UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Multi-Tasking Learning for Personalized Recommendation Systems

Diploma Thesis

Georgios Kirtsanis

Supervisor: Aspassia Daskalopulu

Month 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Multi-Tasking Learning for Personalized Recommendation Systems

Diploma Thesis

Georgios Kirtsanis

Supervisor: Aspassia Daskalopulu

Month 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εκμάθηση Πολλαπλών Εργασιών για Εξατομικευμένα Συστήματα Συστάσεων

Διπλωματική Εργασία

Γεώργιος Κιρτσάνης

Επιβλέπων/πουσα: Ασπασία Δασκαλοπούλου

Μήνας 2022

Approved by the Examination Committee:

Supervisor	**Aspassia Daskalopulu** Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly
Member	**Hariklia Tsalapata** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly
Member	**George Thanos** Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Aspassia Daskalopulu, for her supervision throughout my research, the guidance during my studies at the Electrical and Computer Engineering Department and the critical advises for my academic future.

I am also thankful to the entire university both my professors and the laboratory teaching staff for the important knowledge that I gained during my studies.

Lastly, I'd like to thank my family and especially my parents for keeping my motivation high throughout those years due to their emotional support.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Diploma Thesis

Multi-Tasking Learning for Personalized Recommendation Systems

Georgios Kirtsanis

Abstract

The following research introduces a new approach on deep learning and especially in the problem of recommendations. The proposed methods can be described as Multi-Tasking Learning User Modeling paradigms, whose aim are to produce efficient personalized recommendations in a set of different tasks. Recommendation Systems are a vital sector of study for researchers and because of the complexity and size of the input data, deep learning algorithms show great results. In this specific study, a wide range of data has been used, in order to train different architectures in different tasks. The data consists of users' watching sequences, interactions and personal information. The innovative dropout-like technique which is implemented, is called "Random Pruning" and is applied to one of the most efficient Multi-Tasking Learning User Representation algorithm, CONURE. The proposed methods, CONURE^{+1}, CONURE^{+2} and CONURE^{+3}, have been tested and compared with other widely used algorithms of this subject, CONURE and ADER.

Keywords

Recommendation Systems, Deep Learning, Multi-Tasking Learning, Personalized Recommendations, Transfer Learning, Continual Learning, User Modeling

Διπλωματική Εργασία

Εκμάθηση Πολλαπλών Εργασιών για Εξατομικευμένα Συστήματα Συστάσεων

Γεώργιος Κιρτσάνης

Περίληψη

Η συγκεκριμένη έρευνα εισάγει μια νέα προσέγγιση στη βαθιά μάθηση και ιδιαίτερα στο πρόβλημα των συστάσεων. Τα προτεινόμενα μοντέλα μπορούν να περιγραφεί ως ένα πρότυπο Εκμάθησης Πολλαπλών Εργασιών Μοντελοποίησης Χρηστών, στόχος των οποίων είναι να παράγουν αποτελεσματικές εξατομικευμένες συστάσεις για διαφορετικές εργασίες. Τα Συστήματα Συστάσεων είναι ένας σημαντικός τομέας μελέτης για τους μηχανικούς και λόγω της πολυπλοκότητας και του μεγέθους των δεδομένων εισόδου, οι αλγόριθμοι βαθιάς μάθησης παρουσιάζουν εξαιρετικά αποτελέσματα. Στη συγκεκριμένη μελέτη, έχει χρησιμοποιηθεί ένα ευρύ φάσμα δεδομένων, προκειμένου να εκπαιδευτούν διαφορετικές αρχιτεκτονικές σε διαφορετικές εργασίες. Τα δεδομένα αποτελούνται από ακολουθίες παρακολούθησης, αλληλεπιδράσεις και προσωπικά δεδομένα των χρηστών. Η καινοτόμος τεχνική, ονομάζεται «Τυχαίο Κλάδεμα» και εφαρμόζεται σε έναν από τους πιο αποτελεσματικούς αλγόριθμους Εκμάθησης Πολλαπλών Εργασιών Μοντελοποίησης Χρηστών, CONURE. Τα προτεινόμενα μοντέλα, CONURE^{+1}, CONURE^{+2} and CONURE^{+3}, έχουν δοκιμαστεί και συγκριθεί με άλλους πλατιά διαδεδομένους αλγόριθμους του θέματος, CONURE και ADER.

Λέξεις Κλειδιά

Συστήματα Προτάσεων, Βαθιά Μάθηση, Εκμάθηση Πολλαπλών Εργασιών, Εξατομικευμένες Προτάσεις, Μεταφορική Μάθηση, Συνεχής Μάθηση, Προτυποποίηση Χρηστών

Table of contents

List of figures

List of tables

Abbreviations

RS	Recommendation Systems
TL	Transfer Learning
CL	Continual Learning
UM	User Modeling
UR	User Representation
MTL	Multi-Tasking Learning

Chapter 1

Introduction

Nowadays, our daily lives are overwhelmed by recommendations. Playlists, advertisements, shopping, news and games are some of the recommendations that we are exposed to daily. Those recommendations are handled by “Recommendation Systems”, which are known to Artificial Intelligence researchers and enterprises. Many famous companies, such as Netflix, Amazon, Youtube, Meta, TikTok etc, use these technological techniques, in order to improve their product, as shown in Figure 1.1.

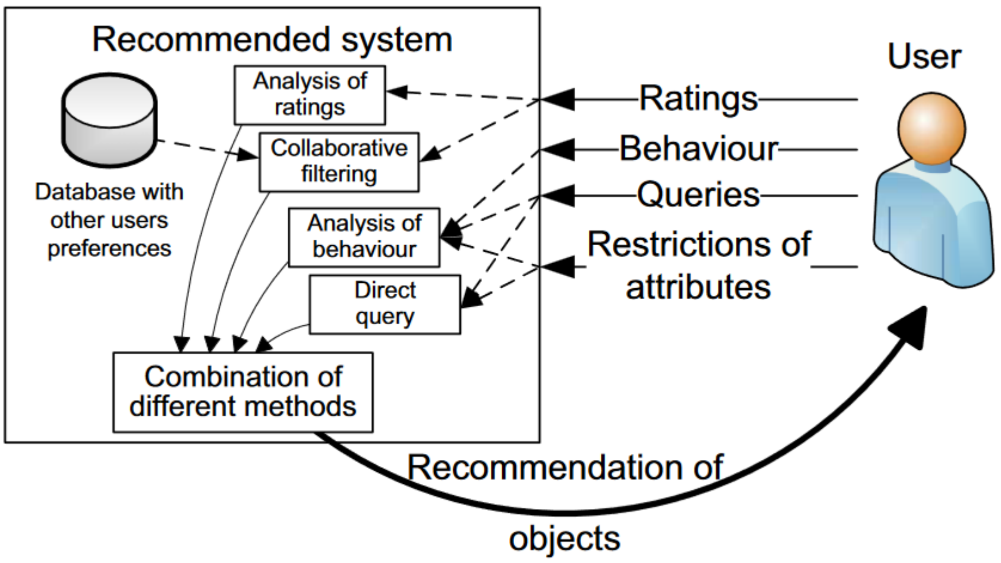
Recommended system
Analysis of ratings
Collaborative filtering
Database with other users preferences
Analysis of behaviour
Direct query
Combination of different methods
User
Ratings
Behaviour
Queries
Restrictions of attributes
Recommendation of objects

Figure 1.1: Recommendation System Overview - ”Various aspects of user preference learning and recommender systems” Paper [1]

Because of the vast flows of data, which is constantly changing and produced, Conventional Programming is not capable of handling the difficult task of recommendations. RS are models that daily process data of millions of users and produce possible recommendations for them. To successfully deal with the extensive amount of data researchers and enterprises use and develop Deep Learning algorithms to produce efficient recommendations.

Collaborative Filtering [6, 7, 8] and Content-Based Filtering [9, 10, 11] are common approaches for RS. The first one focuses on similar preferences made by corresponding users to produce recommendations, without using extra data, such as personal information. However, the second one uses trained models to produce unique recommendations based on the knowledge acquired by all kinds of data (i,e preferences and personal information of the users). The three commonly used paradigms which are discussed in this research are a) Transfer Learning, b) Continual Learning and c) User Modeling. RS can be defined between single-tasking and multi-tasking. On the given thesis, the most widely known multi-tasking recommendation algorithm CONURE[4] is improved by applying a dropout like technique to each task's mask.
Our motivation in this research, are the following two Recommendation oriented algorithms:

- **CONURE**[4], a multi-tasking, non-dynamic, user representation algorithm and the state-of-the-art model
- **ADER**[3], a single-tasking, session-based, Distilled, Exemplar Replaying algorithm

While the contribution is summarized as follows:

- Upgrading the most efficient User Representation algorithm CONURE[4]
- Introducing a dropout-like technique ”Random Pruning” on CONURE[4]'s masks
- Comparing the proposed methods CONURE^{+1}, CONURE^{+2} and CONURE^{+3} with other widely used recommendation algorithms

Chapter 2

Related Work

In this section, the related work of the given research is highly discussed.

2.1 Transfer Learning

Transfer Learning [2, 12, 13] is one of the widely known paradigms in the field of Deep Learning. During the last few years, TL is used in many different sets of problems such as Image Classification [14, 15], Time Series Forecasting [16, 17], Natural Language Processing [18, 19] and Recommendation Systems [4, 20] with excellent performance.

TL can be described, in general, as a double-staged training paradigm consisted of a) pre-training phase , where the model is trained with the whole collection of the data and b) fine-tuning phase, where the model can be trained with a part or whole collection of the data and on specific or whole part of the architecture either in the same task or in new tasks. Although a type of TL can be defined as a transferring of the knowledge of a related learned task into a newly added one.

TL is widely used in the researchers' community, thus it is able to reinforce the architecture's weights or transfer learned knowledge of previous tasks into new tasks' training. The performance comparison is plotted on Figure 2.1 and as a result, the fine-tuned models are efficiently used on multi-tasking training algorithms because of their greater performance.

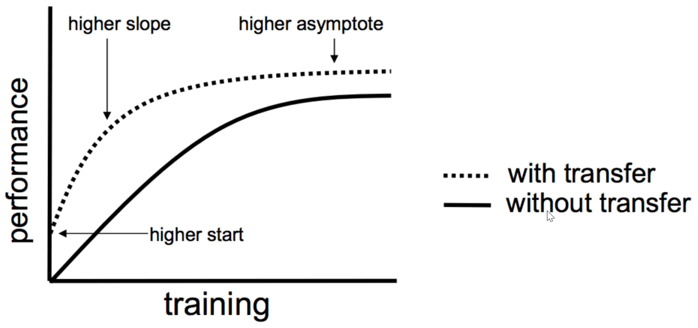

Figure 2.1: Performance Comparison of TL - TL's Paper [2]

2.2 Continual Learning

Continual Learning [4, 3, 21, 22] is a training process, in which the model is periodically trained with different sets of data throughout its lifetime (Figure 2.2). As it is stated on the previous sections, today's data flow increases rapidly. Users produce data, while using the most famous applications on the web. As a result, static training is not preferable especially in RS, in contrast to other problems, such as image classification. User's preferences are constantly changing and that's the reason why it is a challenge to implement a continual training process for our models.

Although, those algorithms are able to train the models in new sets of data, a huge disadvantage has been reported, which researchers call "catastrophic forgetting"[23, 24, 25, 25], which is referred to the ability of a model to easily learn new sets of data while tending to forget previously learned knowledge. The stated problem is an important case study for researchers, who implemented many different approaches to overcome it. **Exemplar Replay** [3, 26, 27, 28], where the model uses previous knowledge's samples in future training, **Regularization**[3, 21, 29, 30], where the model penalizes the change of the weights which are important for previously learned knowledge and **Dynamic Architectures**[31, 32], where the size of the architecture increases upon new training processes are the most commonly used approaches.

2.3 User Modeling

Over the past few years, the users have been interacting with different applications during the day. The data which is produced by those actions could be used collectively within the

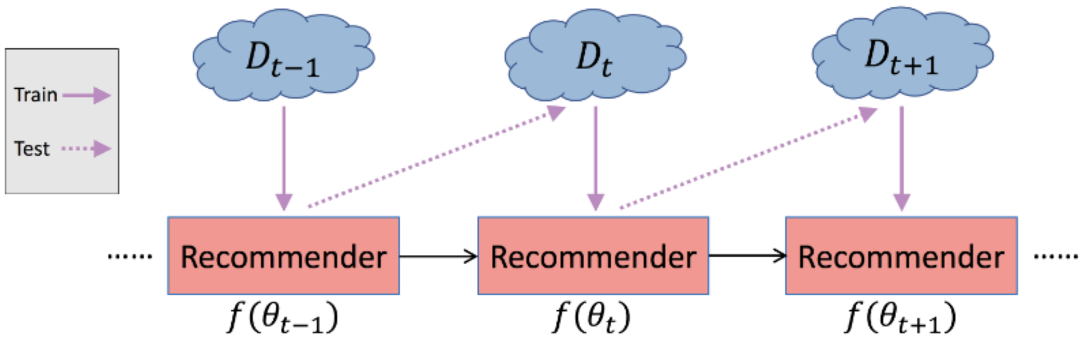

Figure 2.2: CL Overview - ADER's Paper [3]

scope of deep learning. The term "User Modeling" [4, 33, 34, 35, 36] is used by researchers for the concept of modeling user's preferences and personal data, in order to create an efficient representation of the user. Those User Representations can be later used by the Engineers, for creating RS capable of producing personalized recommendations. UR algorithms, which are defined as a sub-set of the famous MTL Algorithms[4, 25, 37], are capable of training architectures over completely different tasks. The initial paradigm was introduced on 29 Sep 2020, with the paper "One Person, One Model, One World: Learning Continual User Representation without Forgetting" and called CONURE [4]. CONURE[4] can be trained upon different tasks and achieve the desired UR and personalized recommendation (Figure 2.3).

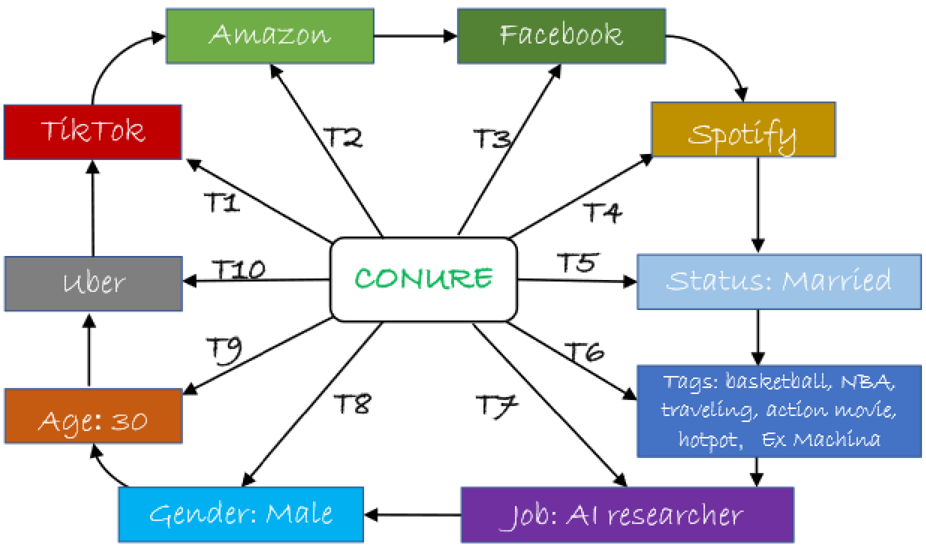

Figure 2.3: UM with CONURE - CONURE's Paper [4]

2.4 Randomness in Deep Learning

As it’s been reported in other Deep Learning techniques, the factor of Randomness [38] always plays a significant role for the engineers. Some widely known examples of such techniques include a) Random Initialization of Weights [39, 40], b) Data Augmentation [41, 42, 43], c) Data Shuffling [44, 45] and d) Dropout [5, 46, 47]. On Figure 2.4, there is presented an example of a Neural Network before and after the application of the Dropout method. The tool of randomness is well recognized for its great results, over different sets of problems. Researchers tend to apply dropout-like methods on their Deep Neural Networks so that to overcome the problem of over-fitting and achieve to create more efficient models.

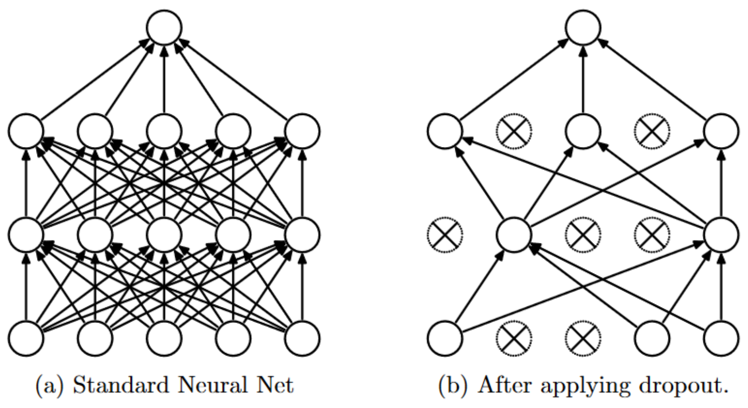

Figure 2.4: Dropout Example - Dropout’s Paper [5]

Chapter 3

State-of-the-art

As stated on Section 2.3, CONURE[4] is a newly created UR model. The authors of this paper, introduce a new non-dynamic architecture paradigm, which aims at overcoming the major problem of catastrophic forgetting (Section 2.2), by using TL (Section 2.1) techniques pretrain and finetune to reinforce the model's weights. Additionally, the creation of a top-K Important Weight Mask with blocked neurons per task, reassures that the model can achieve the desired MTL (Section 2.3), while obtaining a high performance in the whole set of Tasks. The pruning of the non-important weights, which do not affect the model's accuracy on previous tasks, enables new tasks to have a wide capacity of neurons and layers to train for their own purposes. More precisely, the steps of the algorithm are as follows:

- **Pretrain** T_1 in the full capacity of neurons
- **Prune** the unimportant weights of the architecture, depending on their absolute values and free neurons for future tasks
- **Finetune** T_1 to reinforce the Important Weights for the task
- **Continue** the training of future tasks ($\{T_2, T_3, \ldots, T_n\}$) on the freed neurons, without changing the values of the blocked neurons from previous tasks

This innovative technique is the motivation of this study and thus, it's called state-of-the-art model in this research. On the following Figure 3.1, there is a brief representation of the algorithm's steps according to the published paper of CONURE[4].

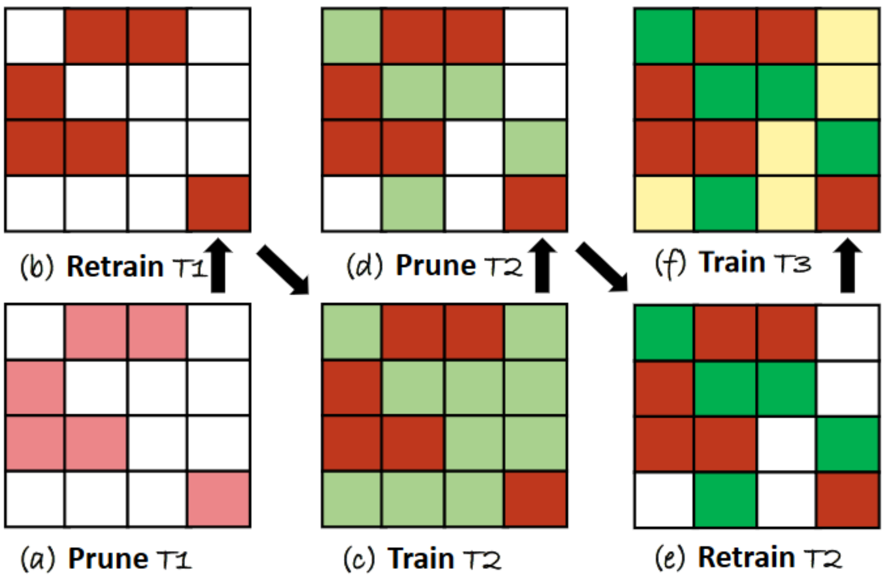

Figure 3.1: CONURE's Overview - CONURE's paper [4]

Chapter 4

Problem Statement

In this section, the main problem, that is attempted to be addresses, is described in mathematical terms as follows:

We define a Tensor named K with dimensions $L * M * N$ which is already trained on n tasks. White blocks are freed neurons, while black blocks are blocked neurons from previous tasks as shown on Figure 4.1. We call T_a and T_b the two tasks which the architecture is trained to in two different orders.

Firstly, we follow CONURE's algorithm[4] for the order O_1: $K => T_a => T_b$. We call M_a (Green) and M_b (Red) the masks produced by the pruning of T_a and T_b with Accuracy A_a and A_b respectively (Figure 4.2).

Secondly, we follow CONURE's algorithm[4] for the order O_2: $K => T_b => T_a$. We call M'_a (Green) and M'_b (Red) the masks produced by the pruning of T_a and T_b with Accuracy A'_a and A'_b respectively (Figure 4.3).

Our problem is defined as the possibility P, where:

1. $m_{b_{l*m*n}} = 0$, element of M_b
2. $m'_{b_{l*m*n}} = 0$, element of M'_b
3. $A_a \approx A'_a$, accuracy of T_a is not changed
4. $A_b < A'_b$, accuracy of T_b is significantly changed

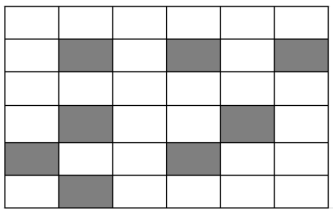
Figure 4.1: Tensor's Overview

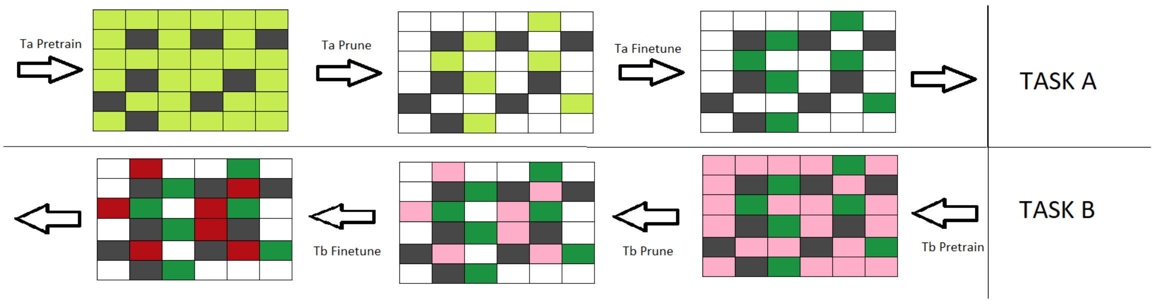

Figure 4.2: O_1: Overview

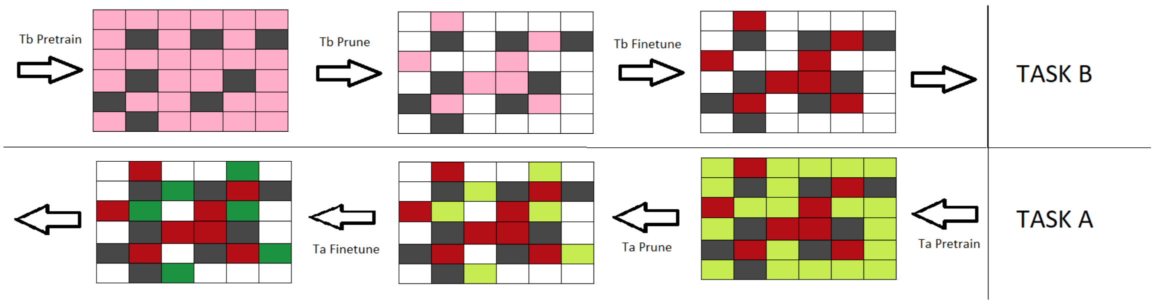

Figure 4.3: O_2: Overview

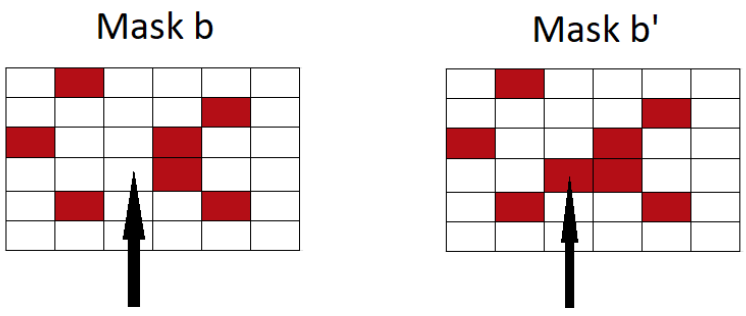

Figure 4.4: Difference between M_b and M_b'

Chapter 5

Proposed Models

In this research, a dropout-like (Section 2.4) technique is proposed, which is called "Random Pruning Technique". The Random Pruning Technique is unblocking neurons of each mask randomly, according to a fixed Random Pruning Percentage. Using this technique, future tasks can be trained on those unblocked neurons, while previous tasks obtain a high performance as well.

The proposed models are CONURE^{+1}, CONURE^{+2} and CONURE^{+3}, whose aim are to overcome the problem that has been explained on Chapter 4. Depending on the tensor's weights, three different sets of weights are used (Figure 5.1). We set the **Highly Important Weights** (Dark Green), **Medium Important Weights** (Light Green) and **Non Important Weights** (White). The Medium Important Weights is created with the help of a new hyper-parameter called Extension (E). The integer value of this hyper-parameter is used to define the limit between the Medium Important Weight Mask and the Non Important Weight Mask. Later, the Random Pruning technique is applied on the mask of each task on three different cases, which are presented in the next sections (5.1, 5.2 and 5.3). The final mask for each task is consisted of the remaining Highly and Medium Important Weights (Dark/Light Green).

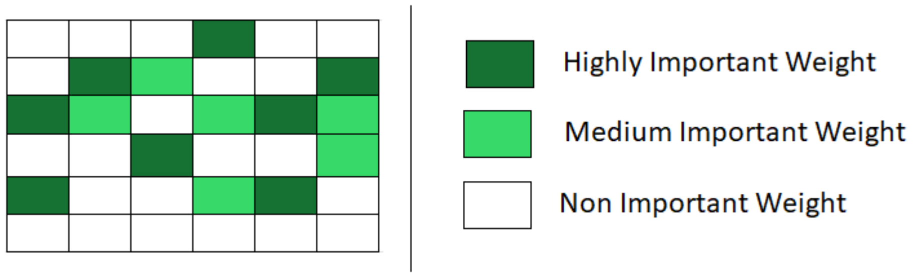

Figure 5.1: Sets of Weights of the tensor

5.1 CONURE^{+1}

The proposed model CONURE^{+1}, is the original idea of this research. This specific model is based on two steps (Figure 5.2). Firstly, we set the Medium Important Weights into Highly Important Weights and then we apply the Random Pruning Technique to the combined Mask. The creation of a wider mask for the Random Pruning Technique to be applied on, according to the first step, makes sure that the number of neurons of the pruning mask can be corresponding to the state-of-the-art CONURE[4].

CONURE^{+1} is a typical example of a dropout-like application. The factor of randomness is applied on the wider combined mask. Although CONURE^{+1} looks promising, it is possible for the mask to unblock some really important weights. This is a potential drawback, since the current's task accuracy could be significantly reduced.

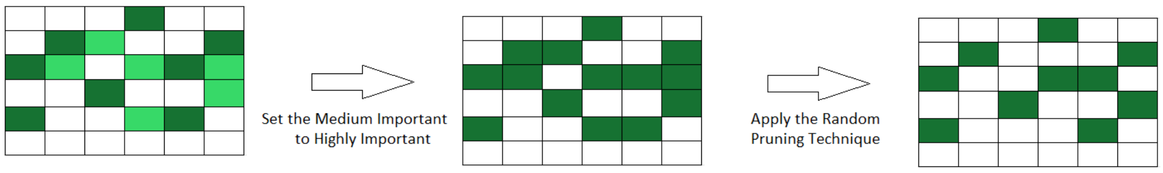
Set the Medium Important to Highly Important
Apply the Random Pruning Technique

Figure 5.2: CONURE^{+1}: Overview

5.2 CONURE^{+2}

In order to overcome the loss of really important weights for current tasks, which is highlighted in Section 5.1, we propose a new model CONURE^{+2}.

According to the stated problem, this model saves the Highly Important Weights, while the Random Pruning Technique is applied explicitly on the Medium Important Weights (Figure 5.3).

Apart from solving the problem stated in Section 5.1, it's been observed that using CONURE^{+2}, results in a constant increase of the number of blocked neurons over the new training sessions. Since later tasks are not expected to have the necessary capacity of neurons to be trained on, the probability of a satisfying performance is remarkably reduced. As a conclusion, the model is not capable of being efficiently trained on the required number of tasks.

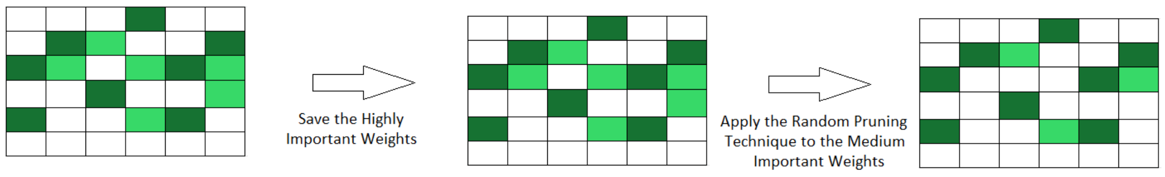
Save the Highly Important Weights
Apply the Random Pruning Technique to the Medium Important Weights

Figure 5.3: CONURE^{+2}: Overview

5.3 CONURE^{+3}

Due to the two problems stated in Section 5.1 and 5.2, there is a need of a third model to be implemented, which is called CONURE^{+3}.

As shown on Figure 5.4 this specific approach is an improvement of CONURE^{+2}'s methodology. Firstly, a new Higher Important Weight Mask is produced, using a smaller set of the highest weights of the model to be saved. This step reassures that the most important weights are masked for the specific task, thus the accuracy of the model is not decreased after the pruning. While, the Random Pruning Technique is applied on the rest High and all the Medium Important Weights, so as to free up some blocked neurons for future tasks and retain the capacity of blocked neurons in relatively low levels.

The given approach looks the most promising among the three above, since it can solve the specific problems stated on the previous sections.

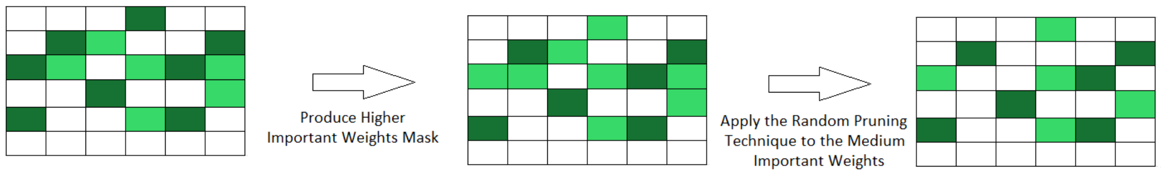
Produce Higher Important Weights Mask
Apply the Random Pruning Technique to the Medium Important Weights

Figure 5.4: CONURE^{+3}: Overview

Chapter 6

Description of the dataset

In this section, the characteristics of the dataset, which is used for the experiments, is explained. The data is collected by the Platform and Content Group of Tencent and has been used by the authors of CONURE[4] and PETERREC[48].

6.1 Task Formulation

We define **T** $= \{T_1, T_2, \ldots, T_n\}$ the set of the tasks, trained on the proposed models. We set T_1 as the watching sequence of the users(basic representation), to assure that all the item IDs are used upon the first training of the model. The watching sequence is generated by the last 100 items viewed by each user in chronological order. In case they are less than 100, a zero padding has been used to fill the 100 items sequence. The set of the rest tasks (i,e $\{T_2, T_3, \ldots, T_n\}$) is a targeted group of tasks with input x the watching sequence of the user and target y either a) interactions (clicks and thumbs-up items) or b) profile information (i,e gender, age, life-status, etc). After the training of **T**, the architecture is expected to give efficient recommendations for all the tasks in **T**.

6.2 Description of Tasks

In this research, a set of four tasks is used to test the efficiency of our proposed algorithm (i,e **T** $= \{T_1, T_2, T_3, T_4\}$). The characteristics of the **T** are explained:

- T_1: basic representation task. T_1 is a 100 items watching sequence of each user in chronological order, in case of less than 100 items by a user, zero padding is applied to complete the 100 items sequence. The input is defined by the first 99 items, while the target is set as the last item of the sequence.

- T_2: interaction task. The input is defined by the 100 items watching sequence of the user, while the target is set as the clicks of the user.
- T_3: interaction task. The input is defined by the 100 items watching sequence of the user, while the target is set as the thumbs-up of the user.
- T_4: profile information task. The input is defined by the 100 items watching sequence of the user, while the target is set as the gender information of the user (i,e $\{0, 1\}$).

The data distribution of the watching sequence per user of **T** are presented on Figures 6.1, 6.2, 6.3 and 6.4. Meanwhile, the size comparison of **T** is presented in a bar plot on Figure 6.5.

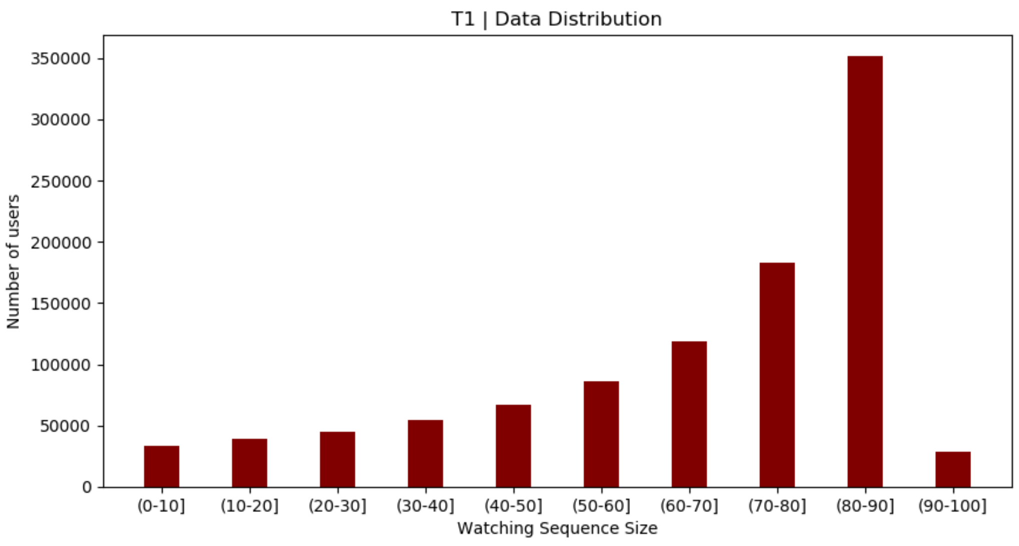

Figure 6.1: Data Distribution of T_1

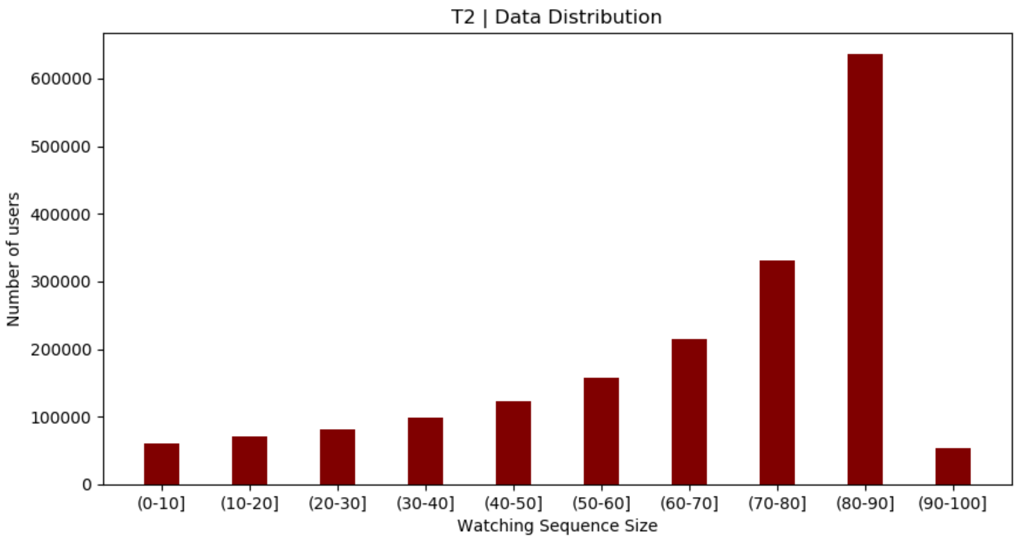

Figure 6.2: Data Distribution of T_2

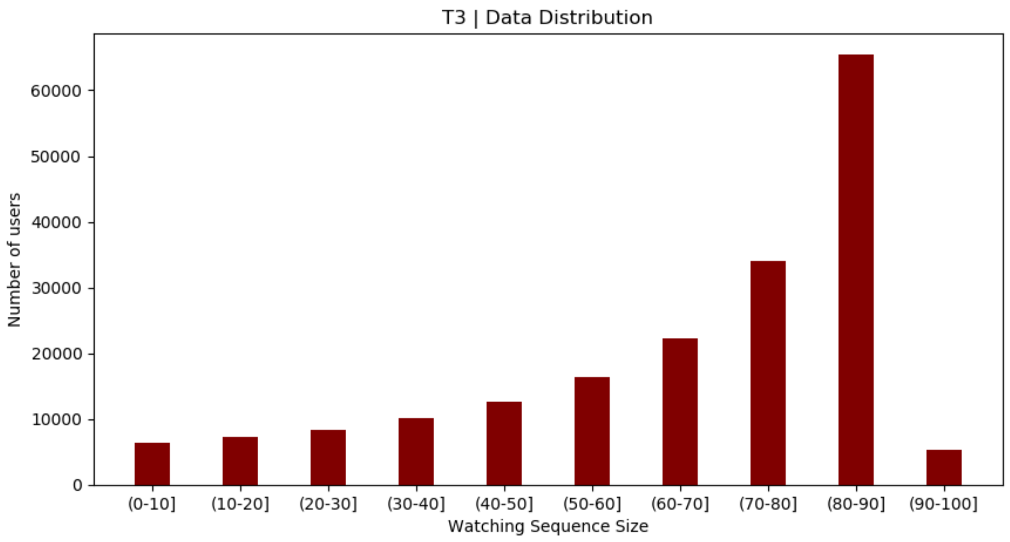

Figure 6.3: Data Distribution of T_3

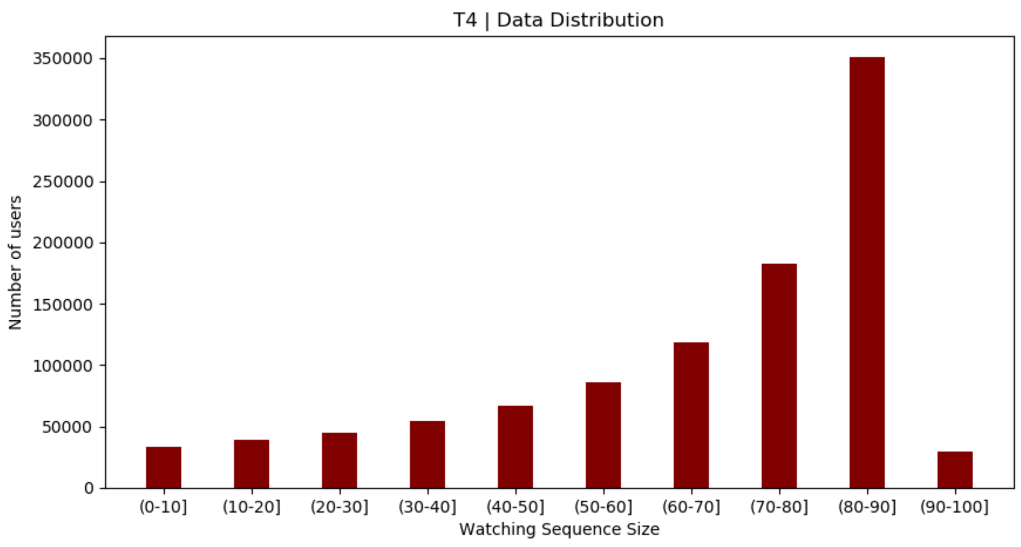

Figure 6.4: Data Distribution of T_4

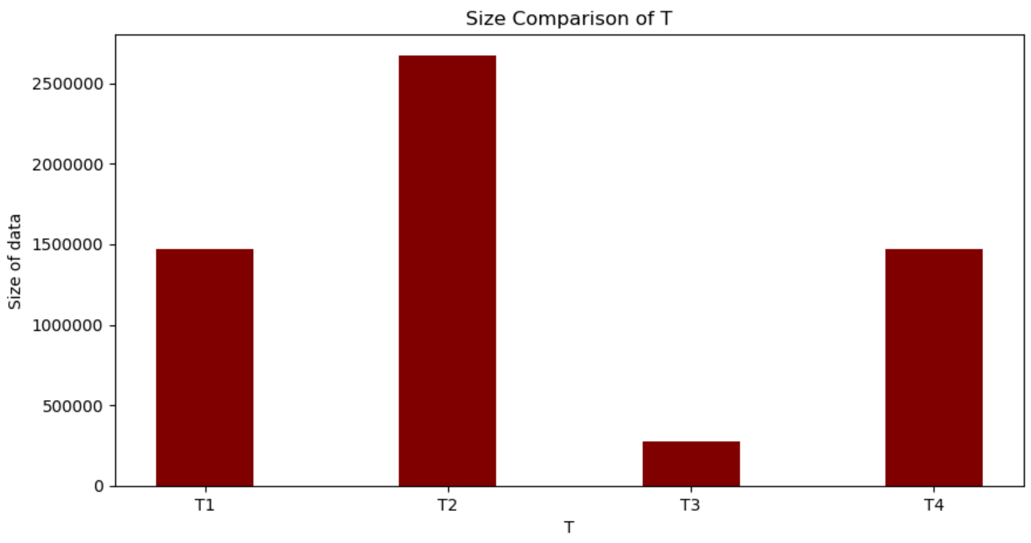

Figure 6.5: Size Comparison of **T**

Chapter 7

Experimental Setup

In this chapter, the experimental setup (i,e Evaluation Protocol and Compared Algorithms) is discussed in detail.

7.1 Evaluation Protocol

In this section, the evaluation protocol of the proposed methods CONURE^{+1}, CONURE^{+2} and CONURE^{+3} is highlighted.

Firstly, the dataset is splited randomly into 80% train set and 20% test set. The train set is used for the needs of the training of each task, while the test set is used to test the models' evaluation metrics.

For the basic representation and the interaction (clicks/thumbs-up) tasks, the popular top-k metric Mean Reciprocal Rank (MRR@k) is used. Mean Reciprocal Rank is a statistic procedure to evaluate the recommendations according to their ranks (i,e the position of the item in the recommendation sequence). While, for the profile information tasks, the metric of accuracy is used. Accuracy is a metric used mostly in classification problems on Machine Learning, since being a Boolean metric.

Furthermore, for each case of the hyper-parameter grid, we conduct five different attempts of the training with random initialization[39, 40]. The metrics' results are collected for each case and are applied to average and standard deviation methods for the needs of plotting. In our experiments the standard deviation is not plotted because of its relatively low value in comparison to the average of the metrics of the five attempts of each hyper-parameters set.

The experiments conducted on a local PC. The PC is a single server machine. The CPU's brand is ”Intel Xeon Bronze 3106” working on 1.70GHz, while the GPU is a ”Geforce RTX 2080 Ti”.

7.2 Compared Methods

In this section, the compared methods CONURE[4] and ADER[3] to the proposed methods are presented. The proposed methods are applied into two different evaluations. The first one is a multi-tasking evaluation using CONURE[4], while the second one is a single-taksing evaluation using ADER[3].

1. **CONURE**[4] is called the state-of-the-art model of this research, it is well explained on Chapter 3
2. **ADER**[3] (Adaptively Distilled Exemplar Replay) is a different type of Continual Learning algorithm. It is a session-based training of a single task which is using the commonly known distillation and exemplar replay techniques for the next sessions of data. ADER[3] is showing great results in the problem of catastrophic forgetting of the previous learned knowledge.

7.3 Evaluation with CONURE

The first stage of evaluation is achieved with the state-of-the-art algorithm CONURE[4]. The hyper-parameters of the default code of the paper are used for the needs of CONURE, CONURE$^{+}1$, CONURE$^{+}2$ and CONURE$^{+}3$, so that there is a base performance to compare with the effect of our proposed drop-out like technique.

According to the proposed models of the Chapter 5, two hyper-parameters are used on the proposed models. The Extension (E) is used to set the Top-(k+E) Weight, which defines the limit for the Medium Important Weight Mask on the proposed models, while the Random Pruning Percentage (RPP) defines the percentage of random pruning applied on each mask of the proposed models. The grids of the two hyper-parameters are set to be $E = [1, 2, 3]$ and $RPP = [0.1, 0.15, 0.25, 0.3, 0.5, 0.6, 0.75]$.

For this evaluation, two results are presented. The HITS represent the values of the grid which are relatively higher compared to the state-of-the-art CONURE for each task separately. Using this evaluation, the performance of the proposed methods can be measured in the total of the grid and thus, general statements can be created for the efficiency of the method.

On the other hand, the second evaluation is a sum-up of the relative differences percentages for each task on each case. This summed percentage can highlight the improvement or deterioration of the proposed method for each case individually. This evaluation can be a measure for conducting the hyper-parameter tuning for each method and thus compare the performance of their best cases.

7.4 Evaluation with ADER

For the needs of the second stage of evaluation, the single-tasking session-based algorithm ADER[3] is used. The importance of this evaluation stage is to show that the proposed methods can obtain relatively good performance on each task individually, in comparison to a widely known single-tasking algorithm of recommendation tasks.

Firstly, a hyper-parameter tuning method is applied to ADER[3], in order to generate the best results for each task. Except for the datasets of each task, three hyper-parameters of the algorithm are used with the following grids $num\_blocks = [1, 2, 3]$, $num\_heads = [1, 2, 3]$ and $hidden\_units = [100, 150, 200]$. On Table 7.1, the results of the hyper-parameter tuning for each task are presented on the right column.

Table 7.1: Hyper-Parameter Tuning for ADER[3]

$Task$	num_blocks	num_heads	$hidden\_units$	$Performance$
T_1	3	3	100	0.21413
T_2	1	1	150	0.21415
T_3	1	1	200	0.21412
T_4	2	3	150	0.37879

Finally, the best performance of each task individually for the proposed methods is compared to the performance of ADER[3]. The best values of the performance tables of the proposed models are extracted and placed on a new table and are individually compared to the ones presented on Table 7.1.

Chapter 8

Experimental Performance

The performance of the experiments of the proposed models CONURE^{+1}, CONURE^{+2} and CONURE^{+3} are explained and presented in the following sections. The results are compared with CONURE[4] and ADER[3] respectively.

8.1 Comparison with CONURE

In this section, the proposed models CONURE^{+1}, CONURE^{+2} and CONURE^{+3} are compared with the state-of-the-art model CONURE[4]. On Table 8.1, the Case Definition is presented on a quadratic table.

Table 8.1: Case Definition of the proposed models

		Random Pruning Percentage						
		0.1	0.15	0.25	0.3	0.5	0.6	0.75
Extension	1	Case$_{01}$	Case$_{02}$	Case$_{03}$	Case$_{04}$	Case$_{05}$	Case$_{06}$	Case$_{07}$
	2	Case$_{08}$	Case$_{09}$	Case$_{10}$	Case$_{11}$	Case$_{12}$	Case$_{13}$	Case$_{14}$
	3	Case$_{15}$	Case$_{16}$	Case$_{17}$	Case$_{18}$	Case$_{19}$	Case$_{20}$	Case$_{21}$

The hyper-parameters set of each case is consisted of i) Extension = [1, 2, 3] and ii) Random Pruning Percentage = [0.1, 0.15, 0.25, 0.3, 0.5, 0.6, 0.75]. The Extension (E) is used to set the Top-(k+E) Weight, which defines the limit for the Medium Important Weight Mask on our proposed models. While the Random Pruning Percentage defines the percentage of random pruning applied on each mask of the proposed models. The presented setup is used to evaluate all the proposed models CONURE^{+1}, CONURE^{+2} and CONURE^{+3} respectively. MRR@5 is used for evaluating T1, T2 and T3 on test set, while accuracy is used for evaluating T4 on test set. We define as HITS the values of the proposed models which are greater, compared to CONURE's values for each task (i,e [0.2141, 0.21416, 0.21404, 0.37868]). The representation of the HITS on our tables are the boxed values.

8.1.1 CONURE^{+1} Performance

In this subsection, the CONURE^{+1} is compared to the state-of-the-art-model CONURE. On Table 8.2, the performance of the model for each task and set of hyper-parameters is presented. The boxed values represent the HITS compared to CONURE's [4] performance [0.2141, 0.21416, 0.21404, 0.37868] for each task respectively.

According to the Table 8.2, it is observed that our proposed algorithm improves the performance of the CONURE[4] in the whole set of the trained tasks. More precisely, the HITS count is observed to be [5, 11, 15, 15] for our tasks respectively. Later tasks' performance, such as T_3 and T_4 are improved in 71.42% of the cases and T_2 is improved in 52.38% of the cases. The behaviour of our model in those tasks, can be logically explained. The proposed technique's aim is to unblock and enable further tensors for future training of tasks. Although, it is reported that the model improves the performance of T_1 in 23.8% of the cases. This is an unexpected result, since T_1 is trained on a smaller set of tensors. It shows that the factor of randomness can always play a significant role upon the Machine Learning algorithms.

The performance of CONURE^{+1} is summarized on Table 8.3. The sum of the percentages of the relative difference between CONURE^{+1} and CONURE[4] is displayed. The sum of HITS per case is shown as well on the second line of the table. The table is visualized on the following heat-maps on Figures 8.1 and 8.2.

Taking into consideration the Figures 8.1 and 8.2. The best value is reported in the set [1, 0.1] of the hyper-parameters. There is a total of 0.095% improvement of the model's evaluation metrics and 4/4 HITS. The area around the global maximum shows great results. Since the small values of the hyper-parameter grid of the proposed Random Pruning Technique are securing the save of the important weights, while they enable future tasks to be further trained on extra freed neurons.

Another remarkable observation is the two local maximums on the [2, 0.5] and [2, 0.75] set of the hyper-parameters. A significant 0.033% and 0.033% improvement is observed

Table 8.2: CONURE^{+1} Performance

T$_1$ evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.21413}$	0.21405	0.21402	0.21398	0.21407	0.21409	0.21407
2	0.21406	$\boxed{0.21412}$	0.21406	0.21402	$\boxed{0.21412}$	0.21408	0.21407
3	0.21405	0.21408	0.21405	0.21401	0.21402	$\boxed{0.21412}$	$\boxed{0.21411}$

T$_2$ evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.21414}$	$\boxed{0.21407}$	$\boxed{0.21413}$	0.21406	$\boxed{0.21407}$	0.21404	$\boxed{0.21407}$
2	$\boxed{0.21409}$	0.21406	0.21402	$\boxed{0.21407}$	0.21401	$\boxed{0.21407}$	$\boxed{0.21415}$
3	0.21405	0.21401	$\boxed{0.21411}$	0.21402	$\boxed{0.21408}$	0.21405	0.21405

T$_3$ evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.21410}$	$\boxed{0.21410}$	$\boxed{0.21409}$	0.21403	$\boxed{0.21408}$	$\boxed{0.21405}$	$\boxed{0.21408}$
2	$\boxed{0.21408}$	$\boxed{0.21408}$	$\boxed{0.21406}$	$\boxed{0.21409}$	$\boxed{0.21410}$	$\boxed{0.21405}$	$\boxed{0.21407}$
3	$\boxed{0.21415}$	0.21404	0.21402	0.21404	$\boxed{0.21405}$	0.21402	0.21403

T$_4$ evaluation - Accuracy

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.37874}$	$\boxed{0.37873}$	$\boxed{0.37873}$	0.37868	$\boxed{0.37876}$	$\boxed{0.37874}$	$\boxed{0.37871}$
2	0.37866	$\boxed{0.37869}$	0.37863	$\boxed{0.37873}$	$\boxed{0.37875}$	$\boxed{0.37876}$	0.37866
3	$\boxed{0.37873}$	0.37865	0.37868	$\boxed{0.37877}$	$\boxed{0.37875}$	$\boxed{0.37872}$	$\boxed{0.3787}$

with 3/4 and 2/4 HITS respectively. The area around those block are showing an impressive HIT rate (3/4), with a relative small improvement of the model's evaluation metrics between 0.001% and 0.030%. The stated observation shows that greater values of the proposed technique can show good results especially on future tasks, which have a wider capacity of neurons to be trained on.

Table 8.3: Comparison of CONURE^{+1} with CONURE[4]

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$+0.095\%$ (4/4)	$+0.023\%$ (3/4)	$+0.032\%$ (3/4)	-0.061% (0/4)	$+0.030\%$ (3/4)	$+0.007\%$ (2/4)	$+0.017\%$ (3/4)
2	$+0.009\%$ (2/4)	$+0.031\%$ (3/4)	-0.041% (1/4)	$+0.004\%$ (3/4)	$+0.033\%$ (3/4)	$+0.021\%$ (3/4)	$+0.037\%$ (2/4)
3	$+0.037\%$ (2/4)	-0.041% (0/4)	-0.009% (1/4)	-0.037% (1/4)	-0.005% (3/4)	$+0.006\%$ (2/4)	$+0.001\%$ (2/4)

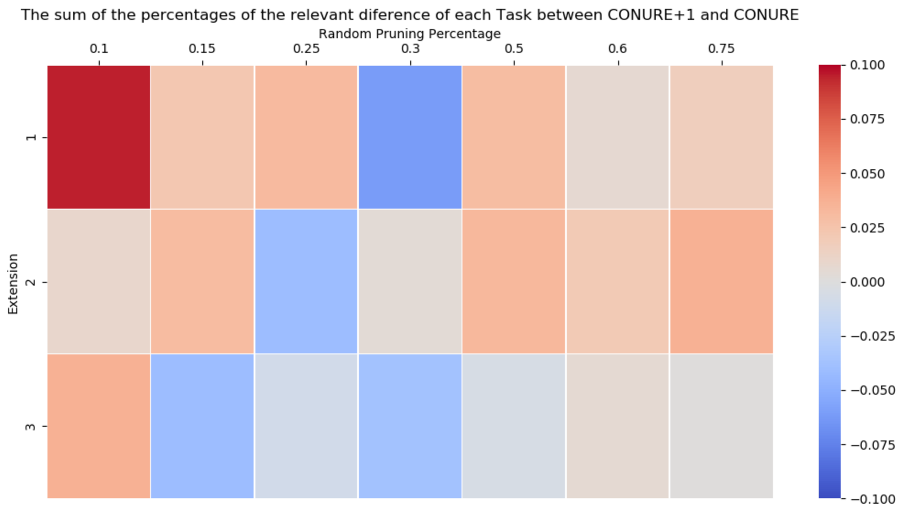

Figure 8.1: Heat-map of the sum of the relevant difference between CONURE^{+1} and CONURE[4]

8.1.2 CONURE^{+2} Performance

In this subsection, the CONURE^{+2} is compared to the state-of-the-art-model CONURE[4]. The main difference between CONURE^{+1} and CONURE^{+2} is summarized by the application of the Random Pruning Technique exclusively on the Medium Important Weights set. This

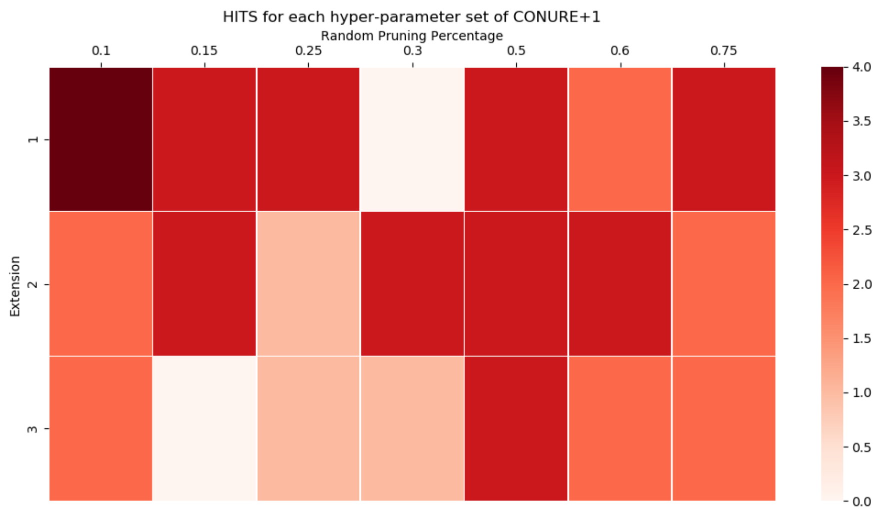

Figure 8.2: Heat-map of the HITS of CONURE^{+1}

approach secures that the full set of Highly Important Weights is saved, while the factor of randomness is applied on the Medium Important Weights set. On Table 8.4, the performance of the model for each task and set of hyper-parameters is presented. The boxed values represent the HITS compared to CONURE's performance [0.2141, 0.21416, 0.21404, 0.37868] for each task respectively.

As a result of the Table 8.4, the proposed model CONURE^{+2} shows great results. The HITS count is reported as [5, 10, 14, 12] for each task respectively. The performance is slightly worse in comparison to CONURE^{+1}'s HITS count (i,e [5, 11, 15, 15]) on future tasks. An important drawback of this method can be reported on the performance of T_4, which is improved on 57.14% of the cases, while we have a slight worse performance as well on T_2 and T_3. The given method's main goal is to give more attention to the first tasks, in contrast to the latest ones. This claim can be easily confirmed by the best values of the evaluation of T_1 and T_2. The reported values of CONURE^{+2} are 0.21418 and 0.21417, in contrast to the best values of CONURE^{+1} which are 0.21413 and 0.21415 on the two tasks respectively. While on the latest tasks, the best performance of T_3 is reported to be significantly lower on 0.21411, and T_4'performance is slightly higher (0.37879) obtaining a significant lower percentage of HITS (57.14%) compared to the first method (71.42%).

The performance of CONURE^{+2} is summarized on Table 8.5. The sum of the percentages of the relative difference between CONURE^{+2} and CONURE[4] is displayed. The sum of HITS per case is shown as well on the second line of the table. The table is visualized on the

Table 8.4: CONURE^{+2} Performance

T_1 evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	0.21404	0.21400	0.21409	0.21408	$\boxed{0.21413}$	0.21403	0.21405
2	0.21407	0.21406	$\boxed{0.21418}$	$\boxed{0.21411}$	0.21401	0.21407	$\boxed{0.21412}$
3	0.21406	0.21401	0.21404	0.21410	0.21405	$\boxed{0.21412}$	0.21405

T_2 evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.21413}$	0.21405	$\boxed{0.21412}$	0.21405	0.21406	0.21405	$\boxed{0.21410}$
2	0.21403	$\boxed{0.21408}$	0.21406	$\boxed{0.21417}$	$\boxed{0.21412}$	$\boxed{0.21408}$	0.21402
3	0.21406	$\boxed{0.21408}$	$\boxed{0.21411}$	0.21404	0.21405	0.21405	$\boxed{0.21410}$

T_3 evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.21411}$	$\boxed{0.21410}$	0.21404	$\boxed{0.21407}$	0.21402	$\boxed{0.21406}$	$\boxed{0.21406}$
2	$\boxed{0.21405}$	0.21404	$\boxed{0.21407}$	$\boxed{0.21411}$	$\boxed{0.21408}$	$\boxed{0.21408}$	$\boxed{0.21408}$
3	0.21403	0.21403	$\boxed{0.21408}$	0.21404	$\boxed{0.21407}$	$\boxed{0.21408}$	0.21401

T4 evaluation - Accuracy

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	0.37861	0.37863	$\boxed{0.37871}$	$\boxed{0.37871}$	$\boxed{0.37872}$	$\boxed{0.37871}$	0.37868
2	$\boxed{0.37877}$	$\boxed{0.37874}$	$\boxed{0.37874}$	0.37868	$\boxed{0.37871}$	$\boxed{0.37879}$	0.37867
3	$\boxed{0.37873}$	0.37867	0.37866	$\boxed{0.37871}$	0.37865	0.37865	$\boxed{0.37873}$

following heat-maps on Figures 8.3 and 8.4.

In conclusion of the Figures 8.3 and 8.4, the best hyper-parameter set is reported on the block [2, 0.3] with an important 0.089% improvement of the state-of-the-art, while a local maximum can be observed in [2, 0.6] showing 0.043% improvement. In the area between [2, 0.25] and [2, 0.6], the hit rate is 75% which is the top value for this specific method in the whole grid, while including the two maximums and the best values of T_1 and T_2.

Comparing the heat-maps of CONURE^{+2} with CONURE^{+1}, the overall performance of the method is slightly worse. On 33.33% of the cases there is negative result, while there is no case with a total of 4 HITS reported.

Table 8.5: Comparison of CONURE^{+2} with CONURE[4]

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$+0.019\%$ (2/4)	-0.037% (1/4)	$+0.031\%$ (2/4)	$+0.008\%$ (2/4)	$+0.015\%$ (2/4)	-0.020% (2/4)	$+0.005\%$ (2/4)
2	$+0.000\%$ (2/4)	$+0.007\%$ (2/4)	$+0.067\%$ (3/4)	$+0.089\%$ (3/4)	$+0.013\%$ (3/4)	$+0.043\%$ (3/4)	$+0.007\%$ (2/4)
3	-0.010% (1/4)	-0.040% (1/4)	$+0.009\%$ (2/4)	-0.001% (1/4)	-0.022% (1/4)	$+0.015\%$ (2/4)	-0.005% (2/4)

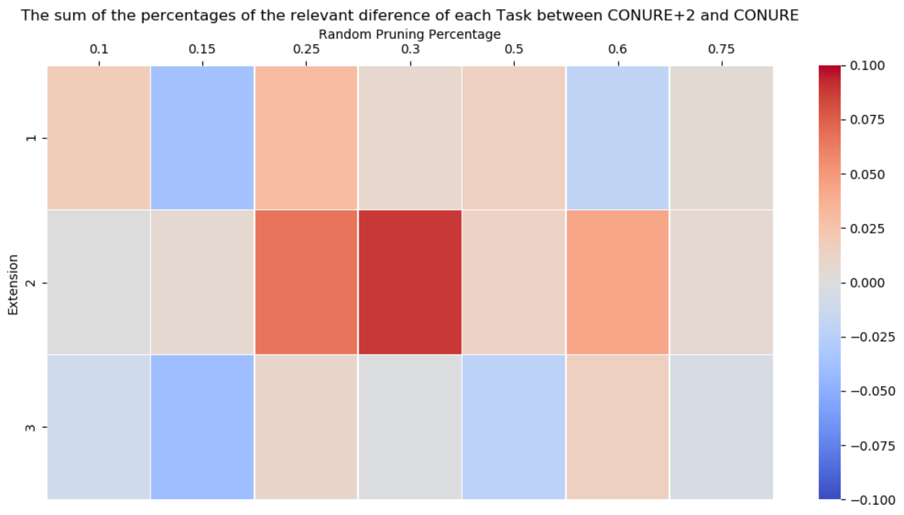

Figure 8.3: Heat-map of the sum of the relevant difference between CONURE^{+2} and CONURE[4]

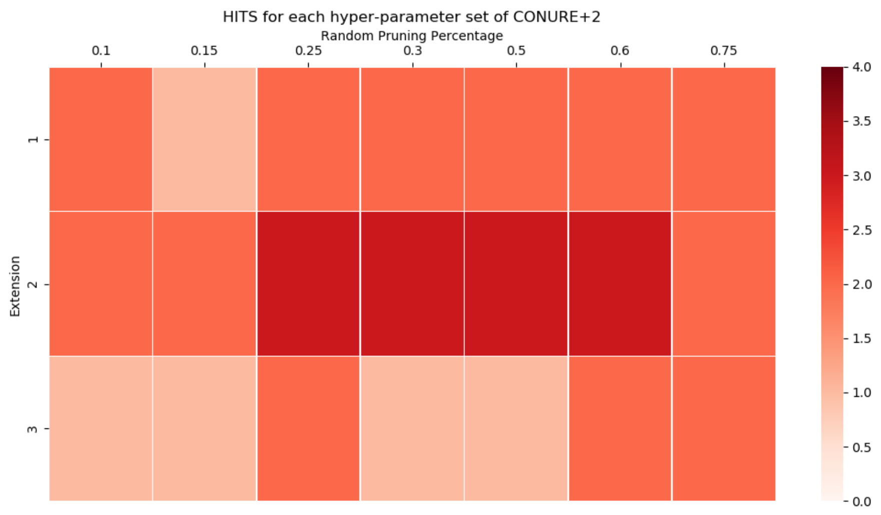

Figure 8.4: Heat-map of the HITS of CONURE^{+2}

8.1.3 CONURE^{+3} Performance

The need of overcoming the problems stated in Subsection 8.1.1 and 8.1.2 results in introducing the new method CONURE^{+3}, which is evaluated compared to the state-of-the-art model CONURE[4] in this subsection. According to this model, a Higher Important Weight Mask is produced and saved, while the proposed technique is applied on the new Medium Important Weight Mask. This approach assures that the number of saved neurons will not exceed the appropriate one, so that future task can have a satisfactory capacity, while first tasks can obtain a good performance because of the saved Important Weights. On Table 8.6, the performance of the model for each task and set of hyper-parameters is presented. The boxed values rep-resent the HITS compared to CONURE's[4] performance [0.2141, 0.21416, 0.21404, 0.37868]for each task respectively.

As, It's been presented on Table 8.6, the HITS count is [6, 13, 19, 15] for each task respectively. The performance of CONURE^{+3} shows the greatest results on HITS rate, in comparison to CONURE^{+1} and CONURE^{+2}. T_1, T_2 and T_3 are improved on 28.57%, 61.9% and 90.47% of the cases respectively, which is the highest rate among the three proposed methods. While T_4 is improved on 71.42% of the cases, meeting the rate of CONURE^{+1}, the highest reported on this task. The highest values for each task are 0.21415, 0.2141, 0.21412 and 0.37877, which shows the equal high performance of this method on the whole set of tasks with the relatively highest HITS rate.

The performance of CONURE^{+3} is summarized on Table 8.7. The sum of the percentages

Table 8.6: CONURE^{+3} Performance

T_1 evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	0.21406	$\boxed{0.21411}$	0.21406	$\boxed{0.21412}$	0.21409	0.2141	0.21402
2	0.21405	0.21406	0.21403	0.21404	0.2141	0.21406	0.21404
3	$\boxed{0.21414}$	$\boxed{0.21411}$	$\boxed{0.21412}$	0.21408	0.21407	$\boxed{0.21415}$	0.21403

T_2 evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	0.21401	$\boxed{0.21412}$	$\boxed{0.21410}$	0.21405	$\boxed{0.21414}$	0.21405	$\boxed{0.21411}$
2	0.21400	0.21402	$\boxed{0.21411}$	$\boxed{0.21407}$	$\boxed{0.21411}$	$\boxed{0.21414}$	$\boxed{0.21410}$
3	0.21404	$\boxed{0.21409}$	$\boxed{0.21410}$	0.21404	$\boxed{0.21409}$	$\boxed{0.21409}$	0.21401

T_3 evaluation - MRR@5

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.21412}$	0.21403	$\boxed{0.21409}$	0.21403	$\boxed{0.21406}$	$\boxed{0.21405}$	$\boxed{0.21411}$
2	$\boxed{0.21406}$	$\boxed{0.21408}$	$\boxed{0.21410}$	$\boxed{0.21406}$	$\boxed{0.21408}$	$\boxed{0.21406}$	$\boxed{0.21409}$
3	$\boxed{0.21406}$	$\boxed{0.21412}$	$\boxed{0.21412}$	$\boxed{0.21408}$	$\boxed{0.21408}$	$\boxed{0.21406}$	$\boxed{0.21405}$

T_4 evaluation - Accuracy

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$\boxed{0.37872}$	0.37866	$\boxed{0.37877}$	0.37867	$\boxed{0.37873}$	$\boxed{0.37876}$	0.37867
2	$\boxed{0.37869}$	0.37866	$\boxed{0.37871}$	$\boxed{0.37871}$	$\boxed{0.37874}$	$\boxed{0.37870}$	$\boxed{0.37872}$
3	$\boxed{0.37873}$	$\boxed{0.37873}$	$\boxed{0.37871}$	$\boxed{0.37870}$	0.37866	$\boxed{0.37873}$	0.37868

of the relative difference between CONURE^{+3} and CONURE[4] is displayed. The sum of HITS per case is shown as well on the second line of the table. The table is visualized on the following heat-maps on Figures 8.5 and 8.6.

Table 8.7: Comparison of CONURE^{+3} with CONURE[4]

	0.1	0.15	0.25	0.3	0.5	0.6	0.75
1	$+0.006\%$ (2/4)	$+0.023\%$ (2/4)	$+0.047\%$ (3/4)	-0.003% (1/4)	$+0.055\%$ (3/4)	$+0.021\%$ (2/4)	$+0.016\%$ (2/4)
2	-0.039% (2/4)	-0.024% (1/4)	$+0.027\%$ (3/4)	-0.006% (3/4)	$+0.058\%$ (3/4)	$+0.033\%$ (3/4)	$+0.025\%$ (3/4)
3	$+0.032\%$ (3/4)	$+0.069\%$ (4/4)	$+0.073\%$ (4/4)	$+0.005\%$ (2/4)	$+0.013\%$ (2/4)	$+0.060\%$ (4/4)	-0.051% (1/4)

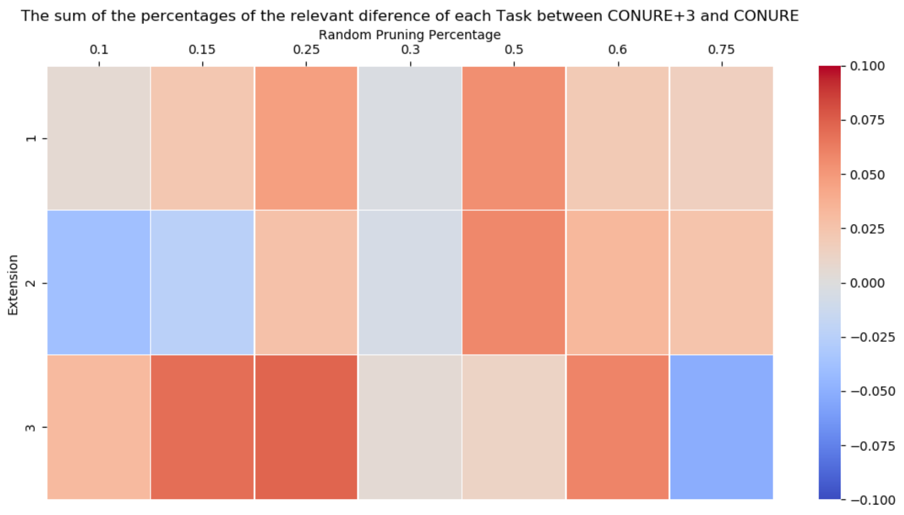

Figure 8.5: Heat-map of the sum of the relevant difference between CONURE^{+3} and CONURE[4]

Observing the Table 8.5 and Figures 8.5 and 8.6, the maximum is pointed out on [3, 0.25] with a 0.073% improvement and 4/4 HITS, while local maximums are [3, 0.6] with 4/4 HITS and 0.06% improvement, [1, 0.25] with 3/4 HITS and 0.047% improvement and [2, 0.5] with 3/4 HITS and 0.058% improvement. The best values of the table are swifted in the third row in comparison to second row of CONURE^{+2}. This is an expected behaviour, since the High Important Weight Mask is reduced by one weight and the Medium Important Weight Mask needs an extra weight to meet the appropriate number of the trainable neurons per task. The local maximums around the heat-map can be a result of the higher HITS rate per task, which

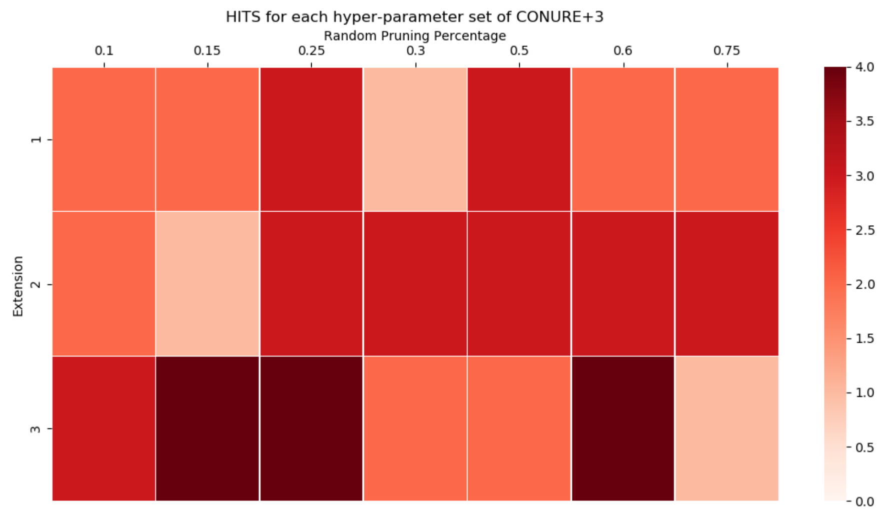

Figure 8.6: Heat-map of the HITS of CONURE^{+3}

is the main goal of this proposed model. Only 23.8% of cases show negative overall results, with 14.28% of the cases with fewer than 2/4 HITS count.

8.1.4 Final Evaluation with CONURE

On the following subsection, we are going to present the final evaluation of the proposed models with CONURE[4]. On Table 8.8, there is presented the overview of HITS for the proposed methods. The rate of hits per task are shown using parenthesis, while the average HITS for each method is presented on the right column. Moreover, on Table 8.9, the best values after the hyper-parameter tuning (E for Extension and RPP for Random Pruning Percentage) of each method are presented. The boxed values represent the HITS, while on the right column the sum of the percentages of the relative difference between each method and CONURE[4] is calculated and presented.

Table 8.8: HITS rate of CONURE^{+1}, CONURE^{+2} and CONURE^{+3}

Method	T_1	T_2	T_3	T_4	Average
CONURE^{+1}	5 (14.28%)	11 (52.38%)	15 (71.42%)	15 (71.42%)	11.50 (54.76%)
CONURE^{+2}	5 (14.28%)	10 (47.61%)	14 (66.66%)	12 (57.14%)	10.25 (48.80%)
CONURE^{+3}	6 (28.57%)	13 (61.90%)	19 (90.47%)	15 (71.42%)	13.25 (63.09%)

Table 8.9: Hyper-parameter Tuning of CONURE^{+1}, CONURE^{+2} and CONURE^{+3}

Method	E	RPP	T_1	T_2	T_3	T_4	Improvement
CONURE^{+1}	1	0.1	$\boxed{0.21413}$	$\boxed{0.21414}$	$\boxed{0.21410}$	$\boxed{0.37874}$	$+0.095\%$
CONURE^{+2}	2	0.3	$\boxed{0.21411}$	$\boxed{0.21417}$	$\boxed{0.21411}$	0.37868	$+0.089\%$
CONURE^{+3}	3	0.25	$\boxed{0.21412}$	$\boxed{0.21410}$	$\boxed{0.21412}$	$\boxed{0.37871}$	$+0.073\%$

Summarizing the Subsections 8.1.1, 8.1.2 and 8.1.3 on Tables 8.8 and 8.9, we present the results of the proposed methods in comparison to the state-of-the-art CONURE. The HITS rate of CONURE^{+3} shows the greatest results and thus, this method improves the performance of CONURE[4] more efficient uniformly among different tasks. The performance of this method compared to CONURE^{+1} and CONURE^{+2} on the HITS rate can be reported with the average HITS per task on the right column of Table 8.8. Showing an average of 13.25 (63.09%) HITS per task, CONURE^{+3} leads the evaluation. While, CONURE^{+2} following with 11.50 (54.76%) HITS per task and CONURE^{+1} reporting the worst results with 10.25 (48.80%) HITS per task.

On the other hand, after conducting the hyper-parameter tuning on the proposed methods and choosing the best values on Table 8.9, it is reported that CONURE^{+1} leads the evaluation with a total of $+0.095\%$ improvement and 4/4 HITS. The following CONURE^{+2} shows an improvement of $+0.089\%$ with 3/4 HITS and finally CONURE^{+3} improves the original method by $+0.073\%$ obtaining a 4/4 HITS count.

8.2 Comparison with ADER

On this section, the proposed models CONURE^{+1}, CONURE^{+2} and CONURE^{+3} are compared to the famous single-tasking session-based algorithm ADER[3]. On the Table 8.10, the performance of the proposed methods and the compared algorithm ADER[3] is presented. On the first row, there are the results of the hyper-parameter tuning of ADER[3], explained on Section7.4. While for the evaluation of the proposed methods, the best values of the hyper-parameter tuning on Tables 8.2, 8.4 and 8.6 for each task separately are used. As ADER[3] is basically a single-tasking high performing algorithm for recommendation tasks, we are interested to check when the proposed models reach its performance. In other words, for this specific evaluation, we set HITS the values of the proposed methods, which are greater or equal to the ones of ADER[3]. The signs $^{=}$ and $^{\vartriangle}$ represent the equal and higher values respectively.

Table 8.10: Evaluating the proposed methods for each task separately with ADER

Method	T_1	T_2	T_3	T_4
ADER	0.21413	0.21415	0.21412	0.37879
CONURE^{+1}	$\boxed{0.21413^{=}}$	$\boxed{0.21415^{=}}$	$\boxed{0.21415^{\vartriangle}}$	0.37877
CONURE^{+2}	$\boxed{0.21418^{\vartriangle}}$	$\boxed{0.21417^{\vartriangle}}$	0.21411	$\boxed{0.37879^{=}}$
CONURE^{+3}	$\boxed{0.21415^{\vartriangle}}$	0.21414	$\boxed{0.21412^{=}}$	0.37877

According to the Table 8.10, it is reported that all the compared methods CONURE^{+1}, CONURE^{+2} and CONURE^{+3} obtain great performance on each task separately as well. More precisely, CONURE^{+2} performs greater in comparison to the rest methods, with HITS on 3/4 tasks. CONURE^{+2}, on T_1 and T_2 shows great improvements with 0.21418 and 0.21417, which are the highest values of those specific tasks. While obtaining equal performance on T_4, the highest among the compared methods, reports a slightly worse performance on T_3, compared to ADER[3]. Moreover, CONURE^{+1} reports good results as well with 3/4 HITS and on T_3 the greater value of 0.21415 in comparison to the other methods. Lastly, CONURE^{+3} performs worse compared to the other methods with only 2/4 HITS and one higher value on T_1 compared to ADER[3].

In comparison to ADER[3], the compared methods show great results in general. Apart from meeting the performance of ADER[3] in all the tasks, in some specific cases the proposed methods obtain higher performance even from this well known single-tasking algorithm. On Figure 8.7, the performance of the proposed methods and ADER[3] are displayed in bar plots for each task separately.

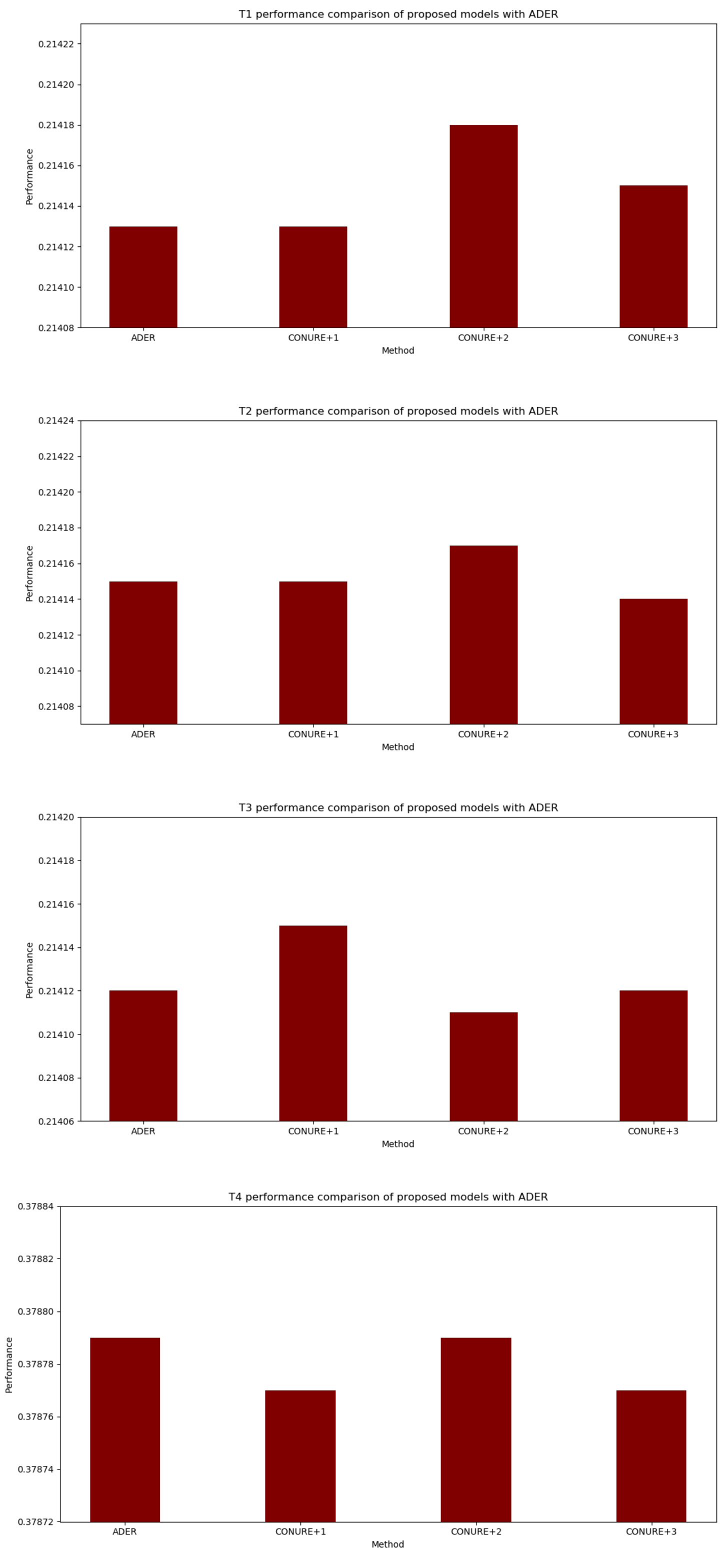

Figure 8.7: CONURE^{+1}, CONURE^{+2} and CONURE^{+3} comparison to ADER[3] for each task

Chapter 9

Conclusions and Future Work

9.1 Conclusions

In this thesis, we addressed the problem of multi-tasking personalized recommendations. According to Chapter 2, many approaches have been tested to solve the stated problem with good results. Transfer Learning (Section 2.1), Continual Learning (Section 2.2) and User Modeling (Section 2.3) are some of those famous approaches. Although, many steps need to be taken in order to improve the performance of those algorithms.

The factor of randomness (Section 2.4) always plays a significant role to the performance of Deep Learning algorithms and thus we conducted experiments to improve one of the most efficient Multi-Tasking Learning algorithms called CONURE[4] which is further explained on Chapter 3. The main problem stated on Chapter 4 focuses on the possibility of the algorithm to block really important neurons for future tasks on the early stages of the training and as a result reduce the possible performance of those future trainings.

In order to overcome this problem, we are proposing the Random Pruning Technique, a dropout-like technique, to the mask of each task so that to unblock neurons for future tasks. The proposed models (Chapter 5) can be summarized by the application of the Random Pruning Technique on three different approaches on the algorithm.

While evaluating the proposed models (Chapter 7) with both CONURE[4] and ADER[3], a famous single-tasking session-based algorithm, great results have been observed (Chapter 8). More precisely, all the proposed models obtain greater results in comparison to the two compared methods as shown on Tables 8.8, 8.9 and 8.10. According to the three different sets of evaluations of the proposed models, it is observed that on each evaluation there is a different higher performing model and thus we back-up the claim that the factor of randomness is really important upon creating efficient Deep Learning methods.

As a result of the HIT rate of the proposed models in comparison to CONURE [4], we set as HIT the set of hyper-parameters that outperforms the CONURE [4] model, we observe that CONURE^{+3} performs greater among the other methods with an average of 63.09% HITS for every task in the whole hyper-parameter grid with $[6, 13, 19, 15]$ hits on each task respectively as shown on Table 8.8. On the other hand, CONURE^{+1} after applying the hyper-parameter tuning, performs greater on the set of $[1, 0.1]$ hyper-parameters, with 4/4 HITS and a significant 0.095% improvement in comparison to CONURE's [4] performance (Table 8.9). Last but not least, while evaluating the proposed methods with the famous session-based single-tasking ADER [3], it is reported on Table 8.10 that CONURE^{+2} performs greater with 3/4 HITS and the best results on 50% of the tasks.

9.2 Future Work

In the process of completing this thesis, many different experiments have been conducted. The methods, which are tested, show great results in the challenging problem of multi-tasking personalized recommendation. Due to the huge computational cost of the training on real data, each experiment needs a big amount of time to be completed and in this section we present the possible future work.

According to the CONURE's [4] algorithm and the proposed methods, we observe that the usage of top-k values for the weight selection of the masks can generally cause problems to the learned tasks. Lower weights can be saved on a specific task thus being part of the top-k values and as a result block those neurons from being trained on future tasks. While, significant weights could be unblocked on other tasks because they don't appear on top-k values of the neurons and as a result lose important information in the process. In order to overcome this problems, we propose to implement a threshold [49, 50, 51] hyper-parameter for the need of choosing the important weights of each task's mask. This method can reassure for the information saved on each task according to the weight of each neuron, since the higher valued weights obtain great amount of information for the trained task.

Another significant improvement could be focused on the algorithm itself. For the needs of the Multi-Tasking Learning methods CONURE [4], CONURE^{+1}, CONURE^{+2} and CONURE^{+3}, the base idea is the creating of a top-k mask for each task. Although this technique shows great results on the given problem, an attention mechanism [52, 53, 54] could be used for the importance of each neuron of the tensors to a specific task. Instead of blocking neurons for each task, some neurons could be used by different tasks equally, in case they can contribute to the training and testing efficiently by the Deep Learning architecture. According to the key values which are generated during each task's training, we can specify the impact of each neuron to every specific trained task. Such attention mechanisms are widely used by the researchers on Deep Learning, because of their great results in different sets of problems.

Bibliography

[1] Alan Eckhardt. Various aspects of user preference learning and recommender systems. In *DATESO*, pages 56–67, 2009.

[2] L. Torrey and J. Shavlik. Transfer learning. *Handbook of Research on Machine Learning Applications*, 01 2009.

[3] Fei Mi, Xiaoyu Lin, and Boi Faltings. *ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-Based Recommendation*, page 408–413. Association for Computing Machinery, New York, NY, USA, 2020.

[4] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, Joemon Jose, Beibei Kong, and Yudong Li. *One Person, One Model, One World: Learning Continual User Representation without Forgetting*, page 696–705. Association for Computing Machinery, New York, NY, USA, 2021.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[6] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

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

[8] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[9] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, 13(5):393–408, 1999.

[10] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105, 2011.

[11] Robin Van Meteren and Maarten Van Someren. Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, volume 30, pages 47–56, 2000.

[12] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019.

[13] Mohammadreza Iman, Khaled Rasheed, and Hamid R. Arabnia. A review of deep transfer learning and recent advancements. *CoRR*, abs/2201.09679, 2022.

[14] Kafeng Wang, Xitong Gao, Yiren Zhao, Xingjian Li, Dejing Dou, and Cheng-Zhong Xu. Pay attention to features, transfer learn faster cnns. In *International Conference on Learning Representations*, 2020.

[15] Xiangxi Mo, Ruizhe Cheng, and Tianyi Fang. Pay attention to convolution filters: Towards fast and accurate fine-grained transfer learning. *CoRR*, abs/1906.04950, 2019.

[16] Rui Ye and Qun Dai. Implementing transfer learning across different datasets for time series forecasting. *Pattern Recognition*, 109:107617, 2021.

[17] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Transfer learning for time series classification. *CoRR*, abs/1811.01533, 2018.

[18] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[20] Xing Fang. Making recommendations using transfer learning. *Neural Computing and Applications*, 33, 08 2021.

[21] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016.

[22] Massimo Caccia, Pau Rodríguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam H. Laradji, Irina Rish, Alexandre Lacoste, David Vázquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *CoRR*, abs/2003.05856, 2020.

[23] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[24] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[25] Abhiram Iyer, Karan Grewal, Akash Velu, Lucas Oliveira Souza, Jeremy Forest, and Subutai Ahmad. Avoiding catastrophe: Active dendrites enable multi-task learning in dynamic environments. *CoRR*, abs/2201.00042, 2022.

[26] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[27] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486, 2019.

[28] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[29] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[30] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 06–11 Aug 2017.

[31] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, 2019.

[32] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[33] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '18, page 596–605, New York, NY, USA, 2018. Association for Computing Machinery.

[34] Jiachun Wang, Fajie Yuan, Jian Chen, Qingyao Wu, Min Yang, Yang Sun, and Guoxiao Zhang. *StackRec: Efficient Training of Very Deep Sequential Recommender Models by Iterative Stacking*, page 357–366. Association for Computing Machinery, New York, NY, USA, 2021.

[35] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 582–590, New York, NY, USA, 2019. Association for Computing Machinery.

[36] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management*, CIKM '20, page 1893–1902, New York, NY, USA, 2020. Association for Computing Machinery.

[37] Nicolas Masse, Gregory Grant, and David Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115, 02 2018.

[38] Donglin Zhuang, Xingyao Zhang, Shuaiwen Leon Song, and Sara Hooker. Randomness in neural network training: Characterizing the impact of tooling. *CoRR*, abs/2106.11872, 2021.

[39] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[41] Jan Kukacka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *CoRR*, abs/1710.10686, 2017.

[42] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[43] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):13001–13008, Apr. 2020.

[44] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017.

[45] Qi Meng, Wei Chen, Yue Wang, Zhi-Ming Ma, and Tie-Yan Liu. Convergence analysis of distributed stochastic gradient descent with shuffling. *Neurocomputing*, 337:46–57, 2019.

[46] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. Survey of dropout methods for deep neural networks. *CoRR*, abs/1904.13310, 2019.

[47] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[48] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. Parameter-efficient transfer from sequential behaviors for user profiling and recommendation. *CoRR*, abs/2001.04253, 2020.

[49] Joan S. Weszka and Azriel Rosenfeld. Threshold evaluation techniques. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(8):622–629, 1978.

[50] Xuehui Wu, Xiaobo Lu, and Henry Leung. An adaptive threshold deep learning method for fire and smoke detection. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1954–1959, 2017.

[51] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International joint conference on neural networks (IJCNN)*, pages 1–8. ieee, 2015.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[53] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[54] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.