster forward and backward pass could be achieved if the scoring method used were 1-(0.1N) (i.e. scoring vs 10% of the entities). However, convergence would be 230% slower on the training set. In electing to use a (1-N) scoring method, Dettmers et al. obtain a higher convergence speed at the cost of lost computational performance [4].

Various hyperparameters exist in ConvE which must be considered. Such parameters included the embedding dropout, feature map dropout, projection layer dropout, embedding size, batch size, learning rate, and label smoothing. The dropout parameters can be manipulated to prevent overfitting [16]. Embedding size in this instance refers to the fully linear layers. As noted, Dettmers et al.'s approach uses convolution in an attempt to reduce the need the increase embedding size to achieve higher performance. Therefore, an increase in embedding size should be avoided.

The neural network proposed by Dettmers et. al. [4] provides a novel convolutional model for link prediction. The scoring method allows for an efficient prediction of relations between entities. The integration of convolution in the link prediction task allows for the model to reach a high degree of accuracy without infeasible memory requirements.

III. Experiment Design

A drawback of Dettmers et al.'s [4] approach is that each model is trained independently for a specific dataset, which requires a large amount of time. For example, for new datasets, the model is initialized with random weights, as it would be when trained for the first time. However, sections of the neural network are

dimensionally invariant from dataset to dataset. Therefore, a proposed method to address this is the use of transfer learning (TL), or the use of previously learned information (i.e., weights) in a new model [7]. Formally, TL is learning, "...which involves methods that utilize any knowledge resource…to increase model learning and generalization for the target task" [7]. In the context of ConvE, it is believed that initializing the model weights at the start of training with weights learned from a different dataset, rather than initializing them randomly, would allow the model to reach equivalent performance in less time.

Therefore, the modification is an identification of any weights in ConvE's network architecture that are not dependent on the shape of the input data set (e.g., the 2D-convolution weights). Such weights include the 2D-convolutional layer, the first, second, and third batch normalization layer's biases and weights, and the final fully connected layer's weights and biases. It is believed the weights on these sections of the neural network could be used as improved starting weights for model training on a new dataset. In this experiment, the model is trained for a given dataset, referred to as the learning dataset (e.g., FB15k-237). Upon training completion, the dimensionally invariant weights will be used to initialize ConvE at the start of training for another dataset, which will be referred to as the TL dataset (e.g., WN18RR) (see **Section I** for a detailed description of these datasets).

The experiment will be conducted on two datasets. The dimensionally invariant weights from a model trained on FB15k-237 will be used to initialize training for WN18RR, and vice versa. To ensure that model performance is similar, an analysis will be performed on the following statistics following training: MR, MRR, hits @ 10, hits @ 3, and hits @ 1. These statistics will be determined from the test dataset. Further, total training times will be analyzed. An early stopping procedure will terminate training when the training loss for the model initialized with weights from TL is approximately equal to the loss for the model initialized with random weights on the same dataset and trained for 1000 epochs. If the stopping criteria is not met, the model will be trained for 1000 epochs, as was the default in [4]. The proposed hypothesis can be seen in Table 1 and conditions for confirmation, contradiction, or unclear results are noted.

The required code changes are included in Table 1. At a high level, these code modifications will function to identify neural network weights from the learning dataset which are dimensionally equivalent to the neural network weights for the TL dataset at epoch = 0. These weights will be used to initialize the neural network being trained on the TL dataset. It should be noted that all hyperparameters as described in **Section IV** are not intended to be modified.

Table 1: Proposed Modifications Experiment Summary

Hypothesis	Introducing TL by initializing a neural network's dimensionally invariant weights with a previously trained model's weights will reduce training time without a cost to accuracy. ● Confirmed: Both models train in a fewer number of epochs to reach the same training loss. ● Contradicted: Both models do not train in a fewer number of epochs and/or reach a lower performance at the end of training. ● Not Clear: One model confirms the hypothesis while the other contradicts it.

Independent Variables	Model Initialization Weights (random vs. TL weights)
Control Variables	Model Training Loss, dropout rate, hidden dropout rate, batch size, epoch size, learning rate, Hardware used
Dependent Variables (Measured Results)	Training Time, MR, MRR, Hits @ 10, Hits @ 3, Hits @ 1
Code Modifications	• Parameter & function for loading model data from pre-trained model to access transfer weights • Function to initialize weights of model being trained on a new dataset with the weights from the TL dataset • Modified stopping procedure based on $Loss_{validation\ transfer} \leq Loss_{validation\ random}$

To expand on the details included in Table 1, the independent variables are the initialization weights of the network. The experimentation will be deemed successful if the model initialized with the learned weights from TL takes less time to reach similar loss and performance metrics as the model initialized with random weights. However, this hypothesis addresses a set of related questions. For one, are the two graph representations of different datasets WN18RR, containing work relations, and FB15k-237, containing knowledge base relations, related in their graphical representations? Further, if the graph representations of the relationships are similar, does the ConvE neural network similarly represent these graphs (i.e., do similar graphs result in similar neural network weights)? This is related to the weight updates and whether using non-random weights (weights learned from another dataset) is different than using entirely random weights. In other words, if the neural networks learn similar representations of similar graphs, it is expected that when TL is used there would not be as many weight updates required to reach a high-performing model. If this is true, it would be assumed the graphs are related in some way despite representing different data as the neural network learns related weight representations.

Concerning the dependent variables, the early stopping procedure is dependent on the training loss of the model. This is not a dependent variable itself, however, it will enable early stopping procedures so that changes in training times can be analyzed without bias. The reason for this selection is based on what loss is considered to represent. Generally, it can be considered as a numeric representation of what a model has yet to learn. Therefore, loss is considered as a proxy for equivalent learning in this experimentation and a measure by which training is terminated. Following early termination (should it occur), the model performance will further be assessed using dependent variables such as MR, MRR, Hits @k, and total training time (i.e., the total number of training epochs). If the training time decreases while the other metrics are much lower, performing this would not confirm the proposed hypothesis and be unexpected. However, if training time decreases and the other performance metrics are approximately equivalent, this would confirm the benefits of TL. If our model takes the same amount of time to train and obtains similar accuracy values, this suggests TL has no effects.

All hyperparameters will be held constant; these include but are not limited to the dropout rate, the hidden dropout rate, batch size, epoch size, and the learning rate. All results and experimentation will be

determined by training and testing the models on a Xeon E5-2650 2.2 GHz CPU and x10 Tesla P4 GPU with GPU acceleration enabled.

IV. Experimental Results and Discussion

To reproduce the results of the paper, ConvE was trained on a selection of the datasets that were used by Dettmers et al. [4], namely FB15k-237 and WN18RR. FB15k and WN18 were not analyzed as the README.md file provided with the source code noted that these models suffered from test leakage.

To execute this testing, the default hyperparameters, obtained through grid search by Dettmers et al. [4] were used. These parameters include embedding dropout = 0.2, feature map dropout = 0.2, projection layer dropout = 0.3, embedding size = 200, batch size = 128, learning rate = 0.003, and label smoothing = 0.1. Further, all models were trained for a total of 1000 epochs. It should be noted there was a discrepancy between the default learning rate included in the code and the one proposed in the paper. Given the belief the code is what was used for analysis, the default it included (0.003) was used, as opposed to the rate given in the paper (0.001).

Included is a verification of the results for WN18RR and FB15k-237 which show a high level of alignment between the reproduced results and the results published by Dettmers et al. (see Table 2). The results were compared for MR, MRR, and various hits@k metrics as included in [4]. For a more complete description of such metrics, see **Section I**. It was found that for both datasets the MRR was slightly lower than the one noted by Dettmers et al. A lower MRR suggests lower model performance. Across hits@k metrics, the reproduced results show slightly higher performance than Dettmers et al. on the WN18RR dataset. On the other hand, for the FB15k-237 dataset, the hits@k metrics calculated are slightly lower than Dettmers et al. Similarly, the MR for WN18RR suggests better performance than that noted by Dettmers et al. while FB15k-237 data suggests poorer performance.

Table 2 Model Metric Performance Testing

	WN18RR					FB15k-237				
	MR	MRR	@ 10	@ 3	@ 1	MR	MRR	@ 10	@ 3	@ 1
Dettmers et al.	5277	0.46	0.48	0.43	0.39	246	0.316	0.491	0.350	0.239
Reproduction	5138	0.43	0.51	0.45	0.39	270	0.305	0.479	0.333	0.219
Transfer Learning	5186	0.42	0.50	0.43	0.38	279	0.301	0.470	0.330	0.216

The discrepancies reported could be attributed to random number generation, given Dettmers et al. code does not set the random number generation to a single value, or true discrepancies between the model's reported performance and its true performance. Given the random distribution of indications showing Dettmers et al.'s model is both better and worse than reported, it is believed that the discrepancies between the models are due to randomness and not some systematic biasing of the results. Given this, it is believed that the performance reported by Dettmers et al. can be trusted.

With the results of Dettmers et al. verified, accurate conclusions can be drawn regarding the effect of TL. In Table 2, the results are notable (see the row "Transfer Learning"). First, consider WN18RR performance (see column WN18RR and the row entitled "TL") after being trained with the final model of FB15k-237 used to initialize the weights as described in **Section III**. Across MRR and hits @ K metrics, the model performs only slightly below that of the reproduced results of Dettmers et al. and Dettmers et al.'s direct

results. For MR, the results of TL are poorer performance than that of this work's reproduction of Dettmers et al.'s results but higher performance than that of Dettmers et al.'s reported results.
For FB15k-237 trained with the weights of WN18RR (see column FB15k-237 and the row entitled "TL") the results follow a similar pattern. The hits @ k metrics and MRR of the model trained with TL are slightly poorer than that of this work's reproduction of Dettmers et al. but not significantly different. Similarly, the MR suggests slightly poorer performance than this work's reproduction of Dettmers et al.'s result and the direct results of Dettmers et al. In conclusion, the models do not perform significantly differently when initiated with transfer learned weights which is a positive indication.

To properly analyze the effect of TL on ConvE and the impact of the proposed modifications, models' performance metrics must be analyzed in conjunction with training times. The effect is substantial when considering the two proposed models (see Table 3). For WN18RR, the model's training time is reduced by approximately nine and a half hours, taking only 22.5% of the time it took to train with random weights. It trained for 243 epochs before reaching the early stopping procedure rather than training for one thousand epochs.

Table 3: Viability Testing of Time and Dataset Attributes

	Training Runtime (HH:MM:SS)	Transfer Learning Training Runtime (HH:MM:SS)	Training Epochs	Transfer Learning Training Epochs
WN18RR	12:11:35	02:44:45	1000	243
FB15k-237	15:46:31	00:39:33	1000	37

Similar effects are prompted for the FB15k-237 model. The full model training time for TL is 4.18% of the total elapsed time when initiated with random weights. This is a reduction of over fifteen hours in training time. Rather than training for one thousand epochs, it trains for thirty-seven epochs before reaching the early stopping procedure.

The significance of these results is best understood in the context of the simultaneous reduction in training time while model performance is maintained. This suggests that the hypothesis of this work, that similar model performance could be reached with a reduction in training time is correct. The results are notable as the graphs on which the original models were trained represent different relationships. As previously noted, WN18RR represents word relationships while FB15k-237 represents knowledge relationships. Regardless of the different meanings of these graphs, the model weights are relatively transferable with both models having a notable reduction in training time while maintaining accuracy when they are initiated with TL data.

A notable result is that the training time of FB15k-237 was reduced more dramatically than the training time of WN18RR was by the implementation of TL. This suggests that the weights learned by WN18RR are more transferable to FB15k-237 than the weights learned by FB15k-237 are transferable to WN18RR. It is unclear what may be causing this unexpected relationship. One potential explanation is the differences in the size of the datasets. For example, it may be the WN18RR was able to learn a better relationship after being trained for 1000 epochs then FB15k-237 due to its reduced dimensionality (i.e., fewer relationship

types, fewer triples, etc.) (see Table 4). Therefore, when WN18RR was transferred to use as the initial weights for FB15k-237 the model's training time was reduced more significantly as the weights it was initialized with were more optimal on the original training graph. This explanation is partially complete as its weights transfer to a model with more relationships (FB15k-237) better than one with more relationships (FB15k-237) does to one with fewer relationships (WN18RR). This is surprising as it would be expected a model which has to learn more relationships would be more robust. Regardless, this points to future work analyzing in more depth what factors cause TL to be more useful.

Table 4: Dataset Attributes

	Triples	Entities	Relationships
WN18RR	93,003	40,943	11
FB15k-237	310,116	14,541	237

Another interesting relationship is that FB15k-237 has far more triples relative to its entity count than WN18RR. This seems to indicate that FB15k-237 is more strongly connected in general than WN18RR. Despite these differences in strong vs. weakly connected graphs, transfer learning is still able to improve training times while maintaining accuracy.

One limitation of this work is the early stopping procedure. In this work, the early stopping procedure for TL was based on the training loss for a dataset observed after 1000 epochs of it being trained with randomly initialized weights. If transfer learning was used in application, this early stopping procedure would not be applicable as the training loss would be unknown (as there would not be any prior training). Future improvements could develo an early stopping procedure based on model fit. Potential measures for model fit could include stopping training once the training loss stops improving by some percentage change or if the training loss is within some threshold of the test loss.

This work addressed the utility of TL on dissimilar link prediction graphs using the ConvE link prediction model proposed by Dettmers et al. Notably, the training time was dramatically reduced for models to reach a similar loss value as when initialized with random weights and trained for 1000 epochs. Further, while training time was reduced the resulting models produced results indistinguishable to if the model was trained for 1000 epochs with randomly initialized weights. While limitations exist in this work, the results presented seem to suggest that for the ConvE model transfer learning has the potential to dramatically reduce training times while maintaining the performance of learned models.

Note: Please find the code repository located at https://github.com/calvinnau4/conve-recreate-and-improve.git

V. References

[1] A. Hogan *et al.*, "Knowledge graphs," *ACM Comput Surv*, vol. 54, no. 4, Jul. 2021, doi: 10.1145/3447772.

[2] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor, "Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done," *Knowledge Graphs*, vol. 17, no. 2, pp. 48–75, 2019.

[3] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Trans Neural Netw Learn Syst*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[4] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D Knowledge Graph Embeddings." [Online]. Available: www.aaai.org

[5] S. M. Kazemi and D. Poole, "SimplE Embedding for Link Prediction in Knowledge Graphs." [Online]. Available: https://github.com/Mehran-k/SimplE.

[6] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, "Link prediction techniques, applications, and performance: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 553. Elsevier B.V., Sep. 01, 2020. doi: 10.1016/j.physa.2020.124289.

[7] L. Wu, P. Cui Jian Pei, and L. Zhao Eds, "Graph Neural Networks Foundations, Frontiers, and Applications."

[8] N. Hubert, P. Monnin, A. Brun, and D. Monticolo, "Knowledge Graph Embeddings for Link Prediction: Beware of Semantics!," 2022. [Online]. Available: https://hal.science/hal-03787512

[9] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating Embeddings for Modeling Multi-relational Data."

[10] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "YAGO3: A Knowledge Base from Multilingual Wikipedias," 2013. [Online]. Available: https://imt.hal.science/hal-01699874

[11] G. Bouchard, S. Singh, and T. Trouillon, "On Approximate Reasoning Capabilities of Low-Rank Vector Spaces," 2015. [Online]. Available: www.aaai.org

[12] K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference."

[13] U. Michelucci, *Advanced applied deep learning: Convolutional neural networks and object detection*. Apress Media LLC, 2019. doi: 10.1007/978-1-4842-4976-5.

[14] E. Charniak, *Introduction to Deep Learning*. The MIT Press, 2019. Accessed: Nov. 11, 2023. [Online]. Available: https://mitpress.mit.edu/9780262039512/introduction-to-deep-learning/

[15] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, "Knowledge graph embedding for link prediction: A comparative analysis," *ACM Trans Knowl Discov Data*, vol. 15, no. 2, Jan. 2021, doi: 10.1145/3424672.

[16] P. Baldi and P. Sadowski, "Understanding Dropout," in *Advances in Neural Information Processing Systems 26*, 2013. Accessed: Dec. 09, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf