Data Augmentation for Few-Shot Knowledge Graph Completion from Hierarchical Perspective

Yuanzhou Yao [1,2], **Zhao Zhang** [1,3]*, **Yongjun Xu** [1] and **Chao Li** [3,1]*
[1] Institute of Computeing Technology, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Zhejiang Lab, Hangzhou, China
{yaoyuanzhou21s, zhangzhao2021, xyj, lichao}@ict.ac.cn

Abstract

Few-shot knowledge graph completion (FKGC) has become a new research focus in the field of knowledge graphs in recent years, which aims to predict the missing links for relations that only have a few associative triples. Existing models attempt to solve the problem via learning entity and relation representations. However, the limited training data severely hinders the performance of existing models. To this end, we propose to solve the FKGC problem with the data augmentation technique. Specifically, we perform data augmentation from two perspectives, i.e., inter-task view and intra-task view. The former generates new tasks for FKGC, while the latter enriches the support or query set for an individual task. It is worth noting that the proposed framework can be applied to a number of existing FKGC models. Experimental evaluation on two public datasets indicates our model is capable of achieving substantial improvements over baselines.

1 Introduction

Knowledge graphs (KGs) are structured semantic knowledge bases used to describe concepts and their interrelationships in the physical world in symbolic form. Many KGs in the real world, such as Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), WordNet (Miller, 1992), Wikidata (Vrandecic and Krötzsch, 2014) and NELL (Mitchell et al., 2018), consist of triple facts in the form of (head entity, relation, tail entity), e.g., (*Paris*, *capitalOf*, *France*) indicates that Paris is the capital of France. KGs have been introduced into various downstream tasks of NLP, such as question answering (Saxena et al., 2020), dialogue systems (He et al., 2017) and information extraction (Hoffmann et al., 2011), etc. The integrity of KG promotes the performance of downstream tasks. However, KGs in the real world are far from complete and comprehensive. Therefore, it is necessary to complete KGs by inferring new triple facts.

To complete KGs, most existing embedding-based KG completion models require adequate triples for each relation as training data, such as TransE (Bordes et al., 2013), RotatE (Sun et al., 2019) and ConvE (Dettmers et al., 2017). However, in reality, the number of triples for each relation conforms to a long-tail distribution (Xiong et al., 2018), i.e., only a small number of relations occur frequently, while most relations only occur a few times in a KG. This phenomenon hinders to learning reliable representations for infrequent relations and further degrades the KG completion performance.

This has motivated an emerging research topic named few-shot knowledge graph completion (FKGC), where one task is to predict the tail entity t in a query $(h, r, ?)$ given only a few entity pairs of the task relation r. GMatching (Xiong et al., 2018) is the first study on the FKGC task, which proposes the basic framework and problem formulation. FSRL (Zhang et al., 2020) and FAAN (Sheng et al., 2020) further improve the attention mechanism of the GMatching framework. MetaR (Chen et al., 2019) and GANA (Niu et al., 2021) adopt the meta-based paradigm in meta-learning as the basic architecture. Although the above methods achieve promising results for the FKGC problem, they still suffer from the limited training data for each relation. To this end, we propose to alleviate the above issue using the data augmentation technique.

Specifically, as shown in Figure 1, we aim to augment the data of each task within its own distribution, and densify the task distribution by providing interpolated tasks. Therefore, we propose to augment data from a hierarchical perspective. The inter-task view generates new tasks for the FKGC model. And the intra-task view provides

* Corresponding author: Zhao Zhang and Chao Li.

Proceedings of the 29th International Conference on Computational Linguistics, pages 2494–2503
October 12–17, 2022.

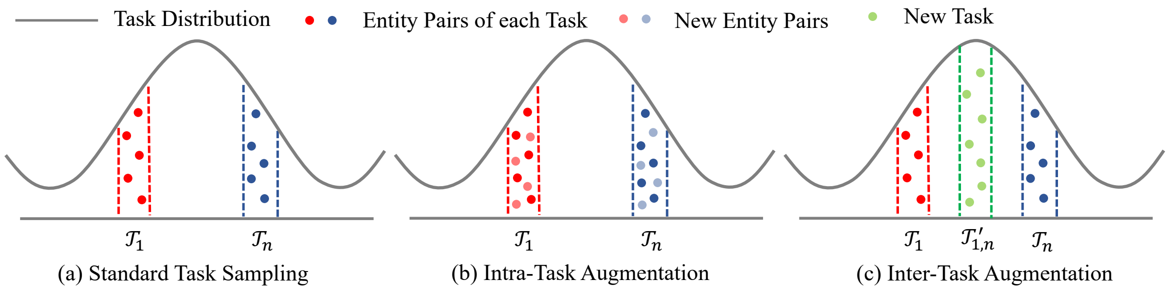

Figure 1: Motivations behind data augmentation for FKGC. (a) Two tasks are sampled from the task distribution; (b) Intra-task augmentation methods that augment each task within its own distribution; (c) Inter-task augmentation densifies the task-level distribution by performing cross-task level interpolation or inversing task.

entity pairs for each individual task. This setting is capable of enriching luxuriant data and densifying the data distribution for FKGC models, which is beneficial to achieving better performances. We propose two data augmentation methods for each view to enhance the existing FKGC model. Particularly, the proposed technique is general and can be applied to a number of existing FKGC models. To the best of our knowledge, this is the first work to solve the FKGC task using the data augmentation technique. Finally, experimental results validate the effectiveness of the proposed method.

In a nutshell, we highlight our main contributions as follows,

- To solve the problem of limited training data, we propose to use the data augmentation technique for the FKGC problem. To the best of our knowledge, this is the first work that utilizes data augmentation for FKGC.
- To provide adequate data for the FKGC models, we propose to conduct data augmentation from hierarchical perspectives, i.e., intra-task perspective and inter-task perspective.
- Experimental results on benchmark datasets show the proposed method can be applied to various existing FKGC models and achieve substantial improvements over baselines competitors.

2 Background

In this section, we provide problem formulation and the settings of FKGC.

2.1 Problem Formulation

A Knowledge graph $\mathcal{G}$ is represented as a collection of triples $\{(h, r, t)\} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity set and relation set, respectively. The task of knowledge graph completion falls into two categories: predicting the unknown relation r between the head entity and the tail entity $(h, ?, t)$, and predicting the missing entity t or h based on the head/tail entity and the relation $(h, r, ?)$ or $(?, r, t)$. In this paper, following previous FKGC work, we aim to predict the missing term in a given query $(h, r, ?)$. Unlike traditional knowledge graph completion task that requires abundant triples for the query relation during training, FKGC is only accessible to a few training triples when predicting the tail entity. Specifically, the goal of FKGC is to rank the true tail entities t_{true} higher than other candidate entities $\mathcal{C}_r$. Each relation r corresponds to a candidate entity set, which is constructed based on entity type constraints (Xiong et al., 2018; Toutanova et al., 2015). In the test phase, the corresponding candidate entities are ranked, and the groud truth tail entity is supposed to rank first among the candidates.

2.2 FKGC Settings

FKGC follows the standard few-shot learning settings, and the training data consists of a series of tasks. In FKGC, each task corresponds to a relation in KG $r \in \mathcal{R}_f$, where $\mathcal{R}_f$ is the few-shot relation set, and the rest of the relations in KG are background knowledge graph relations $\mathcal{R}_b$, which consist of high-frequency relations, $\mathcal{R}_f \cup \mathcal{R}_b = \mathcal{R}$ and $\mathcal{R}_f \cap \mathcal{R}_b = \phi$. The triples corresponding to each relation in R_b form the background knowledge graph G', which is mainly used for pre-training the representations of the entity set $\mathcal{E}$ and background knowledge graph relations $\mathcal{R}_b$. The head and tail entity pairs $\{(h_{k,i}, t_{k,i})\}$ of a few-shot relation constitutes a task. Each task $\mathcal{T}_k$ corresponds to one support set S_k and one query set Q_k, and a part of the task is selected to form the meta-training set $\mathcal{T}_k \in T_{train}$.

Train Prase: The goal of FKGC is to rank all entities in the candidate entity set with S_k as reference, and the ground truth tail entity t_k should be higher than the other false entities t_{false}. We formulize the ranking loss function as $\mathcal{L}_\theta$ and θ denotes the model parameters, and the loss function is set to reflect the rank of the true tail entities in Q_k given S_k. The objective of training the FKGC model is defined as:

$$min_\theta \frac{1}{|T_t|} \sum_{\mathcal{T}_k^t \in T_t} \sum_{(h_{k,i},t_{k,i}) \in Q_k} \frac{L_\theta(t_{k,i}|h_{k,i}, S_k)}{|Q_k|} \tag{1}$$

where $|T_t|$ denotes the number of tasks in T_{train} and the $\mathcal{T}_k^t$ is sampled from the meta-training set T_{train}.

Test Prase: When training is complete and tail entity completion is performed, FKGC models will sample new tasks from the meta-test set $\mathcal{T}_{k'} \in T_{test}$ for prediction. Meta-test set T_{test} also has the support set $S_{k'}$ and query set $Q_{k'}$, which are defined in the same way as in meta-training. Similarly, each task corresponds to a relation in the meta-test relations $r_t \in R_{test}$ that does not appear in the training phase: $\mathcal{R}_{test} \cap \mathcal{R}_{train} = \phi$ and $\mathcal{R}_{train} \cup \mathcal{R}_{test} = \mathcal{R}_f$. These new relations only need to be predicted for tail entity $(t_{k',i}|h_{r',i}, \mathcal{C}_{k'})$ in $Q_{k'}$ with K triples of as $S_{k'}$ a reference.

3 Related Work

3.1 Data Augmentation Strategy

Data augmentation has been widely used to prevent deep neural networks from over-fitting to the training data (Bishop, 1995). Most of the traditional augmentation methods generate new data according to the mixed application transformation of data types or proposed target tasks (Cubuk et al., 2019), which can be independently applied to various data types and tasks, improving the generalization and robustness of deep neural networks. Input mixup (Zhang et al., 2017) linearly interpolates between two input data, and trains the model using mixed data with corresponding soft labels. Following this work, a variety of mixup methods for data augmentation have been proposed. Manifold mixup (Verma et al., 2018) applies the mixup strategy in the hidden feature space, and CutMix (Yun et al., 2019) proposes an image mixup method based on spatial copy and paste. Puzzle Mix (Kim et al., 2020) proposes a mixup method based on saliency and local statistics of the given data. MixSKD (Yang et al., 2022) incorporates Mixup with self-knowledge distillation into a unified framework to regularize the two image views. Most of these methods aim at the field of image processing. In this paper, we specially tailor the mixup strategy for the FKGC task.

3.2 FKGC models

Existing FKGC approaches fall into two categories: metric learning-based methods and meta learner-based methods. We outline the main structures of these two methods and describe them separately in the following

Metric learning-based methods. GMatching is the first research work on FKGC (Xiong et al., 2018), and it utilizes metric learning-based methods as the backbone and divides the model into three subparts: neighbor encoder, entity pairs encoder, and matching processor. *Neighbor encoder* is designed to enhance the representation of each entity with its local connections in the knowledge graph (one-hop neighbors).

Gmatching directly sums all neighbors on average, FSRL (Zhang et al., 2020) uses the attention mechanism (Veličković et al., 2017) to encoding neighbors, and FAAN (Sheng et al., 2020) leverages the relation in task R_f to introduce the adaptive attention network. The embedding of entities $\mathbf{h}, \mathbf{t} \in \mathbb{R}^d$ are then fed into the *entity pairs encoder* $\mathcal{F}_r$:

$$\mathbf{r}_{k,i}^q = \mathcal{F}_r(\mathbf{h}_{k,i}^q, \mathbf{t}_{k,i}^q), \quad \mathbf{r}_{k,\cdot}^s = \mathcal{F}_r(S_k) \tag{2}$$

where $(\mathbf{h}_{k,i}^q, \mathbf{t}_{k,i}^q) \in Q_k$, the query relation $\mathbf{r}_{k,i}^q \in \mathbb{R}^d$ and support set relation $\mathbf{r}_{k,\cdot}^s \in \mathbb{R}^d$ are then compared by *matching processor function*: $Score(\mathbf{r}_{k,i}^q, \mathbf{r}_{k,\cdot}^s) = \mathcal{M}(\mathbf{r}_{k,i}^q, \mathbf{r}_{k,\cdot}^s)$, since $\mathbf{r}_{k,i}^q$ and $\mathbf{r}_{k,\cdot}^s$ represent the same task realtion r_k, their score should be as high as possible.

Meta learner-based methods. MetaR (Chen et al., 2019) is the first model to use the Meta learner-based method as backbone. In contrast to the standard gradient-based meta-learning, MetaR defined two kinds of meta information which are shared between support set and query set. It can be viewed as a bi-level optimization problem.

Formally, the bi-level optimization process can be formulated as:

$$\theta^* \leftarrow argmin_\theta \sum_{\mathcal{T}_k \in T_{train}} [\mathcal{L}_\theta(\mathbf{r}_{k,\cdot}^{meta}, Q_k)]$$

$$s.t.\ \mathbf{r}^{meta}_{k,\cdot} = \mathbf{r}^{s}_{k,\cdot} - \eta \bigtriangledown_{\mathbf{r}^{s}_{k,\cdot}} \mathcal{L}_\theta(\mathbf{r}^{s}_{k,\cdot}, S_k) \tag{3}$$

Where $\mathbf{r}^{s}_{k,\cdot}$ is obtained by Equation 2; $\mathcal{L}_\theta$ and η denote the knowledge graph loss function and inner-loop learning rate. GANA (Niu et al., 2021) shares a similar idea with MetaR, but learns the relation-specific hyper-plane parameters to model complex relations.

4 Methodology

The section describes the details of the data augmentation for FKGC. It falls into two data augmentation methods from the task perspective: intra-task augmentation and inter-task augmentation. inter-task augmentation generates new tasks for the FKGC model, and the intra-task augmentation provides entity pairs for each individual task. We will describe how each of these data augmentation methods is applied to metric learning-based methods and meta learner-based methods.

4.1 Intra-Task Augmentation

Intra-task augmentation only enlarges the pool of triples to be sampled during training within each individual task, not the number of tasks. Since all entity pairs under the same task have the same relation, FKGC uses entity pairs to model few-shot relations $\mathcal{R}_f$. Assume $\mathbf{r}_{k,i}$ is the relation embedding vector modeled by the i-th entity pair $(h_{k,i}, t_{k,i})$ in the k-th task $\mathcal{T}_k$, and since both $\mathbf{r}_{k,i}$ and $\mathbf{r}_{k,j}$ belong to the same task, $\mathbf{r}_{k,j} \approx \mathbf{r}_{k,i}$. We consider the combination of different modeling vectors of the same relation can still represent this task relation: $(\mathbf{r}_{k,i}, \mathbf{r}_{k,j}) \approx \mathbf{r}_{k,j}$. Therefore mixing different entity pairs in the task after the entity pairs encoder can generate a new modeling vector. It is worth noting that what is generated is a new modeling vector belonging to this task relation instead of a new triple. In detail, the mixing strategy follows Manifold Mixup (Verma et al., 2019) where inputs and hidden representations are mixed up. A task contains a query set and a support set: $\mathcal{T}_k = (Q_k, S_k)$, so two types of intra-task augmentation can be derived according to the differences in augmenting settings:

4.1.1 Query Augmentation

Query augmentation enlarges the pool of evaluation data to be sampled during training. Since the structure of the two mainstream FKGC models is different (details in Section 3.2), we will introduce how query augmentation is applied to these two types of models.

Algorithm 1 The Process of Intra-Augmentation

Require: Meta-training set T_{train}, model parameter θ, outer-loop learning rate φ, inner-loop learning rate η, candidate set $\mathcal{C}$.

while not converge **do**
  Randomly sample a task $\mathcal{T}_k$ from T_{train}
  if Metric Learning-Based Methods **then**
    if Query Augmentation **then**
      $\theta = \theta - \varphi \mathbb{E}_{r_k^{new_q}} \bigtriangledown_\theta \mathcal{H}_\theta(\mathbf{r}^{new_q}_{k,i}, \mathbf{r}^{s}_{k,\cdot})$.
    else if Support Augmentation **then**
      $\theta = \theta - \varphi \mathbb{E}_{r_k^{q}} \bigtriangledown_\theta \mathcal{H}_\theta(\mathbf{r}^{q}_{k,i}, \mathbf{r}^{new_s}_{k,\cdot})$.
    end if
  else if Meta Learner-Based Methods **then**
    if Query Augmentation **then**
      $\theta = \theta - \varphi \bigtriangledown_\theta \mathcal{L}_\theta(\mathbf{r}^{meta}_{k,\cdot}, Q^{new}_k)$.
      $s.t.\ \mathbf{r}^{meta}_{k,\cdot} = \mathbf{r}^{s}_{k,\cdot} - \eta \bigtriangledown_{\mathbf{r}^{s}_{k,\cdot}} \mathcal{L}_\theta(\mathbf{r}^{s}_{k,\cdot}, S_k)$.
    else if Support Augmentation **then**
      $\theta = \theta - \varphi \bigtriangledown_\theta \mathcal{L}_\theta(\mathbf{r}^{meta'}_{k,\cdot}, Q_k)$.
      $s.t.\ \mathbf{r}^{meta'}_{k,\cdot} = \frac{1}{|S_k|} \sum_{i=0}^{|S_k|} \mathbf{r}^{meta'}_{k,i}$.
    end if
  end if
end while

Metric Learning-Based Methods try to learn generalizable metrics and the corresponding matching functions $\mathcal{M}(\cdot, \cdot)$ from a set of training tasks. Assume that $\mathbf{r}^{new_q}_{k,i}$ denotes a new modeling vector of query set, we can formulate this change on $\mathcal{M}$ as:

$$\begin{aligned} &\mathcal{M}(\mathbf{r}^{q}_{k,i}, \mathbf{r}^{s}_{k,\cdot}) := \mathcal{M}(\mathbf{r}^{new_q}_{k,i}, \mathbf{r}^{s}_{k,\cdot}) \\ &s.t.\ \mathbf{r}^{new_q}_{k,i} = \lambda \mathbf{r}^{q}_{k,i} + (1-\lambda)\mathbf{r}^{q}_{k,j} \end{aligned} \tag{4}$$

Where $\mathbf{r}^{q}_{k,i}$ and $\mathbf{r}^{q}_{k,j}$ are obtained by Eqn.2 and $\lambda \in [0, 1]$ is sampled from a Beta distribution $Beta(\alpha, \beta)$. Then we construct a set of negative queries $Q^{neg}_k = \{(h_{k,i}, t^{-}_{k,i})\}$ by randomly corrupting the tail entity, where the false tail entity belongs to the task entity candidate set: $t^{-}_{k,i} \in \mathcal{C}_k$. The loss function is formally defined as:

$$\mathcal{H}_\theta(\mathbf{r}^{new_q}_{k,i}, \mathbf{r}^{s}_{k,\cdot}) = [\gamma + \mathcal{M}(\mathbf{r}^{new_q}_{k,i}, \mathbf{r}^{s}_{k,\cdot}) - \mathcal{M}(\mathbf{r}^{neg}_{k,i}, \mathbf{r}^{s}_{k,\cdot})]_+ \tag{5}$$

where $[x]_+ = max(0, x)$ is standard hinge loss, and γ is a margin separating positive and negative queries.

Meta learner-based methods are a bi-level optimization process; query augmentation for meta learner-based methods can improve the outer-loop optimization. Like metric learning-based methods, we also construct a new query set, but due to the outer-loop optimization process does not encode the entity pairs of the query set into the relation vector, we directly mix up the original entity pair:

$$Q_k^{new} = \{(\lambda \mathbf{h}_{k,i}^q + (1-\lambda)\mathbf{h}_{k,j}^q, \lambda \mathbf{t}_{k,i}^q + (1-\lambda)\mathbf{t}_{k,j}^q)\} \quad (6)$$

where $(\mathbf{h}_{k,i}^q, \mathbf{t}_{k,i}^q)$, $(\mathbf{h}_{k,j}^q, \mathbf{t}_{k,j}^q) \in \mathcal{Q}_k$ and the Eqn.3 is reformulated as:

$$\theta^* \leftarrow argmin_\theta \sum_{\mathcal{T}_k \in T_{train}} [\mathcal{L}_\theta(\mathbf{r}_{k,\cdot}^{meta}, Q_k^{new})] \quad (7)$$

4.1.2 Support Augmentation

Support augmentation enlarges the pool of triples to be sampled for the support set, not to increase the value of $K = |S_k|$.

Metric Learning-Based Methods. Like the support augmentation, We also randomly sample two relation modeling vectors for mixup to generate a new support set and the Eqn.4 is reformulated as:

$$\begin{aligned} \mathcal{M}(\mathbf{r}_k^q, \mathbf{r}_{k,\cdot}^s) &:= \mathcal{M}(\mathbf{r}_k^q, \mathbf{r}_{k,\cdot}^{new_s}) \\ s.t.\ \mathbf{r}_{k,i}^{new_s} &= \lambda \mathbf{r}_{k,i}^s + (1-\lambda)\mathbf{r}_{k,j}^s \end{aligned} \quad (8)$$

where the $\mathbf{r}_{k,i}^{new_s}$ is obtained by aggregating all $\mathbf{r}_{k,i}^{new_s}$ to represent the new support set relation and the loss function of support augmentation for metric learning-based methods is reformulated as: $\mathcal{H}_\theta(\mathbf{r}_k^q, \mathbf{r}_{k,\cdot}^{new_s})$.

Meta Learner-Based Methods. Support augmentation can be applied to support set in the inner-loop to fine-tuning the relation vector $\mathbf{r}_k^s$. This strategy enlarges the pool of fine-tuning data. Since both $\mathbf{r}_{k,i}^s$ and $\mathbf{r}_{k,j}^s$ represent the same task relation, their fine-tuning gradients with respect to the task relation should be consistent, Therefore, we mix the respective fine-tuned gradients of the two relation vector and apply the resulting gradient to $\mathbf{r}_{k,i}^s$.

$$\begin{aligned} \mathbf{r}_{k,i}^{meta'} &= \mathbf{r}_{k,i}^s - [\lambda G(\mathbf{r}_{k,i}^s) + (1-\lambda)G(\mathbf{r}_{k,j})] \\ s.t.\ G(\mathbf{r}_{k,j}) &= \triangledown_{\mathbf{r}_{k,i}^s} \mathcal{L}_\theta[\mathbf{r}_{k,i}^s, (\mathbf{h}_{k,i}^s, \mathbf{t}_{k,i}^s)] \end{aligned} \quad (9)$$

where entity pair $(\mathbf{h}_{k,i}^s, \mathbf{t}_{k,i}^s) \in S_k$ and the $\mathbf{r}_{k,\cdot}^{meta'} = \frac{1}{|S_k|}\sum_{i=0}^{|S_k|} \mathbf{r}_{k,i}^{meta'}$ will be used as the relation vector of all entity pairs in the query set $(\mathbf{h}_{k,i}^q, \mathbf{r}_{k,\cdot}^{meta'}, \mathbf{t}_{k,j}^q)$, thus participating in the outer optimization of the model parameters. Changing $\mathbf{r}_{k,\cdot}^{meta}$ in Eqn.3 to $\mathbf{r}_{k,\cdot}^{meta'}$ is the outer-loop optimization process of support augmentation for meta methods.

4.2 Inter-Task Augmentation

Inter-task augmentation increases the number of tasks by creating new relations r_k' to enlarge the task pool of meta-training set T_{train}. To enlarge the value of $|T_{train}|$, we devise two task augmentation methods: inverse augmentation and interpolation augmentation.

4.2.1 Inverse Augmentation

FKGC models represent few-shot relation $\mathcal{R}_f$ using entity pairs, which consist of head and tail entities. Intuitively, flipping the head and tail entities to represent another relation can enrich the dataset, e.g., the triple *(Elon Musk,SonOf, Errol Musk)* can be flipped as *(Errol Musk, ParentOf, Elon Musk)*, where the entity pair *(Errol Musk, Elon Musk)* can represent a new relation *ParentOf.* When we generalize this augmentation to all tasks in the meta-training set, a new reversed meta-training set can be generated: $T_{train}' = \{\mathcal{T}_1', \cdots, \mathcal{T}_N'\}$, where N is the number of tasks in T_{train} and $\mathcal{T}_k' = \{(t_{k,i}, , h_{k,i})\}_{|\mathcal{T}_k|}$. Merge the two meta-training sets to get a new larger meta-training set: $T_{train}^{new} = T_{train}' \cup T_{train}$. Therefore, the number of tasks of T_{train}' is twice that of the original train set T_{train}, and finally T_{train}' will replace T_{train} to participate in the training process.

4.2.2 Interpolation Augmentation

We think that the combination of two different relations can generate a new relation, such as $father+mother = grandma$. We adopt a mixup strategy for linear addition rather than direct combination: $\mathcal{T}_{mix_{i,j}}' = \lambda\mathcal{T}_i + (1-\lambda)\mathcal{T}_j$, which adjusts the weight of the two task relations in the new relation by λ. Since λ is obtained by sampling from the beta distribution $Beta(\alpha, \beta)$, the number of tasks in the meta-training set tends to be infinite in theory. When $\lambda = 0.5$, the mixup strategy is equivalent to a direct combination.

Metric Learning-Based Methods. Input the entity pairs of task i: $\mathcal{T}_i$ and task j: $\mathcal{T}_j$ into Eqn.8 respectively to obtain their corresponding relation modeling vectors, and mix up the relation vectors in these two tasks to generate a new task $r_{mix_{i,j}}'$. We can formulate this process as follows:

$$\begin{aligned} \mathbf{r}_{mix_{i,j},k}^{q'} &= \lambda \mathbf{r}_{i,k}^q + (1-\lambda)\mathbf{r}_{j,k}^q \\ \mathbf{r}_{mix_{i,j},k}^{s'} &= \lambda \mathbf{r}_{i,k}^s + (1-\lambda)\mathbf{r}_{j,k}^s \end{aligned} \quad (10)$$

Then we pass $\mathbf{r}_{mix_{i,j},k}^{q'}$ and $\mathbf{r}_{mix_{i,j},k}^{s'}$ through matching processor function to calculating the similarity

Algorithm 2 The Process of Inter-Augmentation

Require: Meta-training set T_{train}, inner-loop learning rate φ, Beta distribution parameters α, β, candidate set $\mathcal{C}$.

if Inverse Augmentation **then**
  $min_{\theta}\mathbb{E}_{\mathcal{T}'_k} \sum_{(h_{k,i},t_{k,i})\in Q'_k} \frac{L_\theta(t_{k,i}|h_{k,i},S'_k)}{|Q'_k|}$
  $s.t.\ \mathcal{T}'_k \in T^{new}_{train} = T'_{train} \cup T_{train}$
else if Interpolation Augmentation **then**
  while not converge **do**
    Sample two tasks $\mathcal{T}_i$, $\mathcal{T}_j$ from T_{train}.
    if Metric-Based Methods **then**
      $\theta = \theta - \varphi \bigtriangledown_\theta \mathcal{H}_\theta(\mathbf{r}^{q'}_{mix_{i,j},k}, \mathbf{r}^{s'}_{mix_{i,j},k})$.
    else if Meta-Based Methods **then**
      $\theta = \theta - \varphi \bigtriangledown_\theta \mathcal{L}_\theta(\mathbf{r}^{meta'}_{mix_{i,j},k}, Q'_{mix_{i,j}})$.
      $s.t.\ \mathbf{r}^{meta'}_{mix_{i,j},k} = \lambda\mathbf{r}^{meta}_{i,k} + (1-\lambda)\mathbf{r}^{meta}_{j,k}$.
    end if
  end while
end if

score of them: $\mathcal{M}(\mathbf{r}^{q'}_{mix_{i,j},k}, \mathbf{r}^{s'}_{mix_{i,j},k})$. Since they represent the same new task relation $r'_{mix_{i,j}}$, their score should be as high as possible.

Meta Learner-Based Methods are different from metric learning-based methods to generate a new task; it not only needs to mixup the relation vector of the support set in the two tasks: $\mathbf{r}^{meta'}_{mix_{i,j}} = \lambda\mathbf{r}^{meta}_{i,\cdot} + (1-\lambda)\mathbf{r}^{meta}_{j,\cdot}$, but also needs to generate a corresponding query set:

$$Q'_{mix_{i,j}} = \{(\lambda\mathbf{h}^q_{i,k} + (1-\lambda)\mathbf{h}^q_{j,k}, \lambda\mathbf{t}^q_{i,k} + (1-\lambda)\mathbf{t}^q_{j,k})\}_{i\neq j} \tag{11}$$

Substituting $\mathbf{r}^{meta'}_{mix_{i,j}}$ and $Q'_{mix_{i,j}}$ into Eqn.3 is the bi-level optimization process.

5 Experiments

5.1 Experimental Setup

5.1.1 Datasets.

We evaluate our augmentation methods on two public benchmark datasets: NELL-One and Wiki-One[1]. In these datasets, few-shot relations $\mathcal{R}_f$ that have more than 50 but less than 500 triples are selected to construct few-shot tasks. There are 67 tasks and 183 tasks in NELL-One and Wiki-One datasets respectively. Correspondingly, the partition 51/5/11 of the 67 tasks and the partition 133/16/34 of the 183 tasks are used for training/validation/test. Furthermore, the background knowledge graph $\mathcal{G}'$ except few-shot relations are used to pre-train entity vectors and $\mathcal{R}_b$ vectors. The statistic details of both datasets are shown in Table 2.

5.1.2 Comparison Methods.

In order to evaluate the effectiveness of our augmentation methods, We conduct experiments on three metric learning-based methods and two meta learner-based methods: GMatching, FSRL, FAAN and MetaR, GANA (model details in Section 3.2). All the above methods use the original datasets for training without data augmentation.

5.1.3 Implementation Details.

For all the models, we initialize the entity and relation embeddings by background knowledge graphs pre-trained on TransE, released by GMatching. The K-shot (K = 1, 5) support pairs are selected randomly and experimented for all the models. For a fair comparison, we run the official code and adopt the default hyperparameters for each baseline. GMatching and FSRL do not report the experimental results in the 5-shot case, but we can adopt the results reported by FAAN for these two models in the 5-shot case. Moreover, FSRL and FAAN do not report the experimental results in the 1-shot case, so we run their released code to get baseline results in the 1-shot case. we re-implement the GANA model to make a fair comparison. For MetaR, we choose both pre-train setting and in-train setting to evaluation our augmentation methods. We set $\alpha = 2$ and $\beta = 2$ in $Beta(\alpha, \beta)$ and the and the neighborhood's maximum size is fixed to 50 on both datasets. For other hyperparameters, we adopt the default value of their released code.

5.1.4 Evaluation Metrics.

To evaluate the performance of all models on our augmentation methods, which aims to rank the ground truth tail entity $t^q_{k,i}$ for each query among the task candidates $\mathcal{C}_k$. We report two standard evaluation metrics on both datasets: MRR and Hits@N. MRR is the mean reciprocal rank and Hits@N is the proportion of the ground truth entities ranked in the top N; in our experiments, we set N = 1, 5, 10 and the few-shot size is set to K=1, 3.

5.2 Experimental Results and Analysis

The MRR results of FKGC models with all augmentation methods on NELL-One and Wiki-One are shown in Table 1, we can conclude that:

1. Our augmentation methods applied to all baseline models improve their original MRR val-

[1]https://github.com/xwhan/One-shot-Relational-Learning

NELL-One	Methods	Shot	Vanilla	Intra-Task		Inter-Task	
				Query	Support	Inverse	Interpolation
Metric-based	Gmatching	1-shot	0.168	$0.185^{+0.017}$	$0.175^{+0.007}$	$0.179^{+0.011}$	$0.205^{+0.037}$
		5-shot	0.176	$0.191^{+0.015}$	$0.180^{+0.004}$	$0.196^{+0.020}$	$0.211^{+0.035}$
	FSRL	1-shot	0.148	$0.172^{+0.024}$	$0.164^{+0.016}$	$0.157^{+0.009}$	$0.179^{+0.031}$
		5-shot	0.153	$0.178^{+0.025}$	$0.165^{+0.012}$	$0.169^{+0.016}$	$0.185^{+0.032}$
	FAAN	1-shot	0.194	$0.231^{+0.037}$	$0.216^{+0.022}$	$0.209^{+0.015}$	$0.224^{+0.030}$
		5-shot	0.279	$0.304^{+0.025}$	$0.282^{+0.003}$	$0.284^{+0.005}$	$0.294^{+0.015}$
Meta-based	MetaR (Pre-Train)	1-shot	0.164	$0.204^{+0.040}$	$0.227^{+0.063}$	$0.217^{+0.053}$	$0.194^{+0.030}$
		5-shot	0.209	$0.224^{+0.015}$	$0.240^{+0.031}$	$0.233^{+0.024}$	$0.217^{+0.008}$
	MetaR (In-Train)	1-shot	0.250	$0.308^{+0.058}$	$0.319^{+0.069}$	$0.254^{+0.004}$	$0.266^{+0.016}$
		5-shot	0.261	$0.331^{+0.070}$	$0.332^{+0.071}$	$0.275^{+0.014}$	$0.307^{+0.046}$
	GANA	1-shot	0.254	$0.278^{+0.024}$	$0.291^{+0.037}$	$0.286^{+0.032}$	$0.261^{-0.007}$
		5-shot	0.314	$0.326^{+0.012}$	$0.342^{+0.028}$	$0.334^{+0.020}$	$0.318^{+0.004}$
WiKi-One	Methods	Shot	Vanilla	Intra-Task		Inter-Task	
				Query	Support	Inverse	Interpolation
Metric-based	Gmatching	1-shot	0.200	$0.234^{+0.034}$	$0.224^{+0.024}$	$0.218^{+0.018}$	$0.215^{+0.015}$
		5-shot	0.245	$0.278^{+0.033}$	$0.263^{+0.018}$	$0.261^{+0.016}$	$0.256^{+0.011}$
	FSRL	1-shot	0.128	$0.157^{+0.029}$	$0.155^{+0.027}$	$0.136^{+0.008}$	$0.147^{+0.019}$
		5-shot	0.158	$0.186^{+0.028}$	$0.176^{+0.018}$	$0.171^{+0.013}$	$0.165^{+0.007}$
	FAAN	1-shot	0.272	$0.301^{+0.029}$	$0.285^{+0.013}$	$0.289^{+0.017}$	$0.279^{+0.007}$
		5-shot	0.341	$0.358^{+0.025}$	$0.349^{+0.008}$	$0.353^{+0.012}$	$0.348^{+0.007}$
Meta-based	MetaR (Pre-Train)	1-shot	0.314	$0.328^{+0.014}$	$0.335^{+0.021}$	$0.325^{+0.011}$	$0.319^{+0.005}$
		5-shot	0.323	$0.334^{+0.011}$	$0.347^{+0.024}$	$0.328^{+0.005}$	$0.331^{+0.008}$
	MetaR (In-Train)	1-shot	0.193	$0.198^{+0.005}$	$0.207^{+0.014}$	$0.190^{-0.003}$	$0.184^{-0.009}$
		5-shot	0.221	$0.232^{+0.011}$	$0.239^{+0.018}$	$0.227^{+0.006}$	$0.209^{-0.012}$
	GANA	1-shot	0.261	$0.272^{+0.011}$	$0.286^{+0.025}$	$0.266^{+0.005}$	$0.273^{+0.012}$
		5-shot	0.322	$0.338^{+0.016}$	$0.342^{+0.020}$	$0.331^{+0.009}$	$0.327^{+0.005}$

Table 1: Evaluation MRR of FKGC models with all augmentation methods on NELL-One and Wiki-One.

Dataset	#Ent.	#Rel.	#Triples	#Tasks
NELL-One	68,545	358	181,109	67
WiKi-One	4,838,244	822	5,859,240	183

Table 2: Statistics of datasets. Each column respectively represents the number of entities, relations, triples and tasks.

ues on both datasets upon all metrics. The experimental results indicate that our augmentation methods are effective for improving the existing FKGC models.

2. After support augmentation on NELL-One data, MetaR (Pre-Train) has increased by 38.4% compared to the original model, which is the largest increase. On WiKi-One data, query augmentation improves the MRR value of FSRL by 22.7%. The improvement on NELL-One is larger than that on WiKi-One because the Wiki dataset is more extensive, so the improvement brought by data augmentation is limited.

3. On the NELL-One dataset, intra-task augmentation is better than inter-task augmentation on metric learning-based models, but the opposite is true on meta learner-based models. On the WiKi-One dataset, intra-task augmentation outperforms inter-augmentation on all FKGC models. We conjecture the reason lies in that the WiKi-One dataset has more tasks than NELL-One, therefore increasing the number of triples within a task is more effective than increasing the number of tasks.

5.3 Combining Augmentations

After studying each mode of data augmentation individually, we combine intra-task augmentation and inter-task augmentation to understand the interplay between these two levels of augmentation methods. We select the best-performing FAAN model among metric learning-based methods for experiments. As shown in Table 4, the augmented

Model: MetaR (Pre-Train)									
NELL-One		MRR		Hits@10		Hits@5		Hits@1	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Vanilla		0.164	0.209	0.331	0.355	0.238	0.280	0.093	0.141
Intra-Task	Query	0.204	0.224	0.376	**0.383**	0.295	0.298	0.131	0.149
	Support	**0.227**	**0.240**	**0.380**	0.376	**0.303**	**0.323**	**0.161**	0.157
Inter-Task	Inverse	0.217	0.233	0.375	0.359	0.289	0.296	0.156	**0.172**
	Interpolation	0.194	0.217	0.379	0.359	0.284	0.289	0.100	0.158
Wiki-One		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Vanilla		0.314	0.323	0.404	0.418	0.375	0.385	**0.266**	**0.270**
Intra-Task	Query	0.328	0.334	**0.515**	0.504	0.437	0.442	0.233	0.242
	Support	**0.335**	**0.347**	0.509	0.507	**0.447**	**0.451**	0.235	0.256
Inter-Task	Inverse	0.325	0.328	0.499	**0.513**	0.433	0.436	0.233	0.237
	Interpolation	0.319	0.331	0.500	0.509	0.426	0.444	0.223	0.235

Table 3: Evaluation results of MetaR (Pre-Train) with data augmentation on NELL-One and Wiki-One.

mode	1-shot	5-shot
FAAN	0.194	0.279
+Query, Inverse	$0.240^{+0.046}$	$0.307^{+0.028}$
+Query, Interp.	$0.225^{+0.031}$	$0.297^{+0.018}$
+Support, Inverse	$0.221^{+0.027}$	$0.288^{+0.009}$
+Support, Interp.	$0.217^{+0.023}$	$0.286^{+0.007}$

Table 4: MRR results of FAAN combining augmentations variants on NELL-One dataset. Interp. denote interpolation.

model outperforms the original one under all settings. Combined with Figure 2, we find the jointly augmented models achieve better results than models using only one augmentation method. It shows that the combination of inter-task augmentation and intra-task augmentation is able to further improve the results.

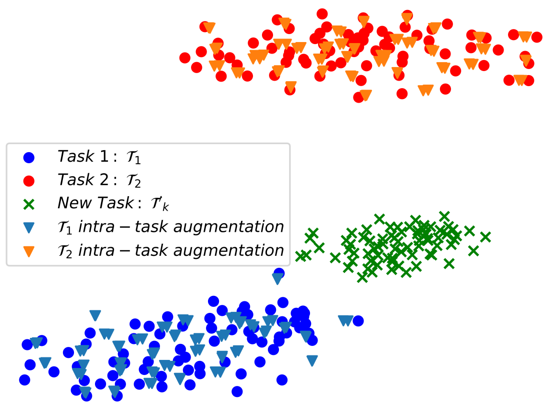

Figure 2: Visualization of relation vectors generated by different augmentation methods

5.4 Hits@N for case study

MetaR improves the most with all augmentation methods, and to get a complete picture of its performance; we further analyze it using Hits@N, which is summarized in Table 3. The augmentation methods bring the greatest improvement on Hits@5 and Hits@10, indicating that the augmentation methods mainly rely on the top-ranked recall to improve the overall MRR value. The Hit@N of the WiKi-One dataset is generally better than that of the NELL-One dataset under the same settings, because the former has more data to train.

5.5 Visualization

To better demonstrate the effectiveness of our augmentation methods, we visualize the new relation vectors generated by various augmentation methods in a 2-dimensional plane, i.e., using t-SNE (Van der Maaten and Hinton, 2008) for dimension reduction. As shown in figure 2, a new task is generated using interpolation augmentation, which can be well distinguished from other tasks and has a small intra-task distance. Intra-task augmentation for existing tasks can generate more relational vectors within the cluster. Therefore, the visualization results validate the effectiveness of our data augmentation method for FKGC.

5.6 Results on Different Relations

In addition to evaluating the augmented performance of all models, we also conduct experiments with FSRL on the NELL-One test data to evaluate the performance of each task relation. Table 5 reports the original performance of FSRL and the MRR after our augmentation methods. It can be seen from the table that no matter which augmentation method is used, the variance of the results is high in different task relations. The main reason for this is that the number of candidate entities is different, and large candidate sets make prediction difficult. Nonetheless, our augmentation methods outperform the baseline results on all relations, especially the interpretation augmentation method performs best on the FSRL model, indicating that our augmentation methods are robust to different task relations.

R-ID	Vanilla	Intra-task		Inter-task	
		Query	Support	Inver.	Interp.
1	0.975	0.982	0.980	0.982	**0.983**
2	0.064	0.072	0.068	0.070	**0.085**
3	0.472	0.601	0.602	0.595	**0.610**
4	0.005	0.008	0.007	0.008	**0.011**
5	0.210	0.242	0.232	0.268	**0.272**
6	0.045	0.048	0.047	0.049	**0.063**
7	0.141	0.163	0.149	0.156	**0.231**
8	0.118	0.121	0.123	**0.133**	0.128
9	0.561	0.562	0.550	0.566	**0.586**
10	0.009	0.011	0.010	0.012	**0.023**
11	0.373	0.397	0.378	0.394	**0.427**

Table 5: FSRL mrr results with 5-shot reference decomposed over different relations in NELL-One test dataset. R-ID denote relation id, Inver. and Interp. denotes Inverse augmentation and Interpolation augmentation.

6 Conclusion

To alleviate the limited data problem in the FKGC task. In this paper, we propose to utilize the data augmentation technique to enrich the training set for FKGC models. Specifically, we design the data augmentation method from hierarchical perspectives. The inter-task perspective generates new tasks for the FKGC task, while the intra-task perspective provides more entity pairs for each task. Furthermore, in order to fully perform data augmentation, we design two augmentation methods for each perspective, i.e., inverse augmentation and interpolation augmentation for the inter-task view, query augmentation and support augmentation for the intra-task view. Experimental results validate the effectiveness of the proposed method.

Acknowledgements

The research work is supported by the National Natural Science Foundation of China under Grant No.62206266. Zhao Zhang is supported by the China Postdoctoral Science Foundation under Grant No. 2021M703273.

References

Christopher M. Bishop. 1995. Training with noise is equivalent to tikhonov regularization. *Neural Computation*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. *international conference on management of data*.

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

Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019. Meta relational learning for few-shot link prediction in knowledge graphs. *arXiv preprint arXiv:1909.01515*.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay K. Vasudevan, and Quoc V. Le. 2019. Autoaugment: Learning augmentation strategies from data. *computer vision and pattern recognition*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. *arXiv: Learning*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *meeting of the association for computational linguistics*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. *meeting of the association for computational linguistics*.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup.

George A. Miller. 1992. Wordnet: a lexical database for english. *human language technology*.

Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka, Partha Pratim Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, T. Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, M. Greaves, and J. Welling. 2018.

Never-ending learning. *Communications of The ACM*.

Guanglin Niu, Yang Li, Chengguang Tang, Ruiying Geng, Jian Dai, Qiao Liu, Hao Wang, Jian Sun, Fei Huang, and Luo Si. 2021. Relational learning with gated and attentive neighbor aggregator for few-shot knowledge graph completion. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 213–222.

Apoorv Saxena, Aditay Tripathi, and Partha Pratim Talukdar. 2020. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. *meeting of the association for computational linguistics*.

Jiawei Sheng, Shu Guo, Zhenyu Chen, Juwei Yue, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2020. Adaptive attentional network for few-shot knowledge graph completion. *arXiv preprint arXiv:2010.09638*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. *the web conference*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *international conference on learning representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, Aaron Courville, David Lopez-Paz, and Yoshua Bengio. 2018. Manifold mixup: Better representations by interpolating hidden states. *international conference on machine learning*.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of The ACM*.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. *arXiv preprint arXiv:1808.09040*.

Chuanguang Yang, Zhulin An, Helong Zhou, Linhang Cai, Xiang Zhi, Jiwen Wu, Yongjun Xu, and Qian Zhang. 2022. Mixskd: Self-knowledge distillation from mixup for image recognition. In *European Conference on Computer Vision*.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. *international conference on computer vision*.

Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. 2020. Few-shot knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3041–3048.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2017. mixup: Beyond empirical risk minimization. *Learning*.