An Evaluation Study of Generative Adversarial Networks for Collaborative Filtering

Fernando B. Pérez Maurera[1,2] (✉)[0000−0001−6578−7404], Maurizio Ferrari Dacrema[1][0000−0001−7103−2788], and Paolo Cremonesi[1][0000−0002−1253−8081]

[1] Politecnico di Milano, Milan, Italy
{fernandobenjamin.perez,maurizio.ferrari,paolo.cremonesi}@polimi.it
[2] ContentWise, Milan, Italy
fernando.perez@contentwise.com

Abstract. This work explores the reproducibility of CFGAN. CFGAN and its family of models (TagRec, MTPR, and CRGAN) learn to generate personalized and fake-but-realistic rankings of preferences for top-N recommendations by using previous interactions. This work successfully replicates the results published in the original paper and expands the experimental analysis comparing CFGAN against a selection of simple and well-known properly optimized baselines, observing that CFGAN is not consistently competitive against them despite its high computational cost. Then, it discusses and analyzes the impact of certain differences between the CFGAN framework and the model used in the original evaluation, such as the absence of random noise and the use of real user profiles as condition vectors, which leave the generator prone to learn a degenerate solution in which the output vector is identical to the input vector. To ensure the reproducibility of these analyses, this work describes the experimental methodology and publishes all datasets and source code.

Keywords: Generative Adversarial Networks · Recommender Systems · Collaborative Filtering · Reproducibility

1 Introduction

In recent years, Generative Adversarial Networks (GANs) have become the state-of-the-art technique inside the group of generative methods, i.e., methods that learn how to generate fake data from the real one. Their primary use have been in the compute vision domain [14,17,18,19]. They have also been used in Information Retrieval [36] and Recommender Systems, being the most notable example Collaborative Filtering GAN (CFGAN) [4], and the family of models based on it, such as TagRec [5], CRGAN [37], MTPR [38], and CFGAN for service recommendations [39].

This work contributes to the rising trend of evaluation studies in Machine Learning, Information Retrieval, and Recommender Systems domains. Other works cover different aspects of reproducible evaluations and domains, such theoretical assessments [12], the need for standardized benchmarks, methodological issues and weak evaluation procedures. [2,10,11,32,33], and others [9,16,24,25,29].

In particular, this work analyzes the replicability, reproducibility, and recommendation quality of CFGAN [4] as well as its numerical stability which is known to be a challenge for GANs [9,24], and a possible early-stopping criteria. Furthermore, it discusses the implications of certain differences between the CFGAN framework and the model that was used in the experimental evaluation, which would adversely affect its learning ability, providing a reference for future works. This discussion is based on the findings of [22], which highlights the importance of describing not only *how* models works, but also *what* works and *why* it works, as well as how experimental inquiries that aim to deepen our understanding are valuable research contributions even when no new algorithm is proposed. The main research questions of this work are:

RQ1: Is CFGAN replicable and numerically stable? i.e., does CFGAN achieve the claimed results using the same experimental setup as in [4]?

RQ2: Is CFGAN reproducible, achieving the claimed recommendation quality when compared to properly-tuned baselines? How does CFGAN compare along other dimensions such as beyond-accuracy and scalability metrics?

RQ3: What is the impact of the differences between the CFGAN framework and the model used for the evaluation in [4], and why they raise theoretical and methodological concerns regarding the learning ability of the model?

2 Collaborative Filtering Generative Adversarial Networks

GANs have been successfully applied to numerous prediction and classification tasks. This work addresses a family of generative models originated from GANs used in Recommender Systems. Briefly, a GAN[3] consists of two neural networks that are trained together in an adversarial setting until until they reach convergence. The first neural network is called the *generator*, denoted as G, while the second network is called the *discriminator*, denoted as D [3,8,13,14]. CFGAN[4] is the most notable GAN method that is used in Recommender Systems [5,39]. Its main attribute is that it generates personalized user or item profiles, mainly by solely using previous interactions, but is able to learn from sources of information as well [4].

CFGAN Training Process Figure 1 shows an illustration of the training process of CFGAN. Every epoch starts by feeding the generator (G) with random noise ($\boldsymbol{z}$) and a condition vector ($\boldsymbol{c}$). The generator creates preferences of users towards items (or viceversa) (GP) which are then masked (MP). The discriminator (D) then receives the real profiles (RP), the masked profiles, and the condition. The discriminator tells the probability that each MP and RP come from the real data. The discriminator is updated based on how well it is able to correctly distinguish fake data from real data. The generator is updated based on how much it could generate *fake but realistic* data.

[3] The supplemental material contains the formal formulation of GANs.

[4] For a detailed explanation of CFGAN we refer the reader to the reference article [4].

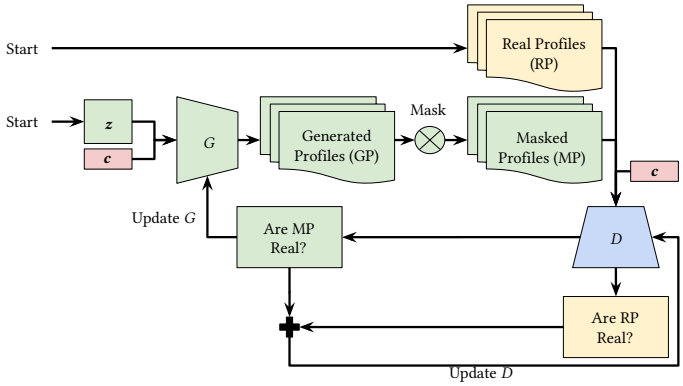

Fig. 1: Training process of CFGAN. G, D, $\boldsymbol{z}$ and $\boldsymbol{c}$ are the generator network, discriminator network, random noise, and condition vectors, respectively.

Modes CFGAN has two modes: *user-based* (u) or *item-based* (i). The first learns to generate user profiles, while the second learns to generate item profiles.

Masks CFGAN applies a mask to the generated profiles by performing an element-wise product between these and the real profiles. If the variant is *Partial Masking*, then the mask changes (see Variants).

Architecture Both the generator and discriminator of CFGAN are fully connected feed-forward neural networks independent from each other where each has its own hyper-parameters, e.g., number of hidden layers, learning rate, regularization, and others. If the *mode* is *user-based*, then the number of input neurons is the number of *items* in the dataset. Conversely, the number of input neurons for an *item-based* CFGAN is the number of *users* in the dataset.

Recommendations In a top-N item recommendation scenario, the trained generator creates user profiles containing the preference scores of users toward items. Recommendations are built by ranking the items from the highest to lowest score and selecting the top-N.

Variants CFGAN has three variants:

- *Zero Reconstruction (ZR)*: Changes the loss function of the generator. It ensures that a sample of non-interacted items are given zero-weights in the generated profiles.
- *Partial Masking (PM)*: The mask applied to the generated profiles combines the user profile and randomly-chosen unseen items.
- *Zero Reconstruction and Partial Masking (ZP)*: Combines ZR and PM.

3 CFGAN Theoretical and Methodological Questions

In addition to the replication and reproduction discussion of CFGAN, this work highlights key differences between the initial description of CFGAN and the final model that is used in the experimental evaluation of that same paper [4]. These differences were not discussed in the original paper but have significant implications on the model's ability to learn user or item preferences.

3.1 Real Profiles as Condition Vectors

What raises concerns? In the experimental evaluation of CFGAN, the condition vector provided to both the generator and the discriminator is the real user/item profile, i.e., the interactions that CFGAN is learning to generate.

Why is it a concern? As a consequence, CFGAN is prone to generate a trivial solution. The generator could learn the identity function between the condition vector and the output, therefore easily deceiving the discriminator without learning to generate new profiles. On the other hand, the discriminator could learn that the generated user profile should be identical to the condition vector to be real, again learning a trivial function. In practice, this will push the generator to behave as an *autoencoder* [21], which reconstructs as output the same input (condition) it was provided with.

How to avoid this concern? Since the condition vector can contain any information, a simple strategy would be to use other feature data related to the items or users or other contextual information. In a pure collaborative recommendations scenario, where no features or context is available, a possible strategy is to change the condition vector to be the user/item classes depending on the CFGAN mode. This decision is aligned with previous works on GANs [26]. In Recommender Systems, using the user/item classes provides a mechanism to generate *personalized recommendations* to every user. In contrast to the original CFGAN reference, using the user/item classes excludes the possibility that the generator and discriminator learn a trivial solution.

3.2 No Random Noise

What raises concerns? In the experimental scenario of the reference article, it is stated that the random noise is not provided as input to the generator because the goal is to generate the single best recommendation list rather than multiple ones.

Why is it a concern? This violates the framework defined in the same article and the design principles of GANs. In practice, discarding noise is problematic because it drastically reduces the input space and the generator will be trained

on a very sparse set of user profiles. This assumes that the user profiles will not change or that they will be the same that were used to train the model itself, which will make CFGAN non-robust to problems like *dataset shift* [30,27] and impractical in a real application.

How to avoid this concern? Feed the generator with a random noise vector $\boldsymbol{z}$ and the condition vector $\boldsymbol{c}$. $\boldsymbol{z}$ is drawn from a normal distribution with zero mean and unit variance, i.e., $\boldsymbol{z} \sim \mathcal{N}(\mu, \sigma^2)$ where $\mu = 0$ and $\sigma^2 = 1$ as suggested by other works [14,26]. The size of $\boldsymbol{z}$ is a key element while training GANs. However, previous works do not have consensus concerning the size of $\boldsymbol{z}$ [8]. We use a heuristic to set the size of the random vector and try different values depending on the number of input neurons: 50%, 100%, or 200% of them. In practice, the condition $\boldsymbol{c}$ and the random vector $\boldsymbol{z}$ are concatenated, and this new vector becomes the input to the first layer of the generator network.

3.3 Methodological Questions

What raises concerns? From the description of CFGAN, it is not stated how the number of training epochs is chosen and which stopping criterion is adopted for the training phase.

Why is it a concern? Two key methodological aspects for most machine learning models are the number of training epochs and stopping criteria. With the current CFGAN formulation, these two are left to be defined by hand instead of automatically chosen by the continuous evaluation of CFGAN, which might lead to a non-optimal model, misuse computational resources, and negatively affect the replicability of the published results In the Recommender Systems domain, there are well-known objective ways to measure recommendation quality in offline scenarios, e.g., with accuracy metrics, and these can be used to avoid human intervention.

How to avoid this concern? Use an early-stopping mechanism based on the one used in previous work for other machine learning recommenders, such as matrix factorization or linear regression [10,11]. The early-stopping mechanism periodically evaluates CFGAN on validation data while CFGAN is being trained on train data. The training stops when the CFGAN quality does not improve over the best evaluation for a fixed number of evaluations.

4 Experimental Methodology

The experiments, results, and discussion are based on one of the following two experiments: (1) execution of the source code provided in the CFGAN reference article as-is to asses the result replicability; (2) hyper-parameter tuning of different recommenders using a well-known evaluation framework to study

the reproducibility of the results and evaluate along different dimensions (see [10,11]). We release the source code of our experiments online[5].

Datasets The experiments use the same datasets[6] (a sub-sampled version of Ciao[7] [4,35], ML100K [15], and ML1M [15]) and splits (*training* and *testing*) provided with the CFGAN reference article [4]. For scenarios that required a *validation* split, we created one by applying the same strategy as the reference: random holdout at 80% of the *training* split. Given the modest size of these datasets, all experiments are done on the CPU.

Technologies The implementation of all experiments, is based on the evaluation framework published in [10]. This framework includes the implementation of some simple yet competitive state-of-the-art baselines for Recommender Systems. For the replication study the original implementation has been used as provided. For the reproducibility study and the other experiments the original CFGAN source code has adapted to the framework with no changes to the core algorithm.

4.1 Methodology for the Replicability of CFGAN

The original CFGAN source code includes the implementation of CFGAN and its training loop using a fixed set of hyper-parameters that are dataset-dependent. The training procedure is the following: it fits a CFGAN recommender using the *train* split of the selected dataset and evaluates the recommender using the *test* split. With respect to the evaluation metrics, this source code evaluates CFGAN on *accuracy* metrics: precision (PREC), recall (REC), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG) at recommendation list length 5 and 20. The limitations of this source code are the lack of the implementation of the baselines and of the hyper-parameter tuning of all recommenders, e.g., baselines and CFGAN. Due to this, the replication study is only possible for CFGAN.

4.2 Methodology for the Reproducibility of CFGAN

The reproducibility study expands the original CFGAN evaluation by including: (i) new baselines that were shown to provide high recommendation quality; (ii) a well-defined hyper-parameter optimization strategy; (iii) a well-defined early-stopping strategy; and (iv) a comparison along both accuracy, beyond-accuracy and scalability metrics.

In particular, the goal of (i) and (ii) is to asses the recommendation quality of CFGAN under a wider set of recommendation models which are properly tuned

[5] Supplemental material URL: https://bit.ly/3AVD4Yl

[6] The Watcha[4] dataset was not provided with the reference article.

[7] The reference article does not provide instructions to reproduce this version of the dataset. We contacted the authors for clarifications but did not receive a reply.

under the same conditions. The models we report range from non-personalized, neighborhood-based, and non-neural machine learning approaches. This decision is aligned with results obtained by previous evaluation studies in the domain [11,10]. Regarding the hyper-parameter optimization of CFGAN it should be noted that the search-space described in the reference article, considering that it is done using a grid-search, contains more than $3 \cdot 10^8$ cases, which cannot be reproduced in a reasonable time. Due to this, this work adopts a different optimization strategy: Bayesian Search as used in [10]. The hyper-parameter ranges and distributions of CFGAN are reported in Table 1. The Bayesian Search starts with 16 initial random searches and performs a total of 50 cases for each algorithm. Each model in this search is fit with the *training* split and evaluated against the *validation* one. The best hyper-parameters are chosen as those with highest NDCG at 10. Once the optimal hyper-parameters set is chosen, it trains the final models using this set and the union of the training and validation splits, evaluating the final models against the test set.

Evaluation metrics Recommenders are evaluated using the original accuracy metrics (PREC, REC, MRR, and NDCG) and against the following*beyond-accuracy* metrics: novelty [40], item coverage (Cov. Item, quota of recommended items), and distributional diversity (Div. MIL [40] and Div. Gini[1]). Using these new metrics provide a broader picture of the quality of all recommenders.

Baselines Due to space limitations, this work provides only a list of the baseline recommenders. A thorough description of all baselines, the list of their hyper-parameters of each baseline, and their range and distribution are in [10]. The baselines list is the following: **Top Popular** [10] as a non-personalized approach. **UserKNN CF** and **ItemKNN CF** [10] as neighborhood-based CF (similarities: cosine, dice, jaccard, asymmetric cosine, and tversky) and shrinkage term. **RP3beta** [6] as a graph-based approach. **PureSVD** [7] and **MF BPR** [31] as matrix factorization models. **SLIM ElasticNet** [10,28] as a machine learning approach. Lastly, **EASE R** as a fast linear auto-encoder [34].

CFGAN recommenders The hyper-parameter tuning is done on a total of 18 different CFGAN models: three datasets (Ciao, ML100K, and ML1M), two modes (tem-based i and user-based u), and three variants (ZR, PM, and ZP).

To ensure a clear stopping criteria and a fair training for CFGAN, it is trained using the early-stopping criteria defined in [10] and presented in Section 3. The number of minimum and maximum epochs is in Table 1. The early-stopping selects the best number of epochs by using the validation data. The optimal number of epochs is used to train the final model. We recall that the original description of CFGAN *does not provide* an early-stopping mechanism.

Table 1 lists all hyper-parameters of CFGAN, where hyper-parameters like *optimizer*, *activation* are left unchanged with respect to the reference article. Apart from the number of training epochs, the optimizer, and activation, the rest of the hyper-parameters are set by the Bayesian Search.

Table 1: Hyper-parameters for CFGAN. These are divided in two groups. The first group contains specific hyper-parameters of CFGAN. The second group are hyper-parameters of the generator and discriminator neural networks, values between networks can be different.

Hyper-Parameter	Type	Range	Distribution
# of Epochs	Integer	$200 - 400^a$	early-stopping
ZR Coefficient	Real	$0 - 1$	uniform
ZR Ratio	Integer	$10 - 90$	uniform
PM Ratio	Integer	$10 - 90$	uniform
# of Hidden Layers	Integer	$1 - 4$	uniform
# of Hidden Features	Integer	$50 - 300$	uniform
# of Steps	Integer	$1 - 4$	uniform
$l2$ Regularization	Real	$1 \cdot 10^{-4} - 1 \cdot 10^{-1}$	log-uniform
Learning Rate	Real	$1 \cdot 10^{-4} - 5 \cdot 10^{-3}$	log-uniform
Batch Size	Integer	$32 - 256$	uniform
Optimizer	Categorical	ADAM [20]	-
Activation	Categorical	sigmoid	-

[a] Due to how the training is performed, this range is close to the 1.000 and 1.500 epochs used in the reference article

5 Experiments Results & Discussion

In this section we report the results and discuss the different aspects and experiments on CFGAN.

5.1 RQ1: CFGAN Replicability & Numerical Stability

To address RQ1 we report the results of the replication study, as described in Section 4.1, by using the original source code and data. This experiment has two goals: (i) verify that published results are replicable, (ii) measure the numerical stability of CFGAN given the stochastic nature of its architecture [24,9].

Table 2 shows the results of the experiment, we only report two metrics due to space limitations. [8] The results reported in the reference article are denoted as *Reference*. Due to the stochastic nature of CFGAN models, we do not expect to achieve *exact* numerical replicability. For all datasets we see that the replicated results are *lower* than those reported in the reference article. For the ML1M dataset, the difference between the average and reported NDCG is of -0.59%. On the smaller ML100K, the results are more varied: -2.60% between the average and reported NDCG. For the Ciao dataset, the results could not be replicated due to two factors (i) the original source code trained a different variant (iZR) than the reported in the reference article (iZP) and (ii) lack of reproducible hyper-parameters sets for this dataset in the reference article. Lastly, with respect

[8] A table with all metrics is available in the supplemental materials of this work.

Table 2: Comparison between the accuracy metrics in the reference article [4] and those obtained in the replicability experiment (see Section 5.1) at recommendation list length of 20. Statistics calculated over 30 executions, evaluating on the last epoch using recommendation lists of length 20. We consistently obtain *lower* results across two of the three datasets on average. For the Ciao dataset, the original source code trains a different variant (in bold) than the reported in the reference article.

Dataset	Variant	Stats	PREC	NDCG
Ciao	**iZR**	Mean ± Std	0.0402 ± 0.0013	0.1144 ± 0.0042
	iZP	Reference [4]	0.0450	0.1240
ML100K	iZP	Mean ± Std	0.2864 ± 0.0022	0.4217 ± 0.0043
	iZP	Reference [4]	0.2940	0.4330
ML1M	iZP	Mean ± Std	0.3077 ± 0.0007	0.4036 ± 0.0012
	iZP	Reference [4]	0.3090	0.4060

to the numerical stability, under 30 executions of this replication, the results indicate that the reference implementation of CFGAN is numerically stable.

5.2 RQ2: Reproducibility Evaluation Against Properly-Tuned Baselines

To address RQ2 we report the recommendation quality of CFGAN and baseline recommenders using a Bayesian hyper-parameter tuning approach, as described in Section 4.2. The goal is to evaluate [4] on the same top-N recommendation scenario of the reference paper against a set of properly-tuned baselines on accuracy and beyond-accuracy metrics and study if published results are reproducible.

Table 3 shows the results of accuracy and beyond-accuracy metrics of properly-tuned recommenders. Due to space constraints, the focus of this discussion is on the dataset with the highest number of interactions studied in the reference article [4], i.e., ML1M. The results obtained with other datasets are comparable. [9]

The results indicate that CFGAN is outperformed by three simple baselines in NDCG, sometimes by almost 10%, in particular by other autoencoder based recommendation models like EASE R and SLIM Elastic Net. These findings are consistent to those reported in several other evaluation studies [11,10,12,23,2]. The accuracy across CFGAN models varies depending on the CFGAN mode and variant. For instance, the most and least accurate variant are uZR and iZP, respectively, with approximately $21,76\%$ difference in their NDCG metrics. Under the current methodology we cannot confirm the claim that item based models or ZP variants outperform other variants, as indicated in the reference article [4]. In fact, our most accurate variant is uZR. When looking at beyond-accuracy metrics, item-based CFGAN models have equal or higher diversity than baselines. In

[9] The full results are in the additional material.

Table 3: Accuracy and beyond-accuracy metrics for tuned baselines and CFGAN on the ML1M dataset at recommendation list length of 20. Higher accuracy values than CFGAN models reached by baselines in bold. ItemKNN and UserKNN use asymmetric cosine. CFGAN results are different than Table 2 due to the hyper-parameter tuning.

	PREC	REC	MRR	NDCG	Novelty	Cov. Item	Div. MIL	Div. Gini
Random	0.0099	0.0056	0.0326	0.0108	0.0732	1.0000	0.9946	0.8977
TopPop	0.1552	0.1146	0.3852	0.1938	0.0473	0.0299	0.4529	0.0095
UserKNN CF	0.2891	**0.2570**	**0.6595**	**0.3888**	0.0513	0.3286	0.8921	0.0655
ItemKNN CF	0.2600	0.2196	0.6254	0.3490	0.0497	0.2097	0.8148	0.0362
RP3beta	0.2758	0.2385	**0.6425**	0.3700	0.0506	0.3427	0.8565	0.0528
PureSVD	0.2913	0.2421	**0.6333**	0.3783	0.0516	0.2439	0.9142	0.0712
SLIM ElasticNet	**0.3119**	**0.2695**	**0.6724**	**0.4123**	0.0514	0.3153	0.8984	0.0696
MF BPR	0.2485	0.2103	0.5753	0.3242	0.0512	0.3126	0.8855	0.0631
EASE R	**0.3171**	**0.2763**	**0.6795**	**0.4192**	0.0518	0.3338	0.9146	0.0803
CFGAN iZR	0.2862	0.2547	0.6312	0.3770	0.0542	0.4123	0.9583	0.1459
CFGAN iPM	0.2505	0.1950	0.5454	0.3138	0.0523	0.3669	0.9218	0.0901
CFGAN iZP	0.2407	0.1742	0.5230	0.2972	0.0530	0.4894	0.9256	0.0901
CFGAN uZR	0.2955	0.2473	0.6222	0.3799	0.0523	0.2167	0.9205	0.0837
CFGAN uPM	0.2367	0.1928	0.5513	0.3054	0.0516	0.1782	0.8962	0.0550
CFGAN uZP	0.2764	0.2342	0.6208	0.3620	0.0513	0.1833	0.9062	0.0617

particular, iZR has the highest novelty, item coverage, and distributional diversity, while also being the second-most accurate variant with respect to NDCG. User-based CFGAN models have less coverage than all baselines.

It can be seen that the results of the replicability study using hyper-parameter optimization and early-stopping reported in Table 3 are lower then those reported in the replication study in Table 2. This indicates that the non-reproducible hyper-parameter search and early-stopping criteria have an important impact on the recommendation quality. As a last observation, using the results reported in the reference article CFGAN would not be competitive against the baselines.

Scalability Concerning the recommendation time, all algorithms are able to create recommendations lists to all users in a total time between 7 and 20 seconds. Differently from other neural models [10], CFGAN models provide fast recommendations due to the lack of random noise, consequently, they generates static recommendation lists.

Concerning the training time, CFGAN models take more time to train than any baseline. We categorize models into three groups: (i) ItemKNN, UserKNN, PureSVD, RP3beta, and EASE R take between 2 and 25 seconds on average. (ii) Machine learning approaches, i.e., SLIM and MF BPR take between 3 and 9 minutes to train on average. (iii) All CFGAN models take between 25 and 40

Table 4: Accuracy and beyond-accuracy values for different CFGAN models for the ML1M dataset at recommendation list length of 20. The suffix RN-X means that the model uses random noise of size X. The suffix Class indicates that the model uses the user/item class as the condition vector. The suffix NO-ES indicates that the model does not use early-stopping. The suffix Reference is the model in the reference article. Hyper-parameter sets of variants are the same as those in Table 3 except for those with the Reference suffix. – denotes non published values in the reference article.

Variant	PREC	NDCG	Cov. Item	Variant	PREC	NDCG	Cov. Item
iZP Reference [4]	0.3090	0.4060	–	uZP Reference [4]	–	–	–
iZP Table 3	0.2407	0.2972	0.4894	uZP Table 3	0.2764	0.3620	0.1833
iZP NO-ES	0.2494	0.3111	0.4041	uZP NO-ES	0.2797	0.3639	0.1882
iZP CC	0.0384	0.0507	0.0296	uZP CC	0.0916	0.1106	0.0231
iZP RN-3020	0.2059	0.2475	0.3995	uZP RN-1841	0.2737	0.3591	0.1841
iZP RN-6040	0.1683	0.2000	0.4663	uZP RN-3682	0.2781	0.3651	0.1839
iZP RN-12080	0.1304	0.1471	0.5076	uZP RN-7364	0.2759	0.3626	0.1955

minutes to train on average, even on a comparatively small dataset as ML1M, the difference in training time between the first and the last group is two orders of magnitude. Using more performing hardware, i.e., GPU could reduce this gap.

Under this offline evaluation, which is the same as in the original article [4], CFGAN does not generate more accurate recommendations than simple baselines. As CFGAN is a neural approach, bigger datasets with more complex relations between users, items, and their interactions might increase the accuracy of CFGAN. However, this is unpractical due to the higher computational cost of CFGAN models, therefore, we do not report experiments with bigger datasets.

5.3 RQ3: Impact of Theoretical and Methodological Concerns

This section reports the results of the experiments related to RQ3, those used to measure the impact of the theoretical and methodological concerns raised in Section 3. Table 4 compares the results of the reference CFGAN (denoted as Reference), the models tuned in Section 5.2 (denoted as Table 3), and the variants of this experiment.

Impact of random noise As seen in Section 2, CFGAN receives random noise as part of its input. However, in the experiments of the reference article, the random noise is removed. This experiment included three different sizes of random noise. The results indicate that the recommendation quality is slightly improved by removing the random noise, however, as stated in Section 3, it comes at the cost of risking lower generalizability and lower robustness of the

generator in a practical use case. We argue the random noise should always be present. However, we recall that doing an exhaustive analysis of the impact of random noise in GAN and CFGAN is beyond the scope of this paper.

Impact of condition vector Similarly as before, in the experiments of the reference article the condition vector is set to be the user/item profiles, which increases the risk of reaching a trivial solution. This experiment changed the condition vector to be the user/item classes. The results shows that changing the condition vector with the current CFGAN architecture dramatically lowers the model's ability to learn to generate accurate profiles. This constitutes a *negative result*, as that the current architecture does not appear to be suitable to handle the user/item classes as the condition vector. Identifying an appropriate architecture to do so and an appropriate condition vector to use in scenarios where only past user interactions are available is an open research question that goes beyond the scope of this paper.

Impact of early-stopping The reference article does not provide an early-stopping mechanism for CFGAN, although models in Recommender Systems typically benefit from one, as discussed in Section 3. This experiment removed the early-stopping and set the maximum number of epochs as 400 (this is the maximum number of epochs set for the early-stopping as seen in Table 1). Results show that using early-stopping slightly decreases the recommendation quality of CFGAN, however, we argue that the benefits of using it outweigh the downsides of it, specially if scalability is taken into account. For instance, the iZP variant trains on 645 and 1200 epochs with and without early-stopping, respectively, i.e., a decrease of 46.25% and of $4,47\%$ in training time and NDCG, respectively.

6 Conclusions

This work presents an evaluation study of the family of models of CFGAN addressing three research questions under the same top-N recommendation scenario than the reference article [4]. Are previously published results of CFGAN replicable? Are previously published results of CFGAN reproducible? What is the impact of the differences between the CFGAN framework and the model evaluated in the reference article? The experimental result indicate that CFGAN is replicable and numerically stable, but not reproducible as it can be outperformed by simple but well tuned baselines. This result adds to the recent evidence that properly tuned baselines can outperform complex methods and suggest CFGAN is not yet a mature recommendation algorithm. Regarding the model's architecture, using as condition vector the user profile leaves the model prone to a trivial and not useful solution, therefore we argue a different approach should be used. Identifying an effective strategy is still an open research question. Removing the random noise might increase the recommendation quality but the model's ability to generalize is affected. We also suggest to select the number of epochs using early-stopping to improve the model's scalability.

References

1. Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans. Knowl. Data Eng. **24**(5), 896–911 (2012). https://doi.org/10.1109/TKDE.2011.15
2. Armstrong, T.G., Moffat, A., Webber, W., Zobel, J.: Improvements that don't add up: Ad-hoc retrieval results since 1998. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management. p. 601–610. CIKM '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1645953.1646031
3. Borji, A.: Pros and cons of GAN evaluation measures. Computer Vision and Image Understanding **179**, 41–65 (2019). https://doi.org/10.1016/j.cviu.2018.10.009
4. Chae, D.K., Kang, J.S., Kim, S.W., Lee, J.T.: CFGAN: A generic collaborative filtering framework based on generative adversarial networks. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. p. 137–146. CIKM '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3269206.3271743
5. Chen, H., Wang, S., Jiang, N., Li, Z., Yan, N., Shi, L.: Trust-aware generative adversarial network with recurrent neural network for recommender systems. International Journal of Intelligent Systems **36**(2), 778–795 (2021). https://doi.org/10.1002/int.22320
6. Christoffel, F., Paudel, B., Newell, C., Bernstein, A.: Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In: Proceedings of the 9th ACM Conference on Recommender Systems. p. 163–170. RecSys '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2792838.2800180
7. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems. pp. 39–46. RecSys '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1864708.1864721
8. Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A.A.: Generative adversarial networks: An overview. IEEE Signal Processing Magazine **35**(1), 53–65 (Jan 2018). https://doi.org/10.1109/MSP.2017.2765202
9. Fellicious, C., Weissgerber, T., Granitzer, M.: Effects of random seeds on the accuracy of convolutional neural networks. In: Nicosia, G., Ojha, V., La Malfa, E., Jansen, G., Sciacca, V., Pardalos, P., Giuffrida, G., Umeton, R. (eds.) Machine Learning, Optimization, and Data Science. pp. 93–102. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-64580-9_8
10. Ferrari Dacrema, M., Boglio, S., Cremonesi, P., Jannach, D.: A troubling analysis of reproducibility and progress in recommender systems research. ACM Trans. Inf. Syst. **39**(2) (Jan 2021). https://doi.org/10.1145/3434185
11. Ferrari Dacrema, M., Cremonesi, P., Jannach, D.: Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In: Proceedings of the 13th ACM Conference on Recommender Systems. pp. 101–109. RecSys '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3298689.3347058
12. Ferrari Dacrema, M., Parroni, F., Cremonesi, P., Jannach, D.: Critically examining the claimed value of convolutions over user-item embedding maps for recommender systems. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. p. 355–363. CIKM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3340531.3411901

13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (Oct 2020). https://doi.org/10.1145/3422622
14. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems. NIPS'14, vol. 2, p. 2672–2680. MIT Press, Cambridge, MA, USA (2014)
15. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst. **5**(4) (Dec 2015). https://doi.org/10.1145/2827872
16. Hutson, M.: Artificial intelligence faces reproducibility crisis. Science **359**(6377), 725–726 (2018). https://doi.org/10.1126/science.359.6377.725
17. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5967–5976 (July 2017). https://doi.org/10.1109/CVPR.2017.632
18. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4396–4405 (June 2019). https://doi.org/10.1109/CVPR.2019.00453
19. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8107–8116 (June 2020). https://doi.org/10.1109/CVPR42600.2020.00813
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
21. Kingma, D.P., Welling, M.: An introduction to variational autoencoders. Found. Trends Mach. Learn. **12**(4), 307–392 (2019). https://doi.org/10.1561/2200000056, https://doi.org/10.1561/2200000056
22. Lipton, Z.C., Steinhardt, J.: Troubling trends in machine learning scholarship: Some ML papers suffer from flaws that could mislead the public and stymie future research. Queue **17**(1), 45–77 (Feb 2019). https://doi.org/10.1145/3317287.3328534
23. Ludewig, M., Jannach, D.: Evaluation of session-based recommendation algorithms. User Modeling and User-Adapted Interaction **28**(4–5), 331–390 (Dec 2018). https://doi.org/10.1007/s11257-018-9209-6
24. Madhyastha, P., Jain, R.: On model stability as a function of random seed. In: Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL). pp. 929–939. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/K19-1087
25. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: Statistical and machine learning forecasting methods: Concerns and ways forward. PLOS ONE **13**(3), 1–26 (03 2018). https://doi.org/10.1371/journal.pone.0194889
26. Mirza, M., Osindero, S.: Conditional generative adversarial nets. CoRR **abs/1411.1784** (2014), http://arxiv.org/abs/1411.1784
27. Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012). https://doi.org/10.1016/j.patcog.2011.06.019

28. Ning, X., Karypis, G.: Slim: Sparse linear methods for top-n recommender systems. In: Proceedings of the 2011 IEEE 11th International Conference on Data Mining. p. 497–506. ICDM '11, IEEE Computer Society, USA (2011). https://doi.org/10.1109/ICDM.2011.134
29. Peng, R.: The reproducibility crisis in science: A statistical counterattack. Significance **12**(3), 30–32 (2015). https://doi.org/10.1111/j.1740-9713.2015.00827.x
30. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N.D.: Dataset Shift in Machine Learning. The MIT Press (2009)
31. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. p. 452–461. UAI '09, AUAI Press, Arlington, Virginia, USA (2009)
32. Rendle, S., Krichene, W., Zhang, L., Anderson, J.: Neural collaborative filtering vs. matrix factorization revisited. In: Fourteenth ACM Conference on Recommender Systems. p. 240–248. RecSys '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3383313.3412488
33. Rendle, S., Zhang, L., Koren, Y.: On the difficulty of evaluating baselines: A study on recommender systems. CoRR **abs/1905.01395** (2019), http://arxiv.org/abs/1905.01395
34. Steck, H.: Embarrassingly shallow autoencoders for sparse data. In: The World Wide Web Conference. p. 3251–3257. WWW '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3308558.3313710
35. Tang, J., Gao, H., Liu, H.: Mtrust: Discerning multi-faceted trust in a connected world. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining. p. 93–102. WSDM '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2124295.2124309
36. Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., Zhang, D.: IRGAN: A minimax game for unifying generative and discriminative information retrieval models. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 515–524. SIGIR '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3077136.3080786
37. Wang, Z., Xu, Q., Ma, K., Jiang, Y., Cao, X., Huang, Q.: Adversarial preference learning with pairwise comparisons. In: Proceedings of the 27th ACM International Conference on Multimedia. p. 656–664. MM '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3343031.3350919
38. Xia, B., Bai, Y., Yin, J., Li, Q., Xu, L.: MTPR: A multi-task learning based poi recommendation considering temporal check-ins and geographical locations. Applied Sciences **10**(19) (2020). https://doi.org/10.3390/app10196664
39. Xie, F., Li, S., Chen, L., Xu, Y., Zheng, Z.: Generative adversarial network based service recommendation in heterogeneous information networks. In: 2019 IEEE International Conference on Web Services (ICWS). pp. 265–272 (July 2019). https://doi.org/10.1109/ICWS.2019.00053
40. Zhou, T., Kuscsik, Z., Liu, J.G., Medo, M., Wakeling, J.R., Zhang, Y.C.: Solving the apparent diversity-accuracy dilemma of recommender systems. Proceedings of the National Academy of Sciences **107**(10), 4511–4515 (2010). https://doi.org/10.1073/pnas.1000488107