AL4REC
ADVERSARIAL LEARNING FOR RECOMMENDATION

Vito Walter Anelli, Yashar Deldjoo, Tommaso Di Noia and Felice Antonio Merra

*28/03/2021 – Tutorial at **ECIR 2021**, Virtual Event, Italy*
43rd EUROPEAN CONFERENCE ON INFORMATION RETRIEVAL

ABOUT US

Vito Walter ANELLI

@walteranelli

Yashar DELDJOO

@yashardel

Tommaso DI NOIA

@TommasoDiNoia

Felice Antonio MERRA

@merrafelice

SisInf Lab,
Polytechnic **U**niversity of **B**ari, Italy

Politecnico di Bari

RELATED PUBLICATIONS FOR THIS TUTORIALS BY @SISINFLAB

- **Survey**
  - *A Survey on Adversarial Recommender Systems: From Attack/Defense Strategies to Generative Adversarial Networks,* **ACM Computing Surveys**, March 2021 Article No.: 35 **https://doi.org/10.1145/3439729**

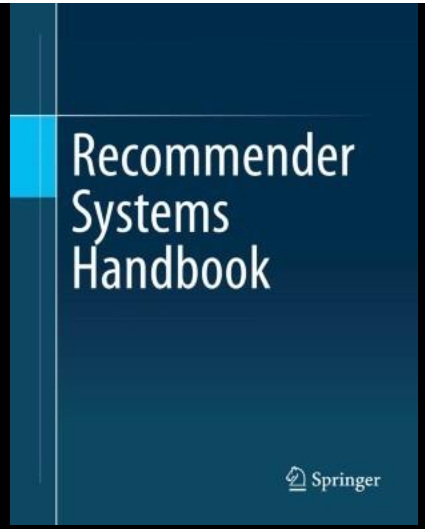

RESEARCH-ARTICLE

A Survey on Adversarial Recommender Systems: From Attack/Defense Strategies to Generative Adversarial Networks

Authors: Yashar Deldjoo, Tommaso Di Noia, Felice Antonio Merra Authors Info & Affiliations

Publication: ACM Computing Surveys • March 2021 • Article No.: 35 • https://doi.org/10.1145/3439729

- **Book Chapter**
  - *Adversarial Recommender Systems: Attack, Defense, and Advances*
  - Sep 2020, accepted to the 3rd Edition of Recommender System Handbook



Politecnico di Bari

UP TO DATE REPOSITORY

https://github.com/sisinflab/adversarial-recommender-systems-survey

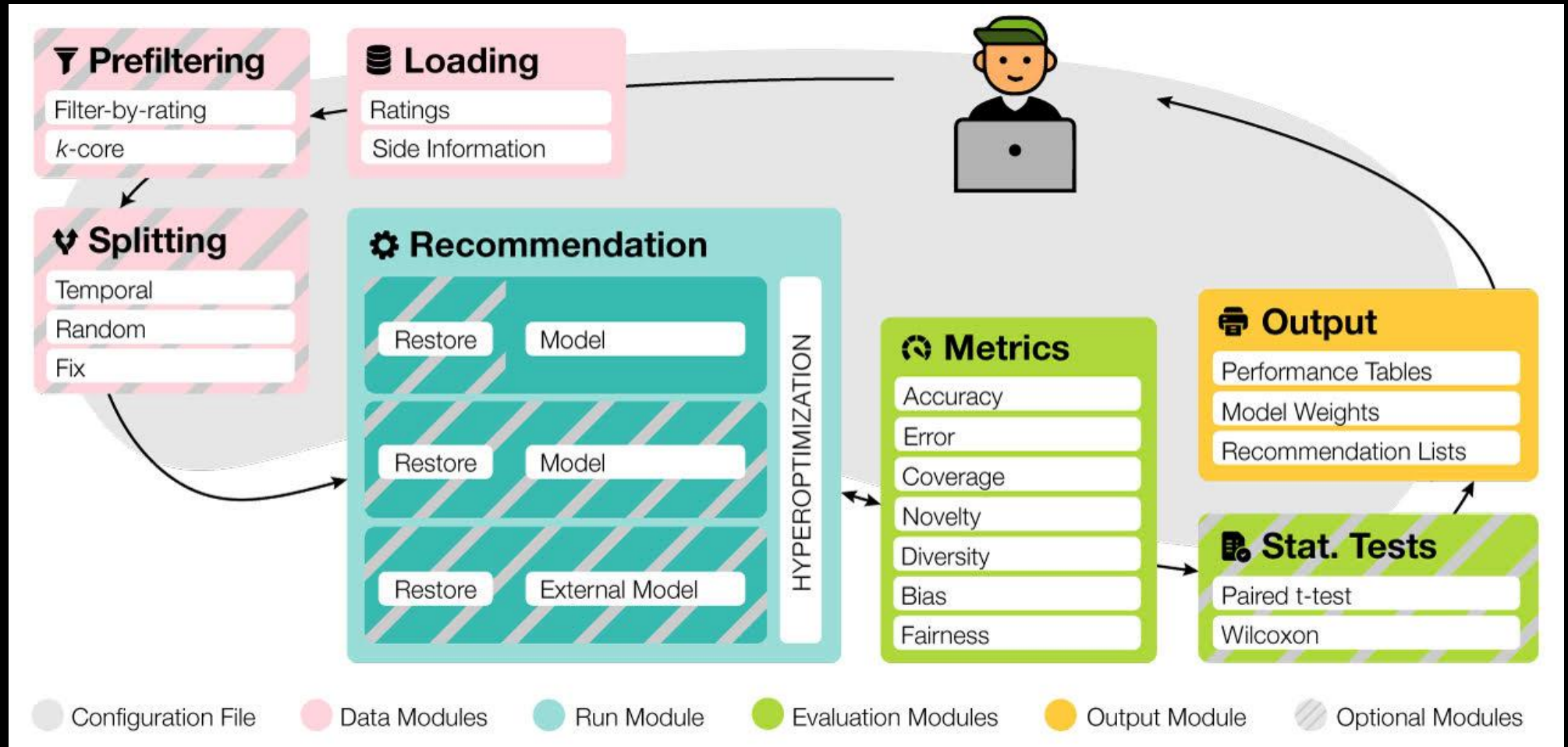
Papers

ADVERSARIAL MACHINE LEARNING FOR SECURITY OF RS

Year	Title	Type	Target Model	Venue	Link	Code
2021	**Adversarial Item Promotion: Vulnerabilities at the Core of Top-N Recommenders that Use Images to Address Cold Start**	Attack	DVBPR/VBPR/AMR	WWW	Link	Code
2021	**A Black-Box Attack Model for Visually-Aware Recommender Systems**	Attack	VBPR/DeepStyle	WSDM	Link	Code
2020	**Assessing Perceptual and Recommendation Mutation of Adversarially-Poisoned Visual Recommenders**	Attack	VBPR/AMR	NeurIPS-WS	Link	Code

ELLIOT

https://github.com/sisinflab/elliot



Politecnico di Bari

ELLIOT

https://elliot.readthedocs.io/en/latest/guide/recommenders.html

Recommendation Models

- Adversarial Learning
- Algebric
- Autoencoders
- Content-Based
- Generative Adversarial Networks (GANs)
- Graph-based
- Knowledge-aware
- Latent Factor Models
- Artificial Neural Networks
- Neighborhood-based Models
- Unpersonalized Recommenders
- Visual Models

PLAN FOR TODAY

14:15 – 15:00

Part I: Introduction

- Goal, Statistics,
- Foundations

15:00 – 16:15

Part II: AML for RS

- Foundations of AML for security of ML systems
- Adversarial Learning for attack-defense strategies in Recommender Systems
- GAN-based Recommendation Framework (GAN-RF)

16:15 – 16:45

COFFEE BREAK

16:45 – 18:00

Part III – Hands-on

- Hands-on Session
- Adversarial and GAN models in ELLIOT

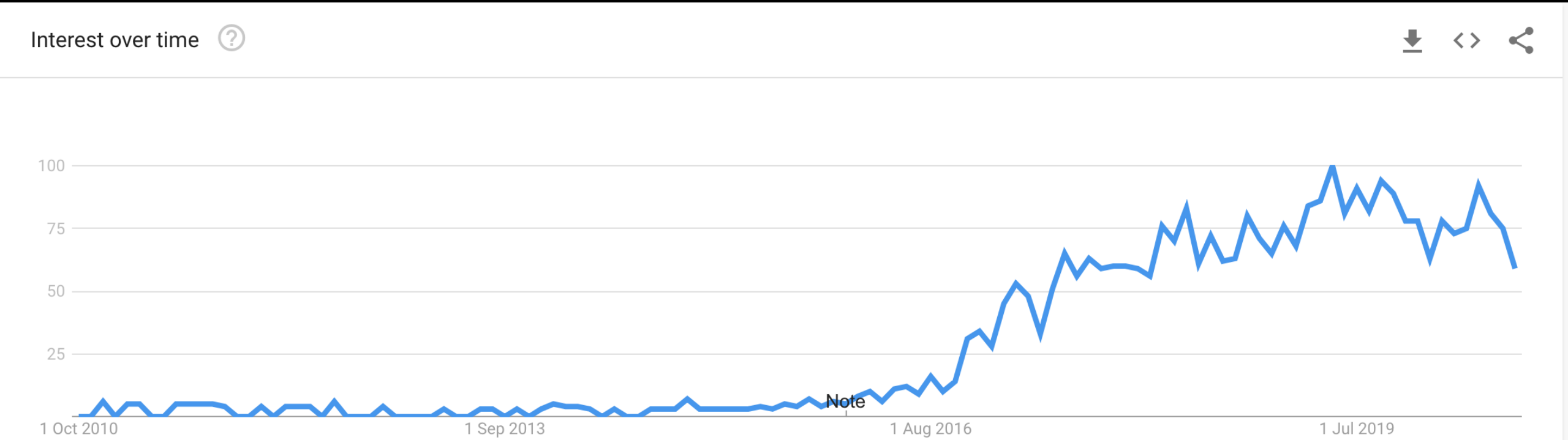
18:00 – 18:15

Q&A

Politecnico di Bari

PART 1
INTRODUCTION

ADVERSARIAL LEARNING: SEARCH TERM FREQUENCY OVER TIME



Politecnico di Bari

[SOURCE GOOGLE TRENDS]

ADVERSARIAL LEARNING: SUGGESTED RELATED TOPICS

Related topics	Rising
1 Learning - Topic	Breakout
2 Generative adversarial networks - Topic	Breakout
3 Adversarial machine learning - Field of study	Breakout
4 Generative model - Topic	Breakout
5 Deep learning - Topic	Breakout

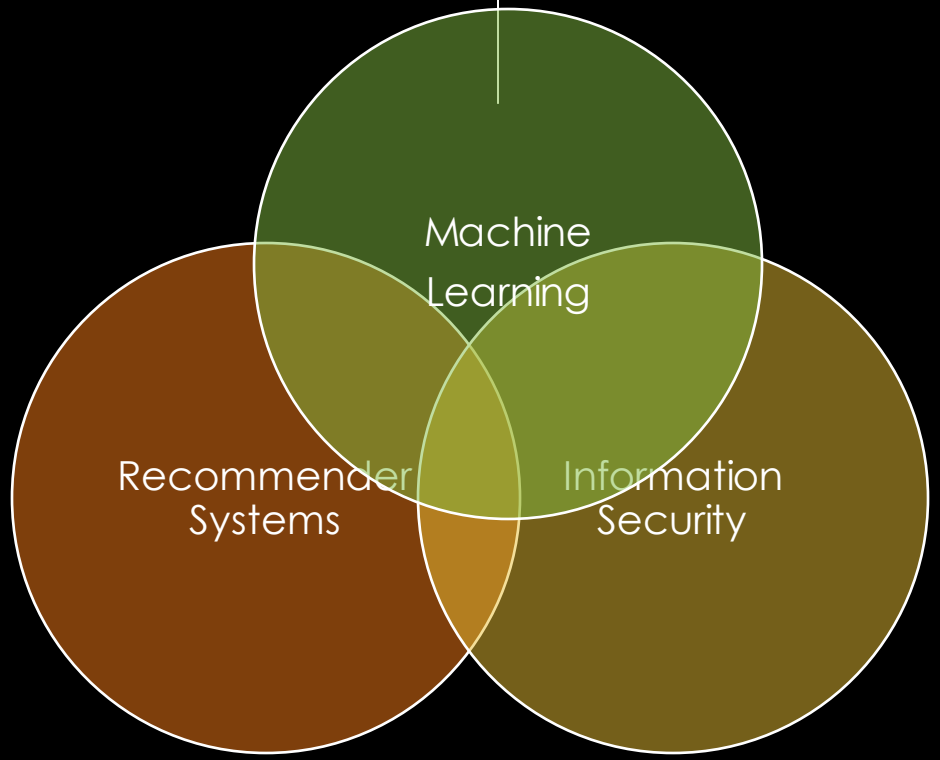
Related queries	Rising
1 deep learning	Breakout
2 adversarial networks	Breakout
3 adversarial machine learning	Breakout
4 generative adversarial networks	Breakout
5 gan	Breakout

Politecnico di Bari

[SOURCE GOOGLE TRENDS]

THE FOCUS OF OUR TUTORIAL

Adversarial Learning for RS security



Concept of Adversarial Learning USED in

Generative Adversarial Networks (GANs)

Politecnico di Bari

ADVERSARIAL LEARNING FOR RS

- More than **80** papers in the last **3** years
- Top Conferences/Journals involved

- WSDM
- SIGIR
- RecSys
- KDD
- WWW
- NIPS
- KDD
- IJCAI
- ICML
- CIKM
- AAAI
- TKDE
- ACL
- MM
- TOIS
- TIST
- CSUR
- ECIR
- …

RECSYS + ADVERSARIAL LEARNING

Evaluation Goals

- Accuracy
- Coverage
- Confidence and Trust
- Novelty
- Serendipity
- Diversity
- Security
- Privacy
- Fairness
- Scalability

Recomendation Models

- Collaborative Filtering
  - Model-based
  - Memory-based
  - Graph-based
  - Deep
- Content-based Filtering
  - Metadata
  - Multimedia (audio and visual)
  - Knowledge-base
- Context-aware
  - Social
  - Temporal
- Hybrid

RECSYS + ADVERSARIAL LEARNING

Evaluation Goals

- Accuracy
- Coverage
- Confidence and Trust
- Novelty
- Serendipity
- Diversity
- Security
- Privacy
- Fairness
- Scalability

Increases the ability of learning in adversarial setting

Recomendation Models

- Collaborative Filtering
  - Model-based
  - Memory-based
  - Graph-based
  - Deep
- Content-based Filtering
  - Metadata
  - Multimedia (audio and visual)
  - Knowledge-base
- Context-aware
  - Social
  - Temporal
- Hybrid

TUTORIAL GOALS

- Bridge the gap between advances made in the field of RecSys and Security
- Understand key concepts in adversarial machine learning
- Adversarial machine learning + Generative Adversarial Network (GANs)
- Present AML-RecSys hands-on for security and generative scenarios.
- Present future directions of AML for RecSys

A flavor of Recommender Systems

SOME DEFINITIONS

- In its most common formulation, **the recommendation problem is reduced to the problem of estimating ratings** for the items that have not been seen by a user.

[G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A survey of the State-of-the-Art and Possible Extension. TKDE, 2005.]

- Recommender Systems (**RS**s) are software tools and techniques providing **suggestions** for items to be of use to a user.

[F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors. Recommender Systems Handbook. Springer, 2015.]

THE PROBLEM

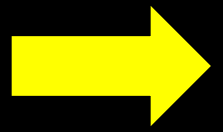
- Estimate a utility function to automatically predict how much a user may like an item which is unknown to them.

Input

Set of users $U = \{u_1, \dots, u_M\}$

Set of items $X = \{x_1, \dots, x_N\}$

Utility function $f: U \times X \rightarrow \mathbb{R}$



Output

$$\forall\, u \in U, x'_u = \arg max_{x \in X}\, f(u, x)$$

RATING PREDICTION

$$L_{PMF}(U,V) = \sum_{i=1}^{M} \sum_{j \in L_i} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda(||U||_F^2 + ||V||_F^2)$$

rating prediction loss

regularization

RANKING

Maximize the probability p that *k* is ranked higher than *j* for all pairs in $\mathcal{D}_s$

$$\prod_{(i,k,j)\in\mathcal{D}_s} p(k >_i j|\Theta), \forall k \in L_{i,+},\ j \in L_{i,-}$$

$\mathcal{D}_s$ contains all$(+,-)$pairs for each user

$$p(k >_i j|\Theta) = \sigma(\hat{x}_{ikj}(\Theta))$$

$$\hat{x}_{ikj}(U,V) = \hat{x}_{ik} - \hat{x}_{ij} = \mathbf{u}_i^T\mathbf{v}_k - \mathbf{u}_i^T\mathbf{v}_j$$

$$L_{BPR}(U,V) = \sum_{(i,k,j)\ \in\mathcal{D}_s} \boxed{log(1+exp(-\mathbf{u}_i^T(\mathbf{v}_k - \mathbf{v}_j)))} + \lambda(\|U\|_F^2 + \|V\|_F^2)$$

pairwise ranking loss

Politecnico di Bari

Motivation

EVERYTHING STARTED WITH A PANDA

"panda"

+

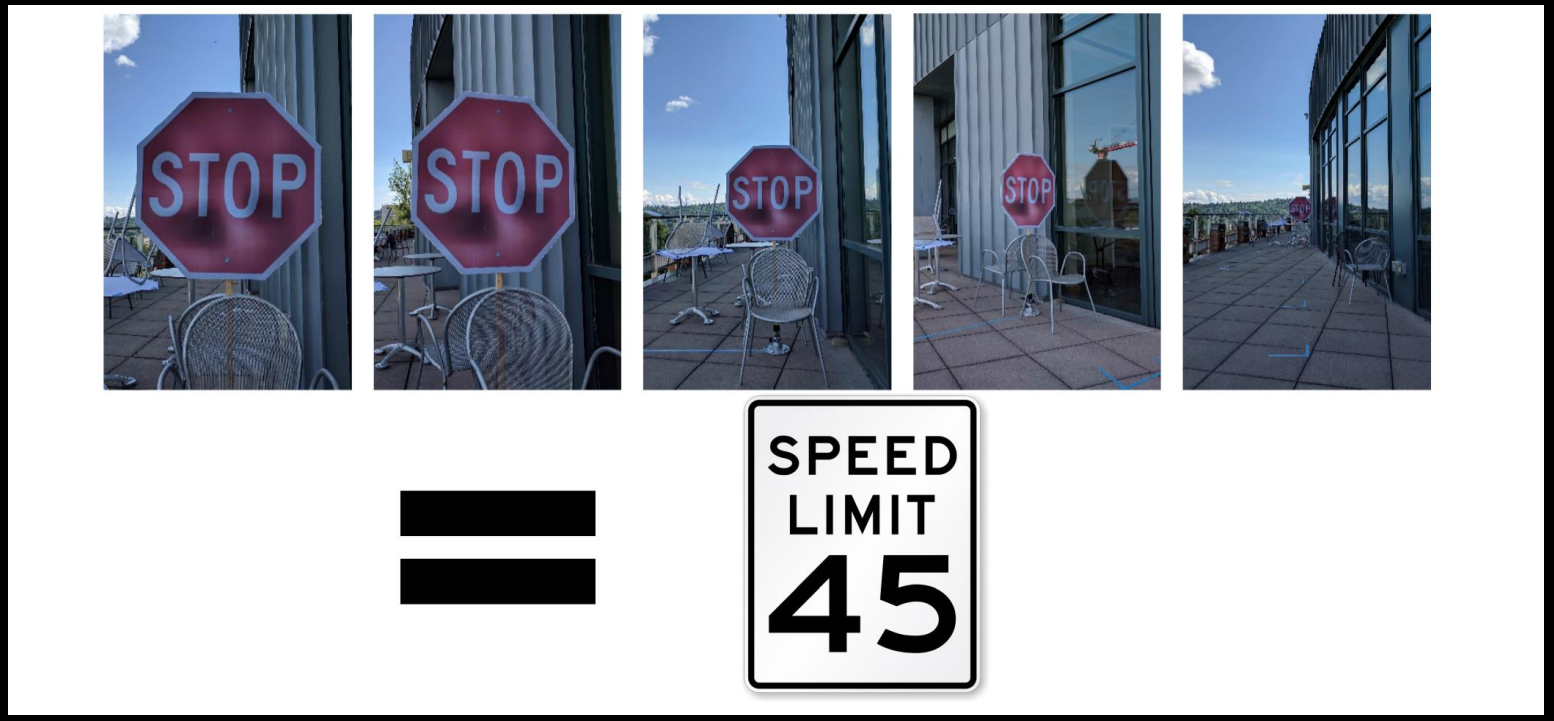
Adversarial Noise

=

"gibbon"

ADVERSARIAL EXAMPLES IN PHYSICAL WORLD

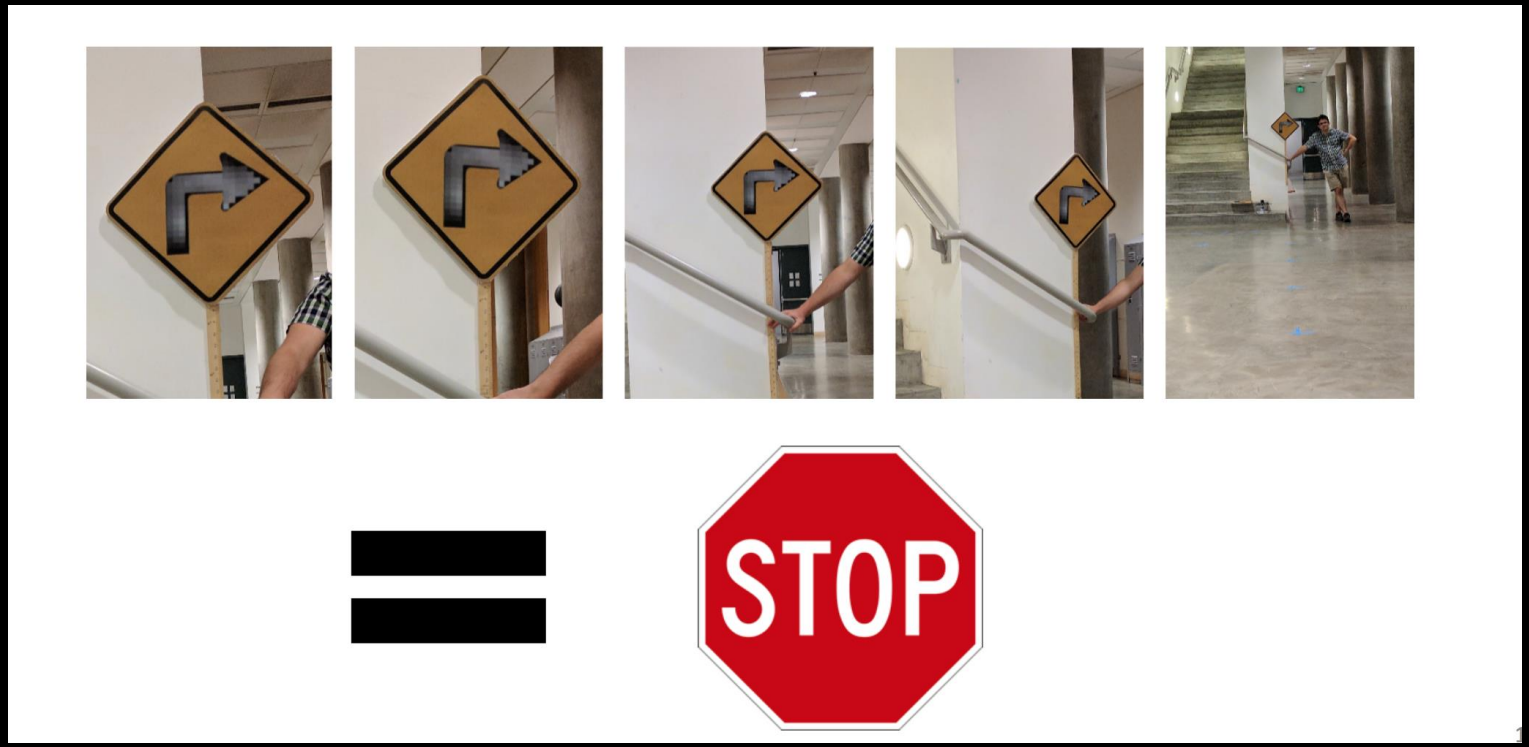
Subtle Perturbations



Source: Song, Dawn. "AI and Security: Lessons, Challenges & Future Directions", UC Berkeley, 2017
Evtimov, Ivan et. al. "Robust Physical-World Attacks on Machine Learning Models." arXiv preprint arXiv:1707.08945 (2017).

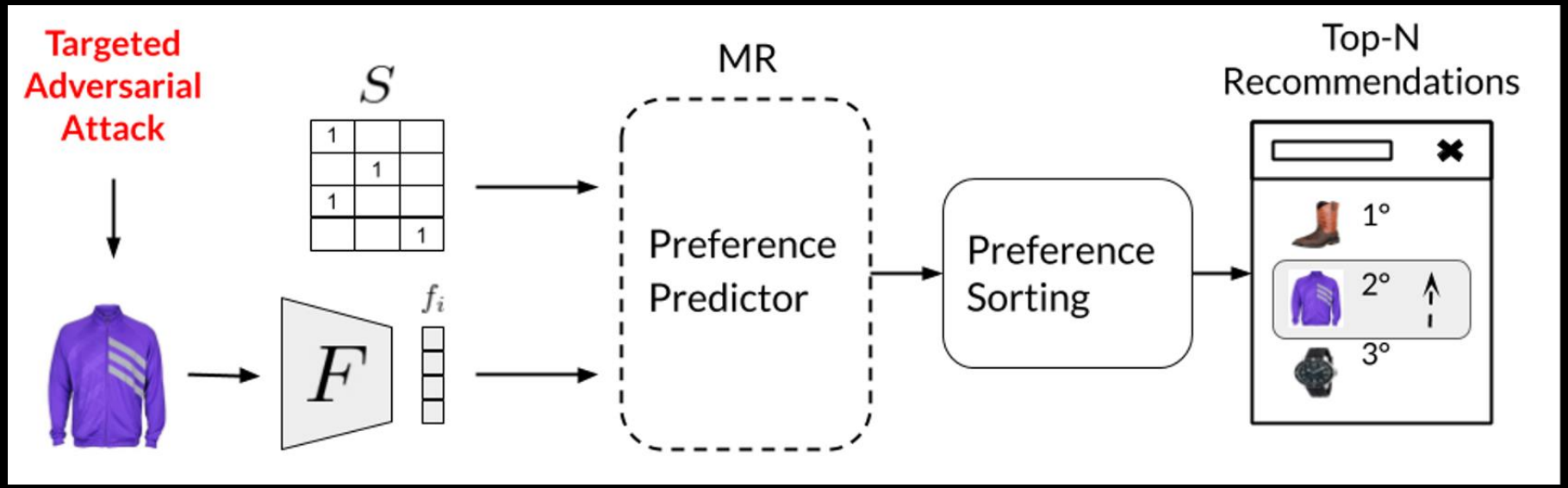
ADVERSARIAL EXAMPLES IN PHYSICAL WORLD

Subtle Perturbations



Source: Song, Dawn. "AI and Security: Lessons, Challenges & Future Directions", UC Berkeley, 2017
Evtimov, Ivan et. al. "Robust Physical-World Attacks on Machine Learning Models." arXiv preprint arXiv:1707.08945 (2017).

ADVERSARIAL EXAMPLES IN RS



Simulation of Targeted Adversarial Attacks against Multimedia Recommender Systems can push low recommended product categories even **3 times more recommended** by **perturbing** product images in a **human-imperceptible way.**

[Di Noia, Tommaso, Daniele Malitesta, and Felice Antonio Merra. "TAaMR: Targeted Adversarial Attack against Multimedia Recommender Systems." the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-DSML'20). 2020.]

Politecnico di Bari


FAILURE

IT'S ALL ABOUT DATA

«Yet, we found that adversarial examples are relatively robust, and are shared by neural networks with varied number of layers, activations or trained on different subsets of the training data.

That is, if we use one neural net to generate a set of adversarial examples, we find that these examples are still statistically hard for another neural network even when it was trained with different hyperparameters or, most surprisingly, when it was trained on a different set of examples»

[C. SZEGEDY ET AL. INTRIGUING PROPERTIES OF NEURAL NETWORKS, ICLR'14]

ACTUALLY NOT REALLY…

«… adversarial examples can be directly attributed to the presence of *non-robust features*: features (derived from patterns in the data distribution) that are highly predictive, yet brittle and (thus) incomprehensible to humans.»

«*Adversarial vulnerability is a direct result of our models' sensitivity to well-generalizing features in the data.*»

«…this perspective establishes adversarial vulnerability as a human-centric phenomenon, since, from the standard supervised learning point of view, non-robust features can be as important as robust ones »

[A. ILYAS ET AL. ADVERSARIAL EXAMPLES ARE NOT BUGS, THEY ARE FEATURES, NIPS'19]

MORE PRACTICALLY

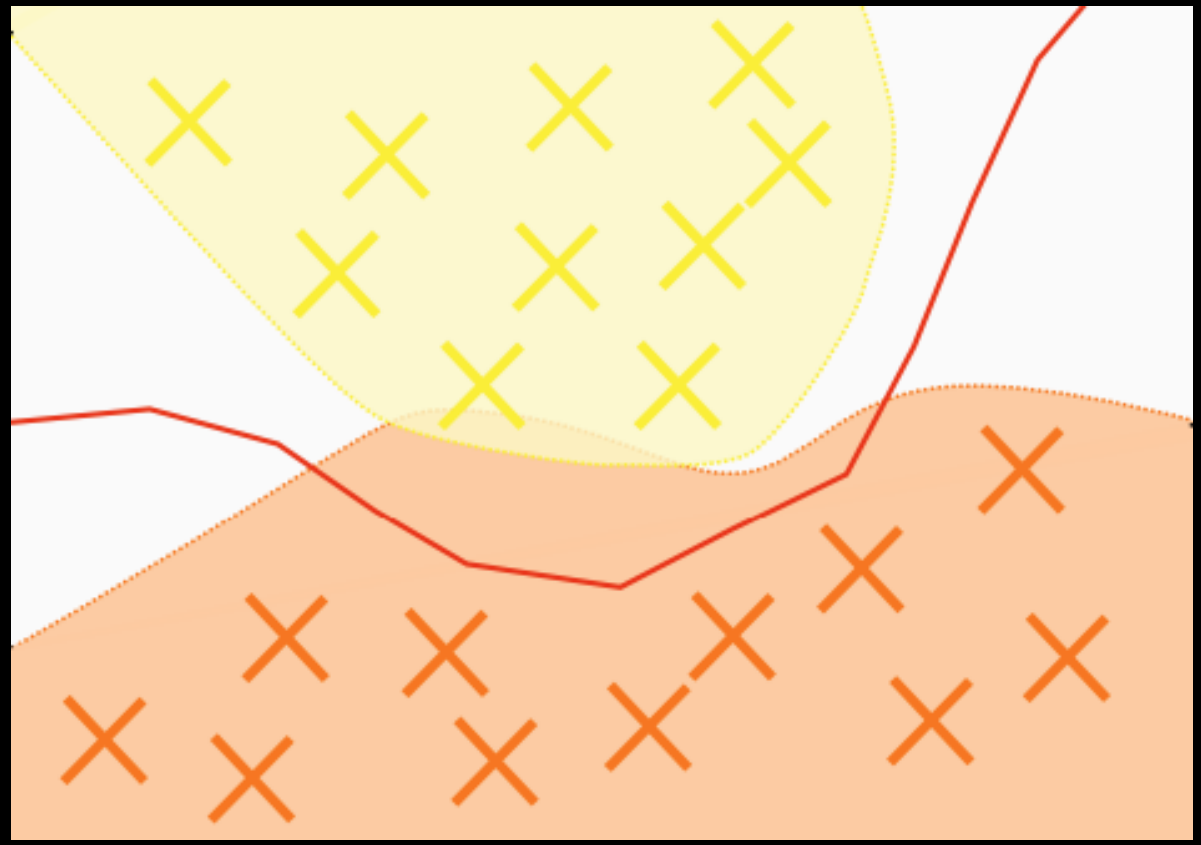
« Specifically, **the inputs x are usually assumed to all be drawn independently from the same probability distribution at both training and test time**.

This means that while test inputs x are new and previously unseen during the training process, they at least have the same statistical properties as the inputs used for training.

Such assumptions have been useful for designing effective machine learning algorithms but implicitly **rule out the possibility that an adversary could alter the distribution at either training time or test time**.»

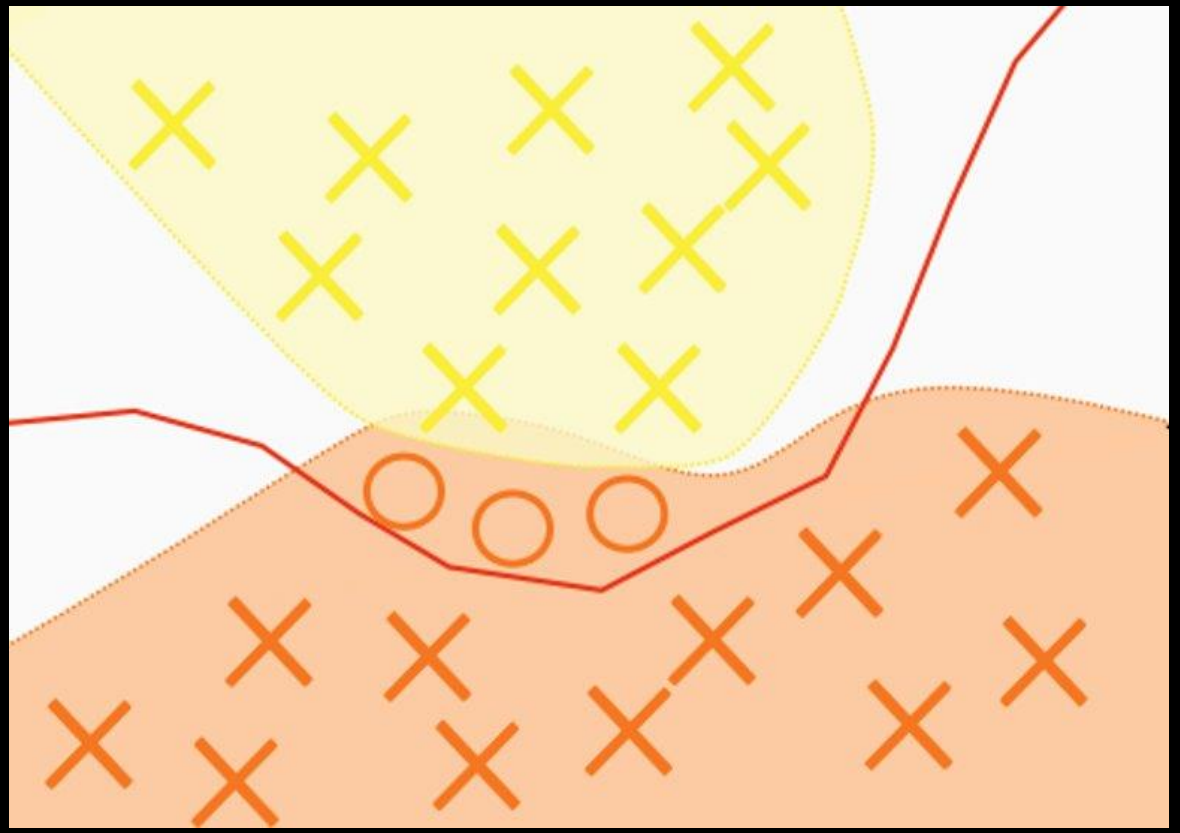
[IAN GOODFELLOW ET AL. MAKING MACHINE LEARNING ROBUST AGAINST ADVERSARIAL INPUTS. COMMUNICATIONS OF THE ACM, JULY 2018]

IDEA: DECISION BOUNDARY OF THE MODEL



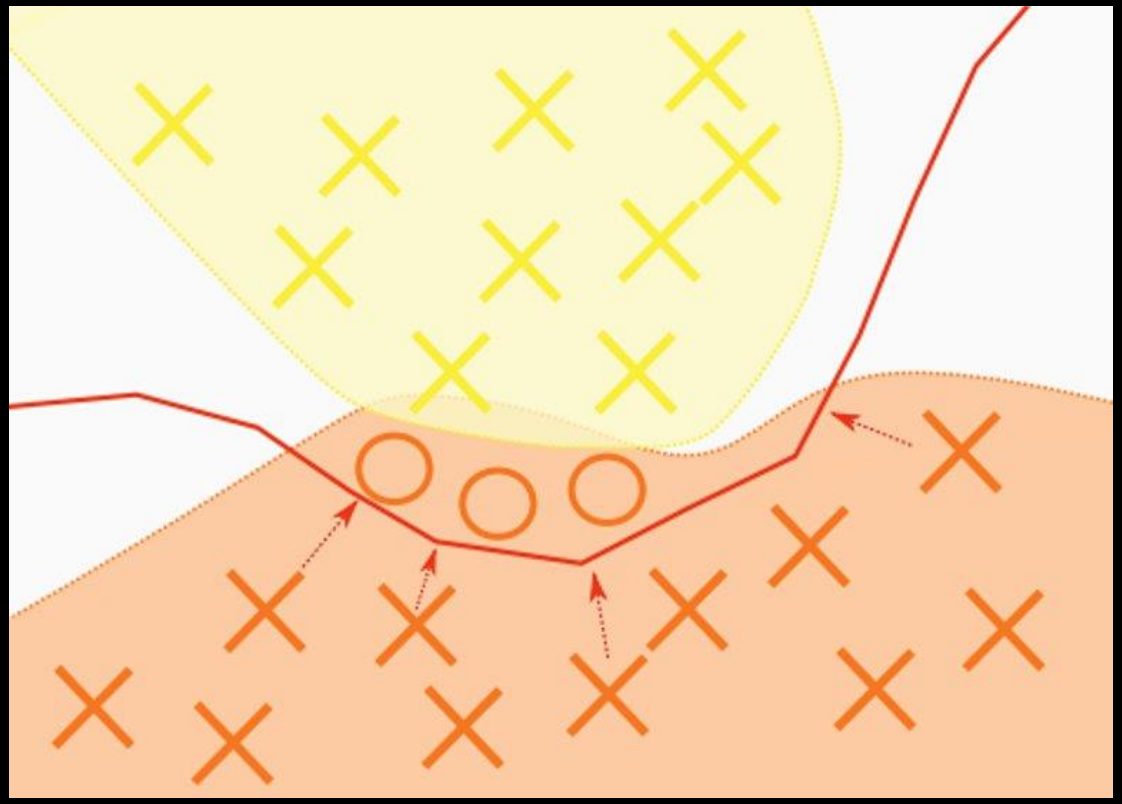
The **learned model** slightly differs from the true data distribution …

IDEA: THE SPACE OF ADVERSARIAL EXAMPLES



…. which makes room for adversarial examples.

ATTACK: USE THE ADVERSARIAL DIRECTIONS



Most attacks try to move inputs across the boundary.

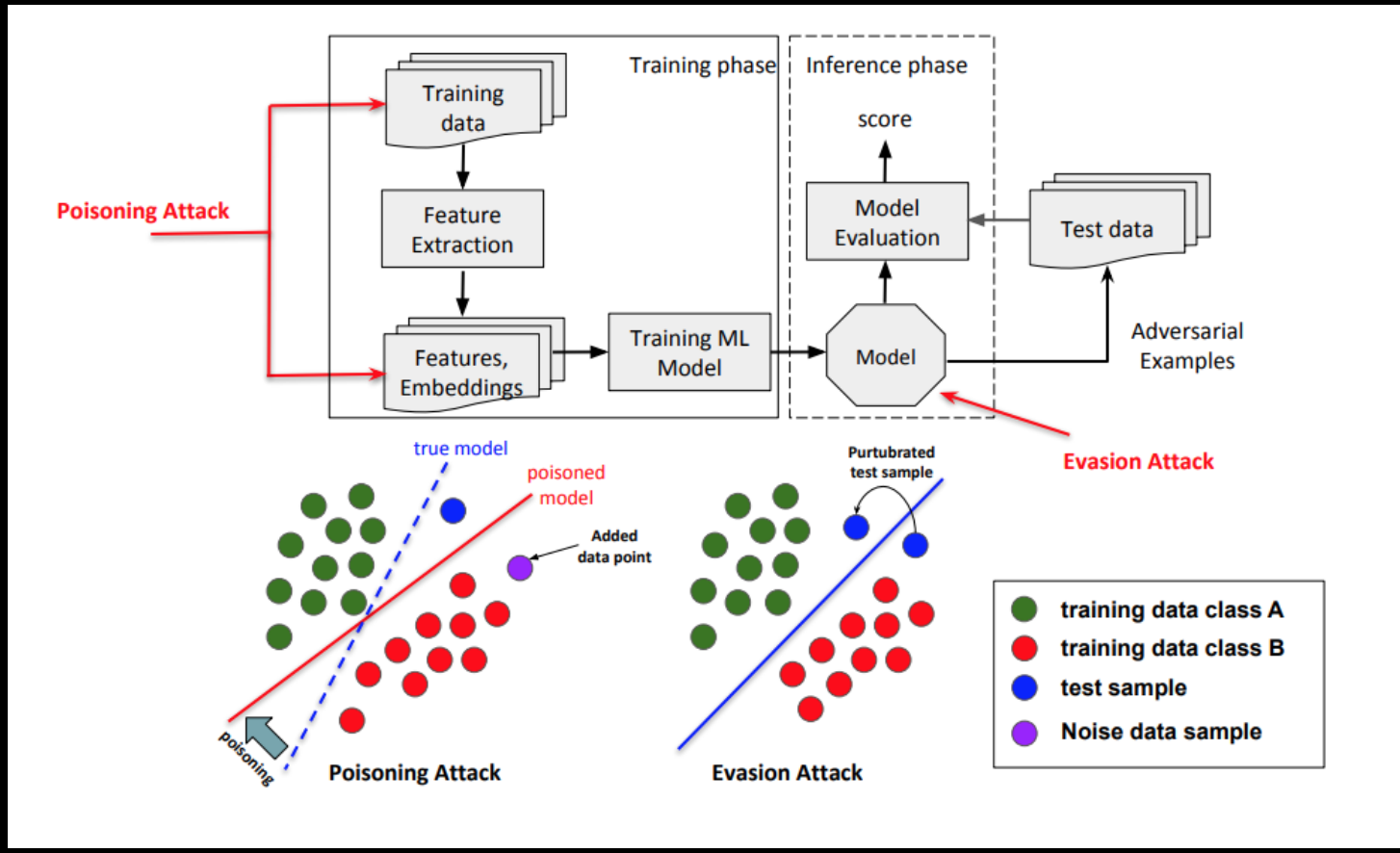
Attacking with a random noise does not work well in real experiment.

Foundations

TAXONOMY OF ADVERSARIAL ATTACKS

(1) Attack's timing:

- Training time (data poisoning)
  - occur before the ML model is trained or in the inference phase
- Inference time (evasive attack)
  - occur after the ML model is trained or in the inference phase
  - aim to evade detection — or evade the decisions made by the learned model

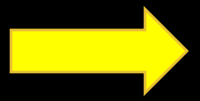


Politecnico di Bari

Picture Source: Yashar Deldjoo, Tommaso Di Noia, Felice Antonio Merra. A Survey on Adversarial Recommender Systems: From Attack/Defense Strategies to Generative Adversarial Networks,ACM Computing Surveys, March 2021 Article No.: 35 https://doi.org/10.1145/3439729

POISONING ATTACK APPLICATION IN LITERATURE

- Attack on Binary Classification
  - Label flipping attack
  - Kernel SVM
- Attack on unsupervised learning
  - Clustering
  - Anomaly detection
- Attack on matrix completion
  - Hand Engineered Poisoning
  - Mimicking user behavior
  - ML-optimized Strategies



Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines. arXiv preprint arXiv:1206.6389.

Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-169.

POISIONING ATTACK APPLICATION IN LITERATURE

- Attack on Binary Classification
  - Label flipping attack
  - Kernel SVM
- Attack on unsupervised learning
  - Clustering
  - Anomaly detection
- Attack on matrix completion
  - Hand Engineered Poisoning
  - Mimicking user behavior
  - ML-optimized Strategies

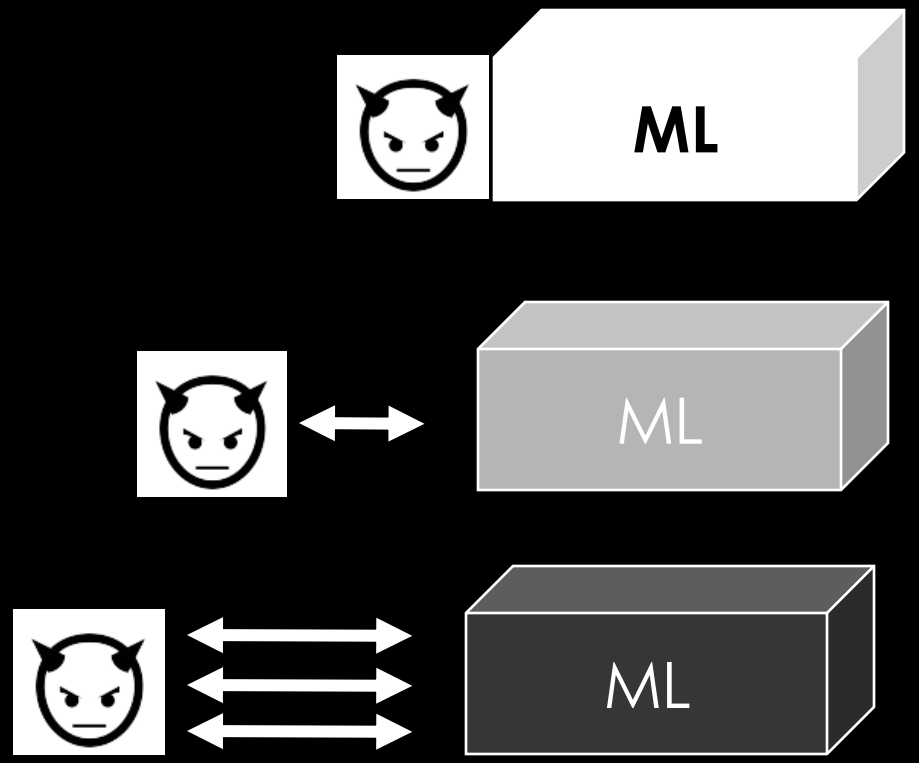
Commonly known as "shilling attack" in RecSys.

Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-169.

TAXONOMY OF ADVERSARIAL ATTACKS

(2) Attacker's knowledge:

- White-box attack
- Gray-box attack
- Black-box attack



TAXONOMY OF ADVERSARIAL ATTACKS

(3) Attacker's goal:

- Targeted attack: forces the classifier (ML model) to make predictions into a target class label.
- Untargeted attack (reliability attack): forces the classifier to make predictions into any incorrect class label.

CLASSICAL MACHINE LEARNING

Supervised learning (classification) problem

$$\min_{\Omega} J(\Omega, x, y)$$

ADVERSARIAL PERSPECTIVE

Supervised learning (classification) problem

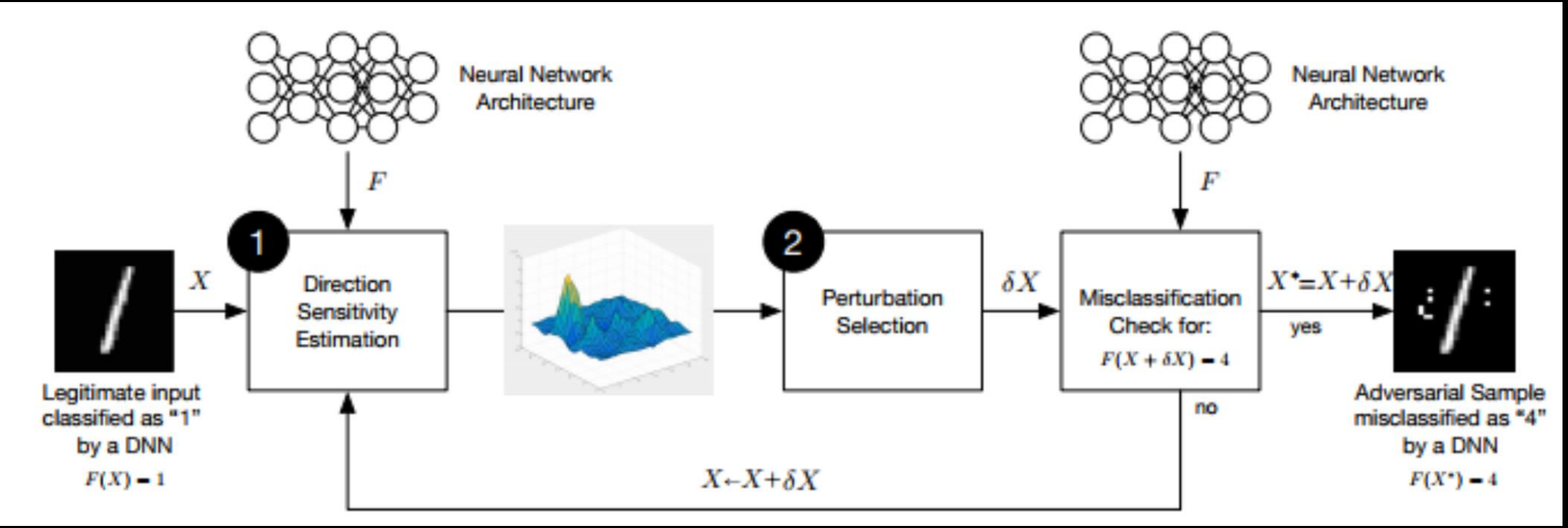
$$\arg\max_{\Delta_{adv}} J(\Omega, x + \Delta_{adv}, y) \;\; s.t., \|\Delta_{adv}\|_p \leq \epsilon$$

Adversarial perturbation of sample x

perturbation budget

Algorithms that aim to find such adversarial perturbations are referred to as adversarial attacks.

THE FRAMEWORK



CREDITS: N. Papernot et al. *Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks*. IEEE Symposium on Security and Privacy, SP 2016

ATTACK METHODS IN COMPUTER VISION DOMAIN

- **L-BFGS** [Szegedy et al., 2013]
  - Uses Limited-memory BFGS (L-BFGS) algorithm to solve the problem with **linear memory requirement**
- **FSGM -** Fast Gradient Sign Method [Goodfellow et al., ICLR '15]
  - Uses the **gradient** of the loss function to work on the costraint problem.
- **Carlini-Wagner** [Carlini and Wagner, 2017a]
  - Refines the L-BFGS attack to **defeat Defensive distillation**
- **JSMA -** Jacobian Saliency Map Attack [Papernot et al., 2015a]
  - Constructs an **input-output mapping** (forward derivatives) to find the minimal perturbation
- **DeepFool** [Moosavi-Dezfooli et al., 2015]
  - Performs an iterative attack to find **the closest decision boundary** to a given input

FGSM(INTUITION 1)
[GOODFELLOW ET AL., ICLR'15]

$$x' = x_0 + \Delta_{adv}$$

$$\omega^T \cdot x' = \omega^T \cdot x_0 + \omega^T \cdot \Delta_{adv}$$

GOAL: Maximize the factor $\omega^T \cdot \Delta_{adv}$ under the constraint $\|\Delta_{adv}\|_\infty \leq \epsilon$

IDEA: $\Delta_{adv} = \epsilon \cdot sign(\omega)$

FGSM(INTUITION 2)
[GOODFELLOW ET AL., ICLR'15]

$$\omega^T \cdot \Delta_{adv} = \epsilon \cdot (|\omega_1| + \cdots + |\omega_n|)$$

If we consider $m = avg(\omega_i)$ then the influence of the perturbation $\omega^T \cdot \Delta_{adv}$ grows as $\epsilon \cdot m \cdot n$

RESULT: the perturbation linearly grows with n

FGSM
[GOODFELLOW ET AL., ICLR'15]

Let Ω be the parameters of a ML model, x the input to the model, y the label of x, and $J(\Omega, x, y)$ be the loss function used for the model.

FSGM linearizes J around the fixed value of Ω, obtaining an optimal max-norm constrained adversarial perturbation.

$$\Delta_{avd} = \epsilon \cdot \mathrm{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\Omega}, \boldsymbol{x}, y)\right)$$

UNTARGETED ADVERSARIAL ATTACK

$$\min_{\Delta_{adv}} \|\Delta_{adv}\|$$

$$s.t.: f(x_0 + \Delta_{adv}, \Omega) \neq y_0$$

Constrained optimization problem formulation

$$=$$

$$\max_{\|\Delta_{adv}\| \leq \epsilon} J(\Omega, x_0 + \Delta_{adv}, y')$$

UnConstrained optimization problem formulation

Politecnico di Bari

TARGETED ADVERSARIAL ATTACK

$$\min_{\Delta_{adv}} \|\Delta_{adv}\|$$

$$s.t.: f(x_0 + \Delta_{adv}, \Omega) = y_T$$

Constrained optimization problem formulation

=

$$\max_{\|\Delta_{adv}\| \leq \epsilon} J(\Omega, x_0 + \Delta_{adv}, y_T)$$

UnConstrained optimization problem formulation

COUNTERMEASURES

- **Proactive** countermeasures
  - **Adversarial Training** [Goodfellow et al., ICLR '15]
    - Additional training epochs with adversarial examples
  - **Defensive Distillation** [Papernot et al., ISS'16]
    - Adapt distillation to increase the robusteness of the network
  - **Robust Optimization** [Madry et al., ICLR'18]
    - design robust DNN to prevent a speciic class of adversarial examples
- **Reactive** countermeasures
  - **Adversarial Detecting**
  - **Input Reconstruction**
  - **Network Verification**

ADVERSARIAL TRAINING
[GOODFELLOW ET AL., ICLR'15]

Including adversarial samples in the **training** of a model makes it **more robust**.
The objective function of the model **adversarially-trained** is:

Adversarial Perturbation

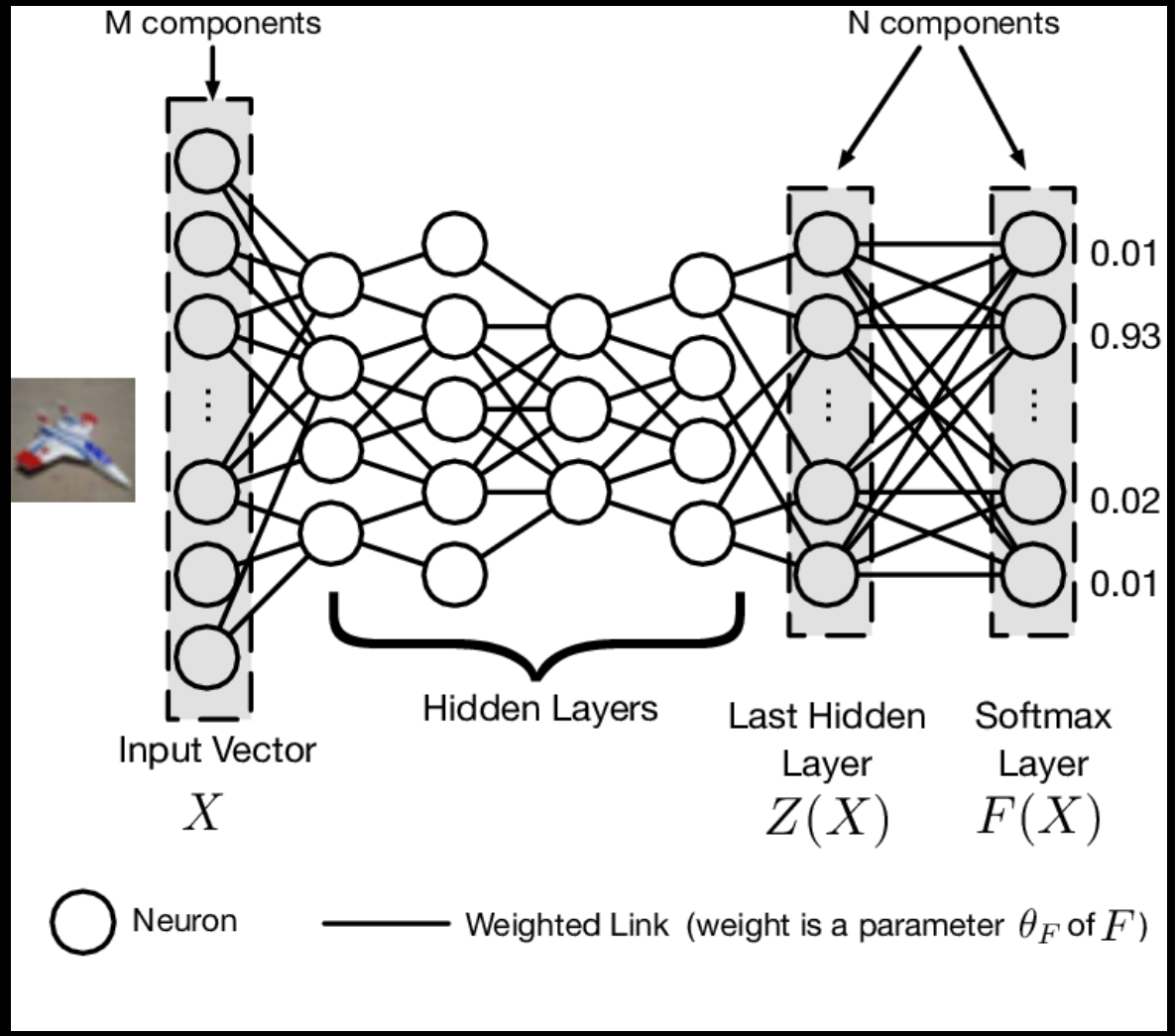
$$J(\boldsymbol{\Omega}, \boldsymbol{x}, y) + \lambda J(\boldsymbol{\Omega}, \boldsymbol{x} + \epsilon \cdot \operatorname{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\Omega}, \boldsymbol{x}, y)))$$

Adversarial Regularization term

Adversarial training provides better generalization performance
[Miyato et al., ICLR'17]

DEFENSIVE DISTILLATION

[PAPERNOT ET AL., ISS'16]

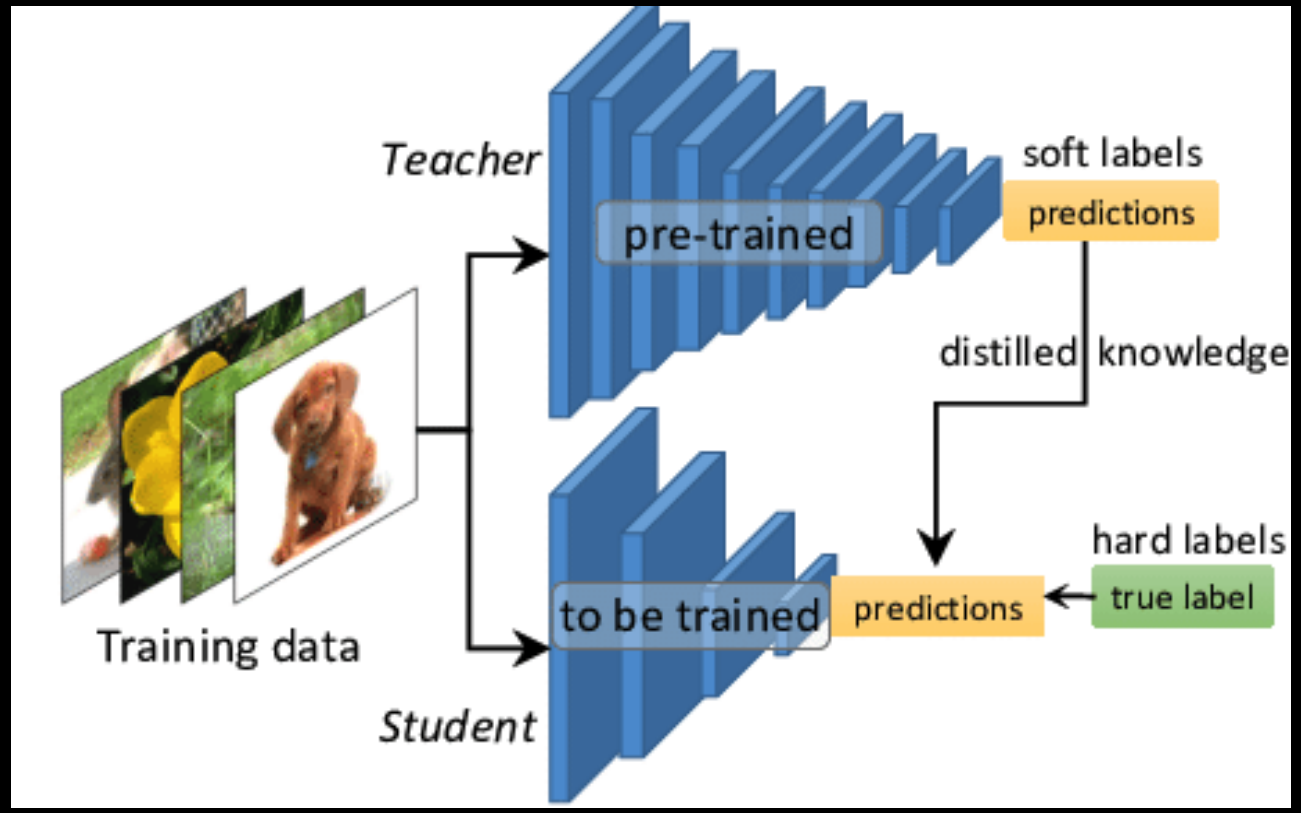


$$F(X) = \left[\frac{e^{z_i(X)/T}}{\sum_{l=0}^{N-1} e^{z_l(X)/T}} \right]_{i \in 0..N-1}$$

Politecnico di Bari

DEFENSIVE DISTILLATION

[PAPERNOT ET AL., ISS'16]

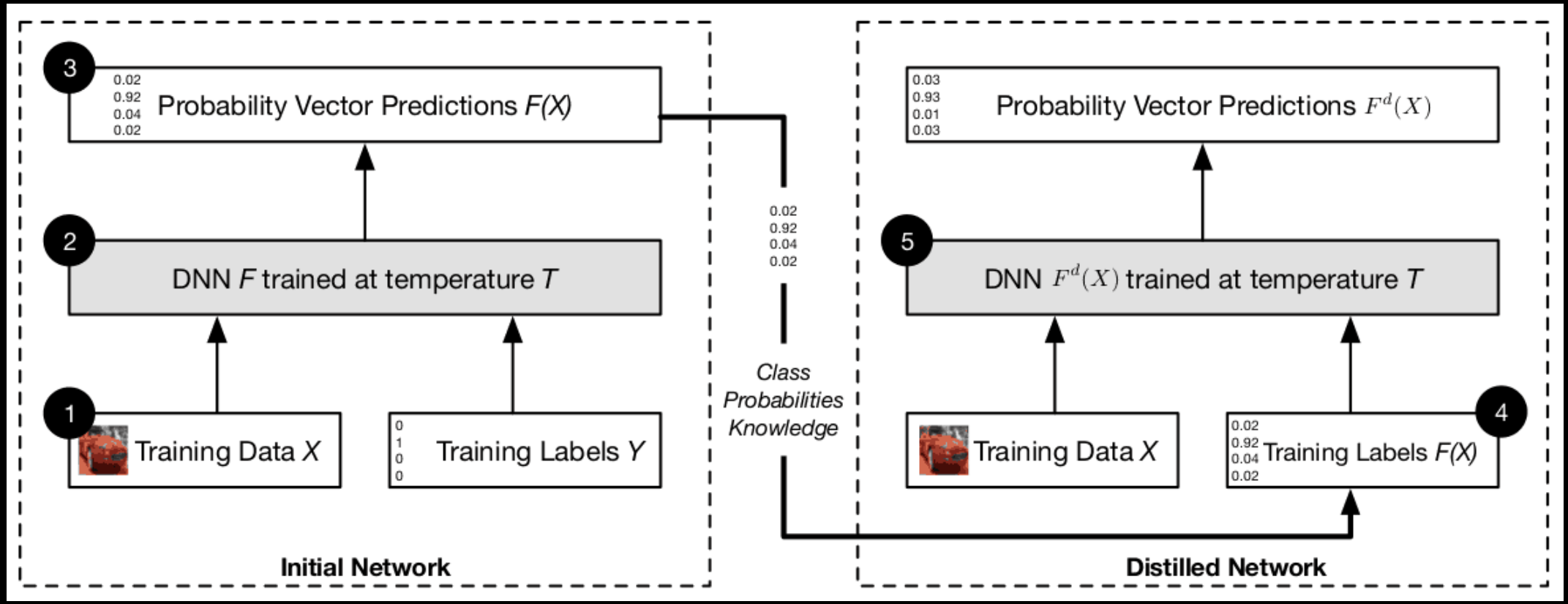


Credits: https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

Originally proposed for model compression

DEFENSIVE DISTILLATION

[PAPERNOT ET AL., ISS'16]



Politecnico di Bari

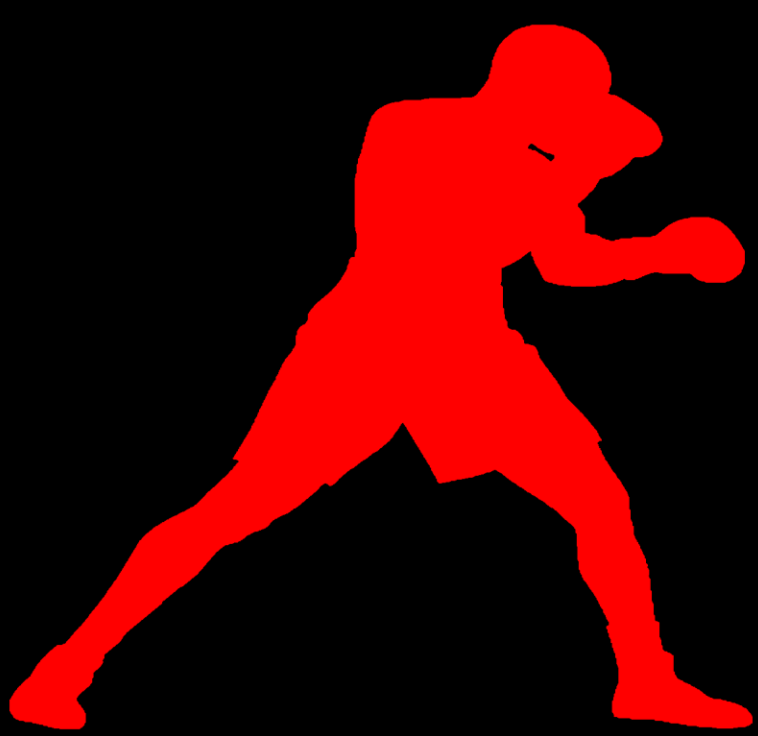
ADVERSARIAL ROBUSTNESS

Supervised learning (classification) problem

$$\min_{\Omega} \max_{\Delta_{adv}} J(\Omega, x + \Delta_{adv}, y)$$

The above min-max formulation is used to capture the notion of security against adversarial attacks = Robust Optimization

ATTACK-DEFENSE GAME

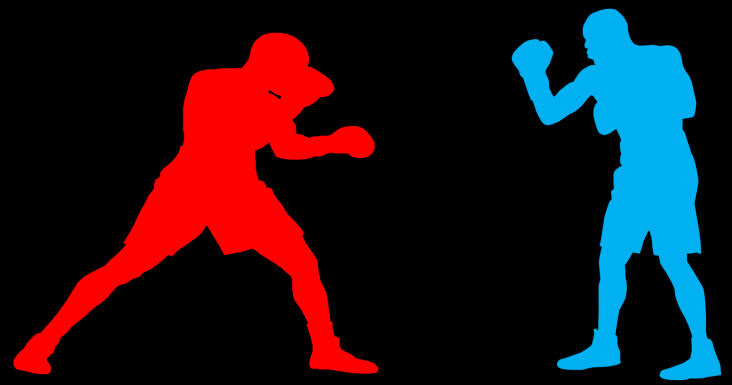


Politecnico di Bari

ATTACK-DEFENSE GAME

The attack-defence game is a **MINIMAX GAME:**

- in the **literature** for each **ATTACK** there is a **DEFENSE** strategy
- in the **adversarial training** we **MINIMIZE** the Loss and **MAXIMIZE** the perturbation
- **Generative Adversarial Network (GAN)** is trained with an **ATTACKER** that tries to alter a **DEFENDER**



Politecnico di Bari

WHY STUDY GAN?

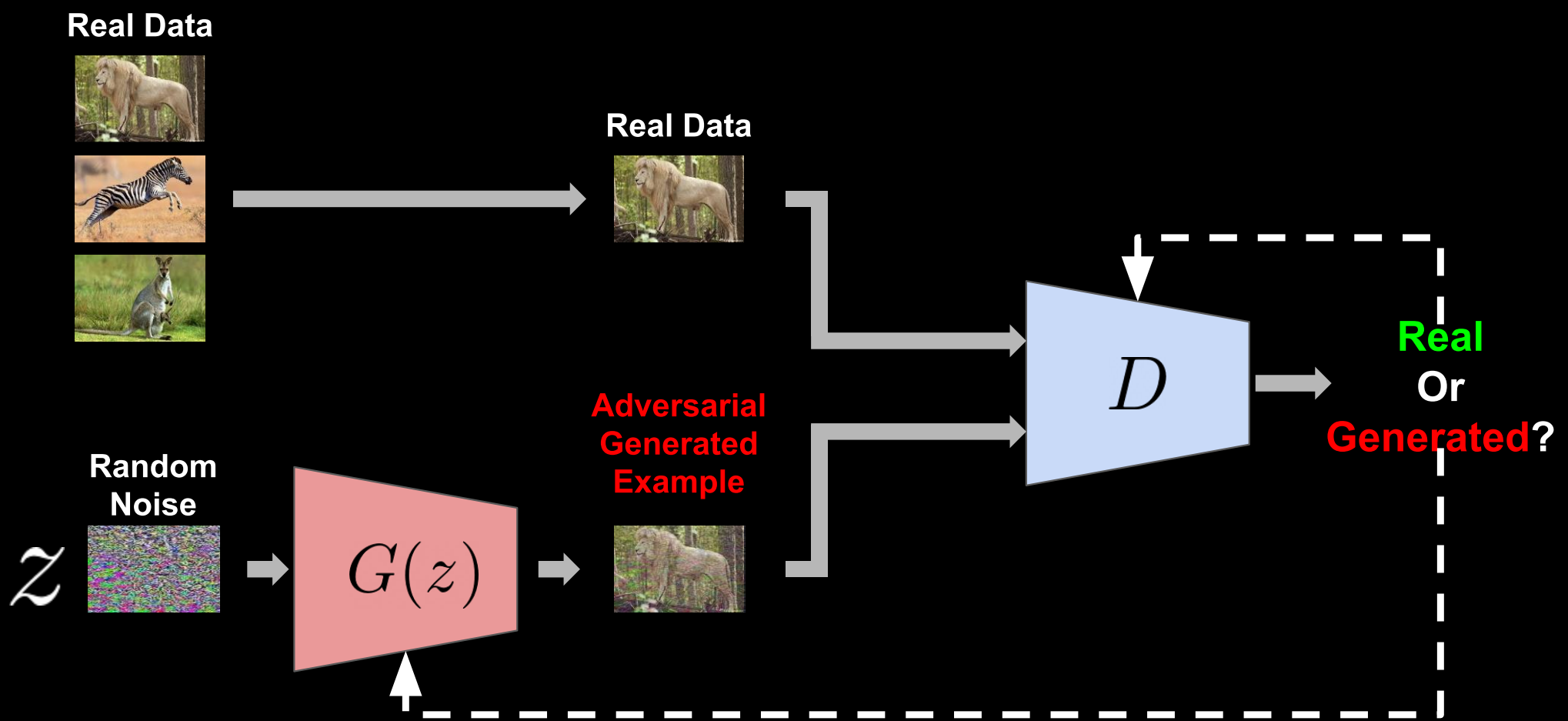
- Generate Examples for Image Datasets
- Generate Photographs of Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Pose Guided Person Image Generation
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation

- Photos to Emojis
- Photograph Editing
- High-resolution image synthesis
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting (Style-transfer)
- Clothing Translation
- Video-motion Prediction
- 3D Object Generation

HOW DO GANS WORK?

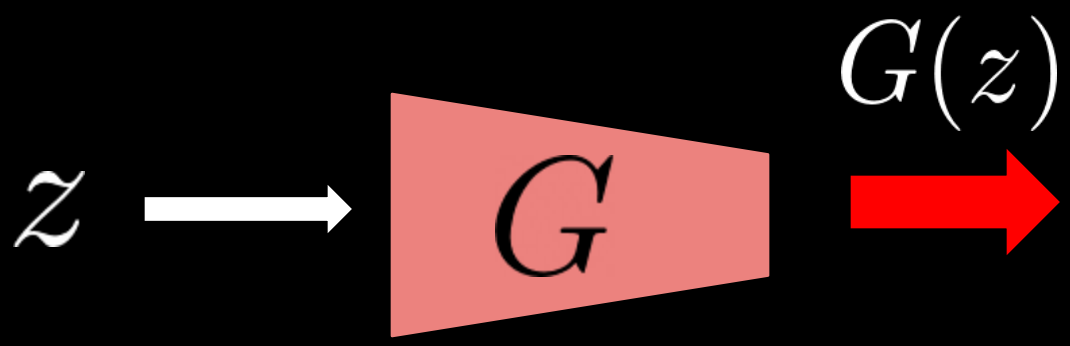
Normal Classifier

HOW DO GANS WORK?



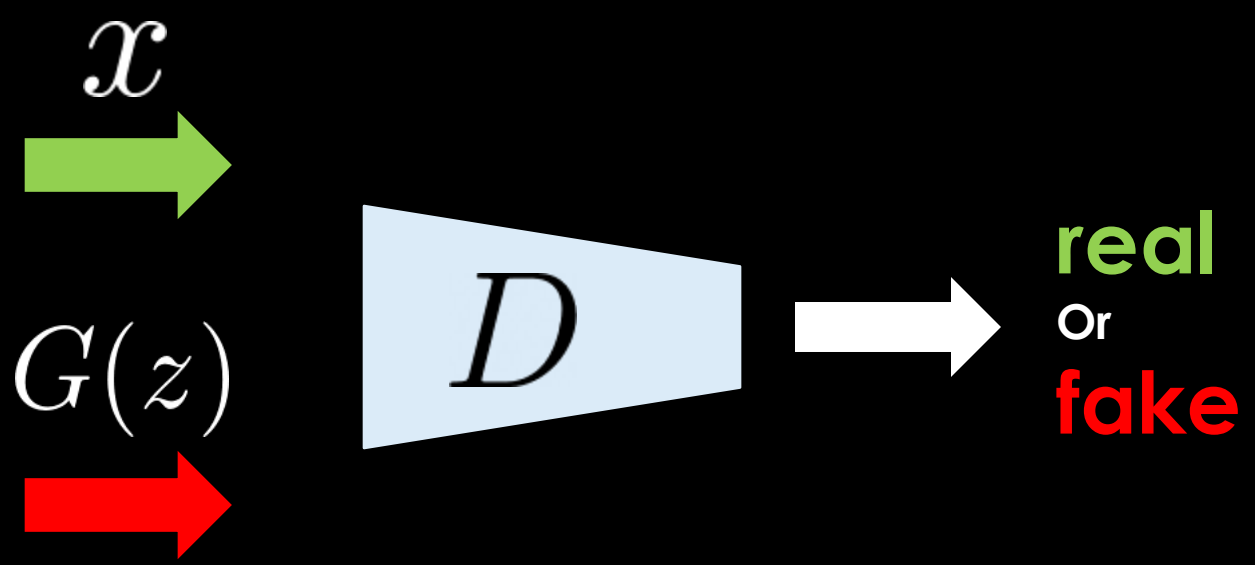
GENERATOR

GOAL? Maps a Gaussian random noise z to a **Fake** data to **FOOL** the discriminator

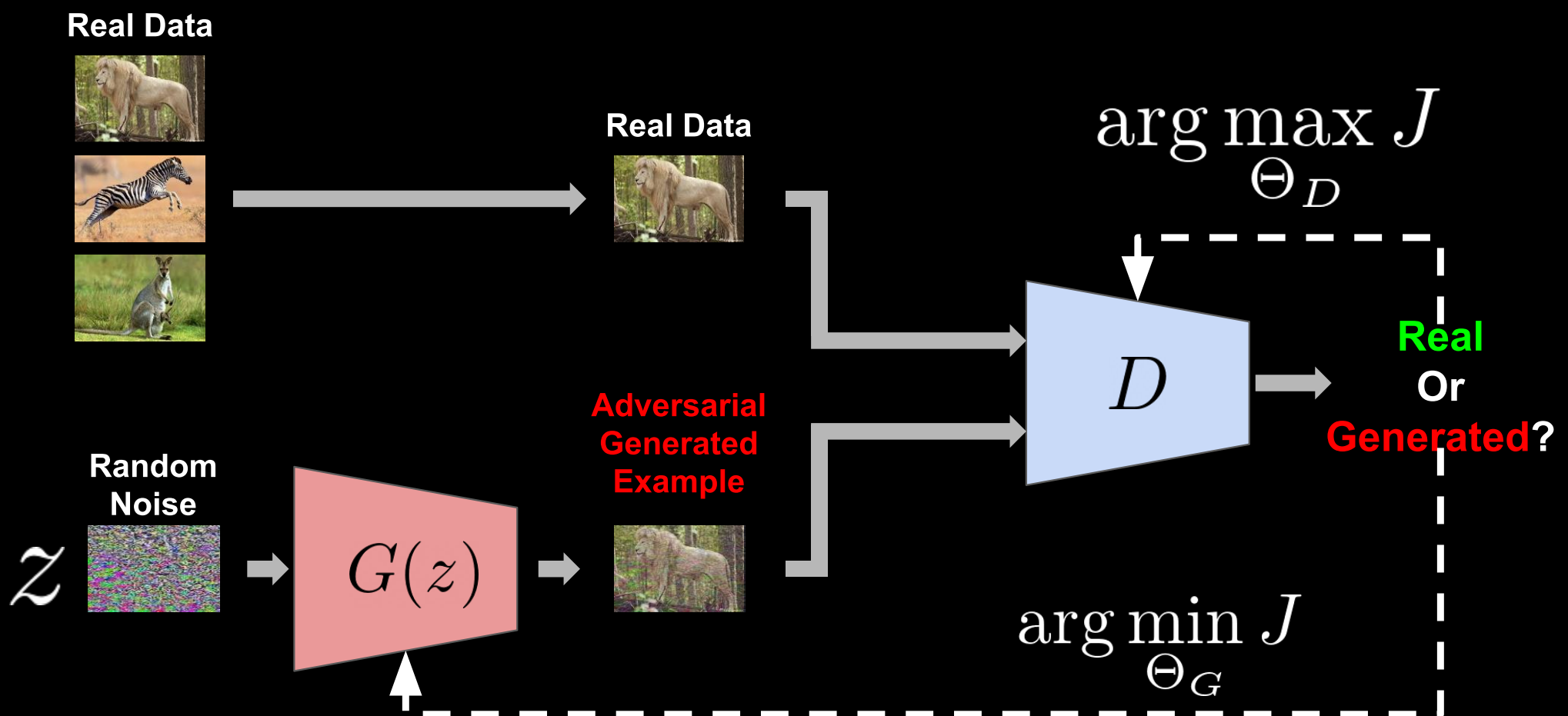


DISCRIMINATOR

GOAL? Output the probability that its input is **real** rather than **fake**



HOW DO GANS WORK?



Politecnico di Bari

MINIMAX GAME

$$J = \mathbb{E}_{\boldsymbol{x} \sim \rho_{\text{data}}\,(\boldsymbol{x})} \left[\log D(\boldsymbol{x})\right] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(z)} \left[\log\left(1 - D\left(G(\boldsymbol{z})\right)\right)\right]$$

DISCRIMINATOR

$$\arg\max_{\Theta_D} J$$

GENERATOR

$$\arg\min_{\Theta_G} J$$

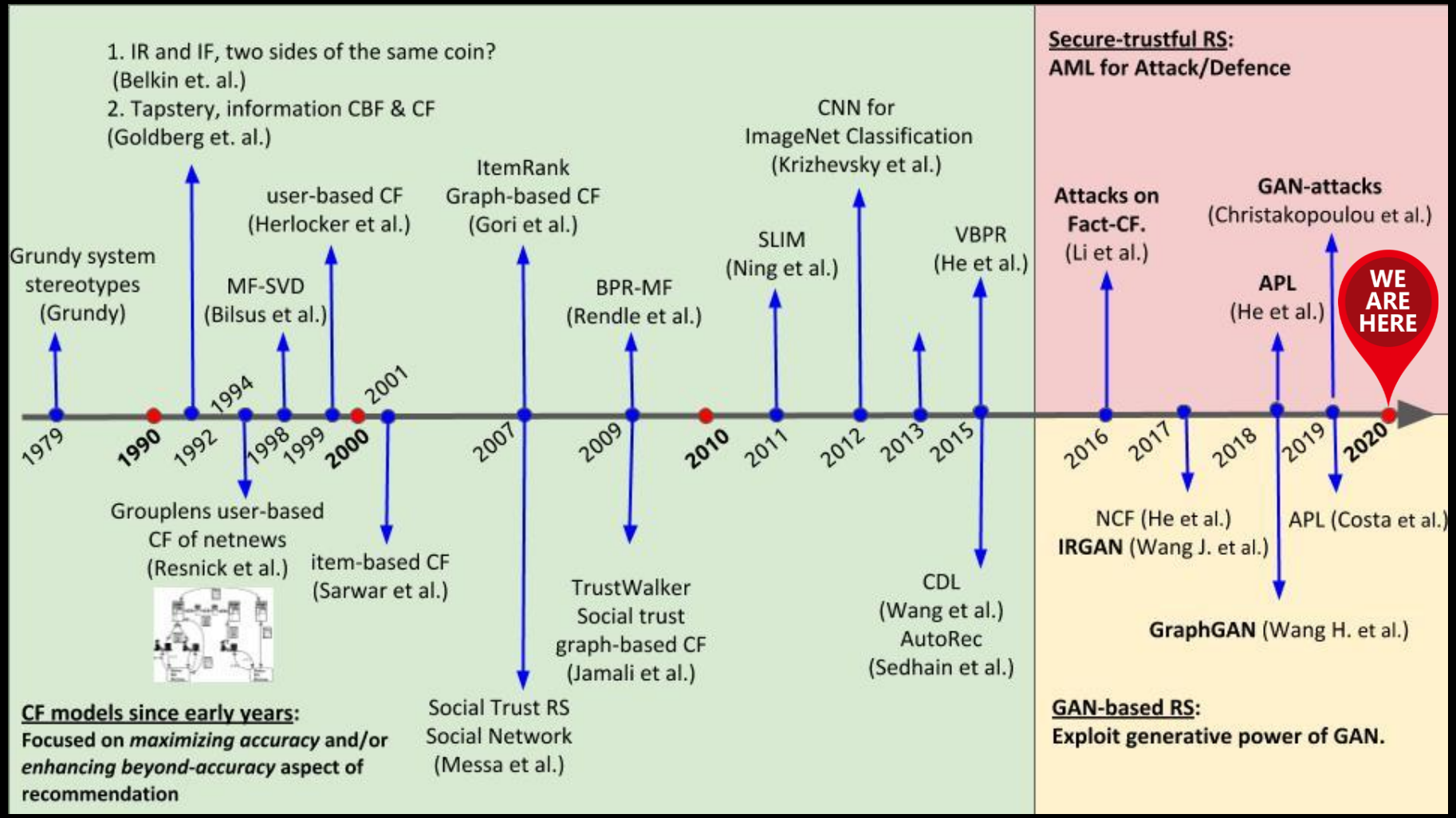
MINIMAX GAME

$$\arg\min_{\Theta_G} \max_{\Theta_D} J$$

Politecnico di Bari

PART 2
ADVERSARIAL MACHINE LEARNING FOR RECOMMENDER SYSTEMS

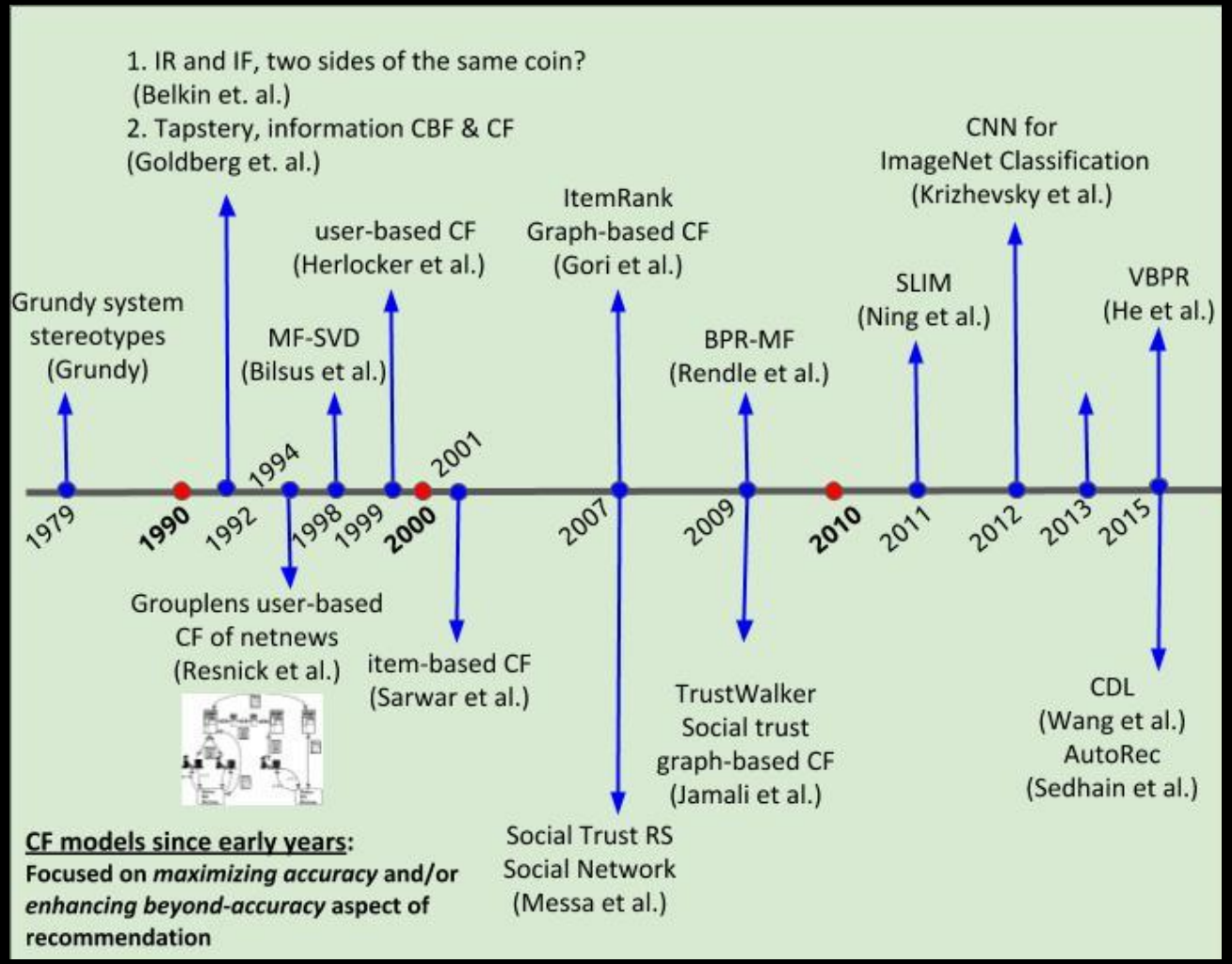
MILESTONES IN RS DEVELOPMENT



Politecnico di Bari

Deldjoo, Y., Noia, T. D., & Merra, F. A. (2021). A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.

MILESTONES IN RS DEVELOPMENT

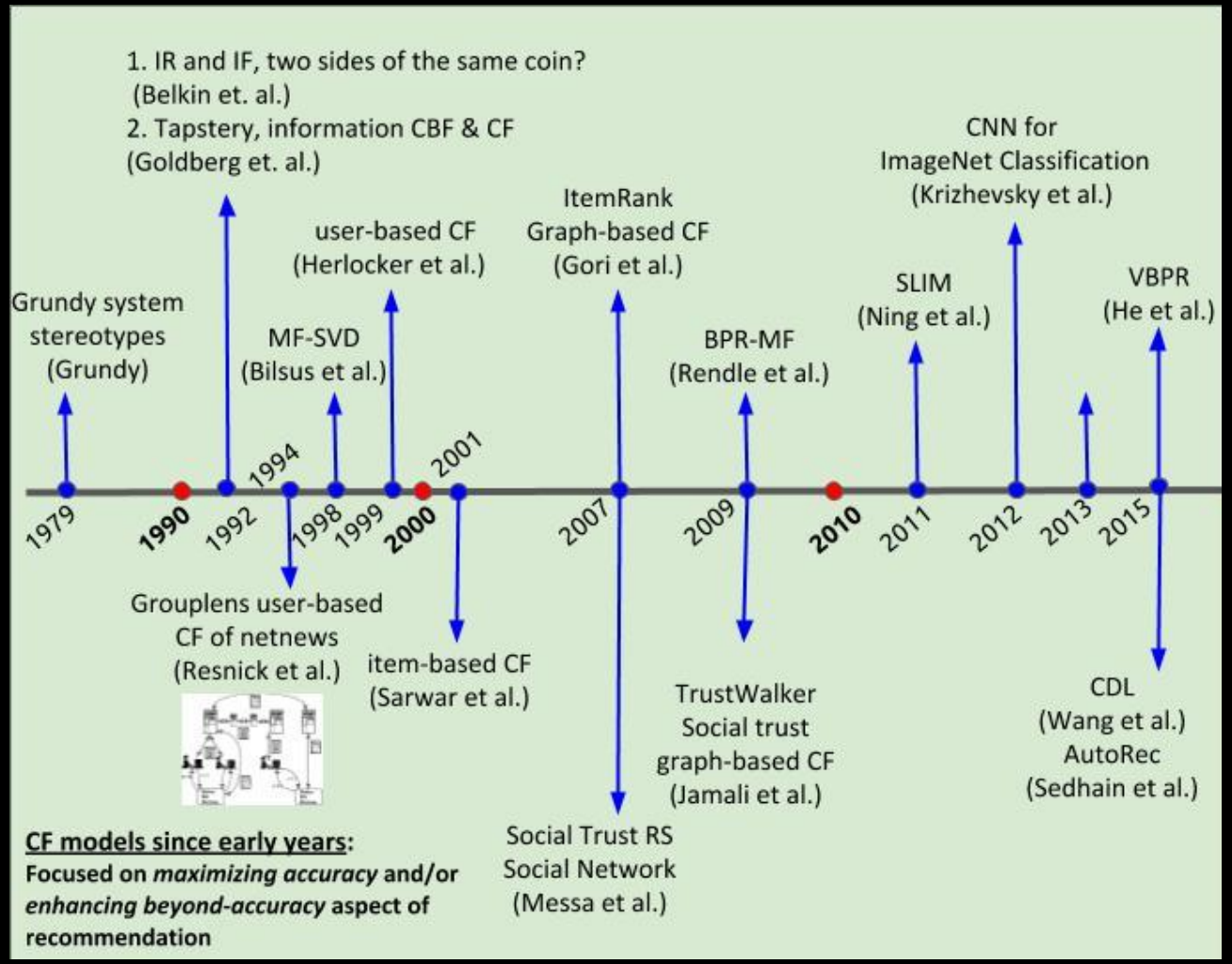


- RS based on **classical ML**
  - CF
  - Generic feature-based models

Politecnico di Bari

Deldjoo, Y., Noia, T. D., & Merra, F. A. (2021). A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.

MILESTONES IN RS DEVELOPMENT



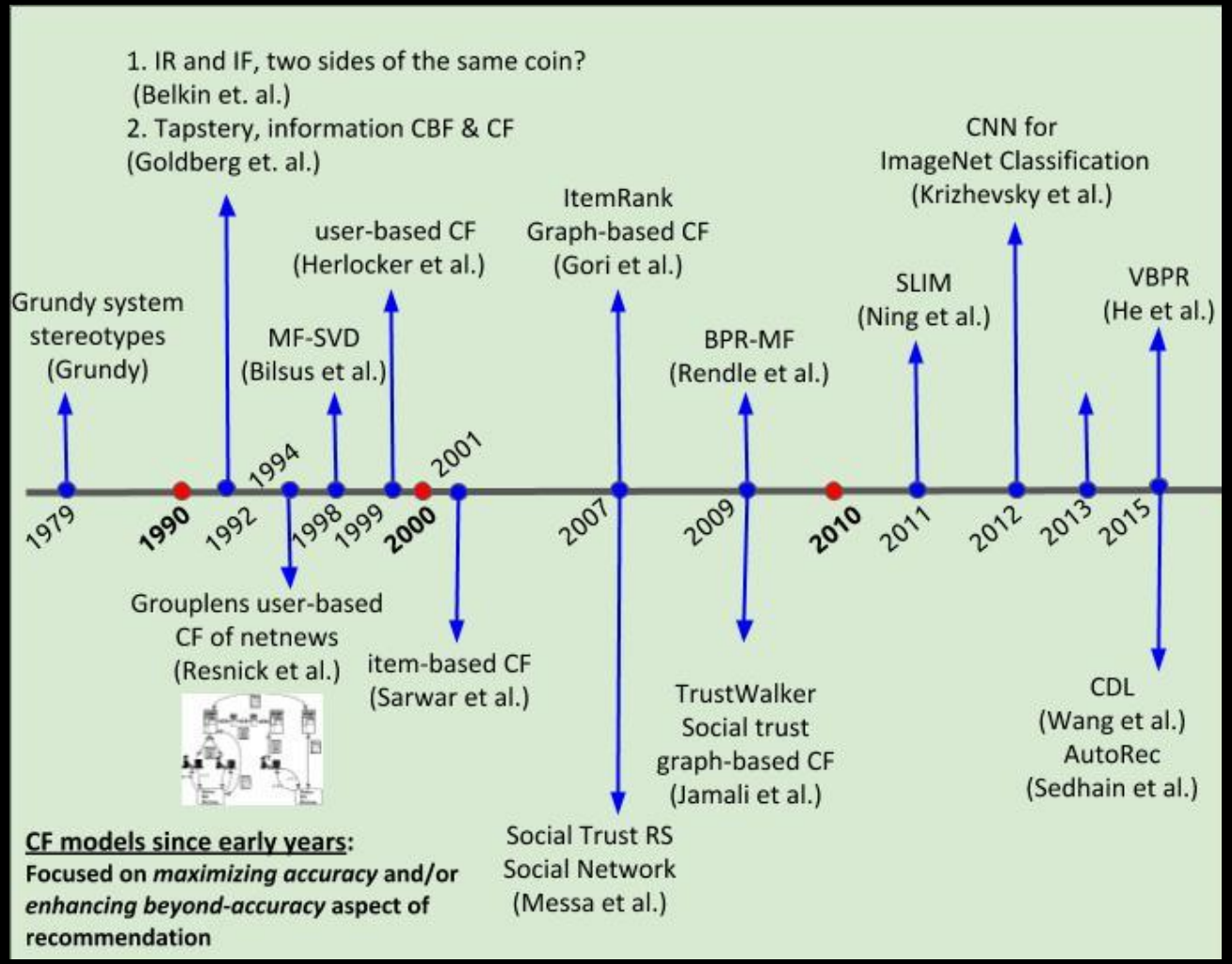
CF MODELS

- (1994-2004) **Memory-based**
  - User similarity (Herlocker, Thomas)
  - Item Similarity (Sarwar, Greg Linden)
  - Cosine/pearson similarity (Stuart/Paul)
- (2009) **Model-based CF**
  - MF (Koren)
  - BPR (Rendle)
- (2009) **Model-based CF (item-based)**
  - FISM (Kabor)
  - SLIM (NING)
- (2016-2020) **Deep-learning based models**
  - NeuMF (He)
  - DeepMF (Xue)
  - ACF (Chen)

Politecnico di Bari

Deldjoo, Y., Noia, T. D., & Merra, F. A. (2021). A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.

MILESTONES IN RS DEVELOPMENT



Generic Feature-based MODELS

- (2010-2016) **Memory-based**
  - FM (Rendle)
  - FFM (Juan)

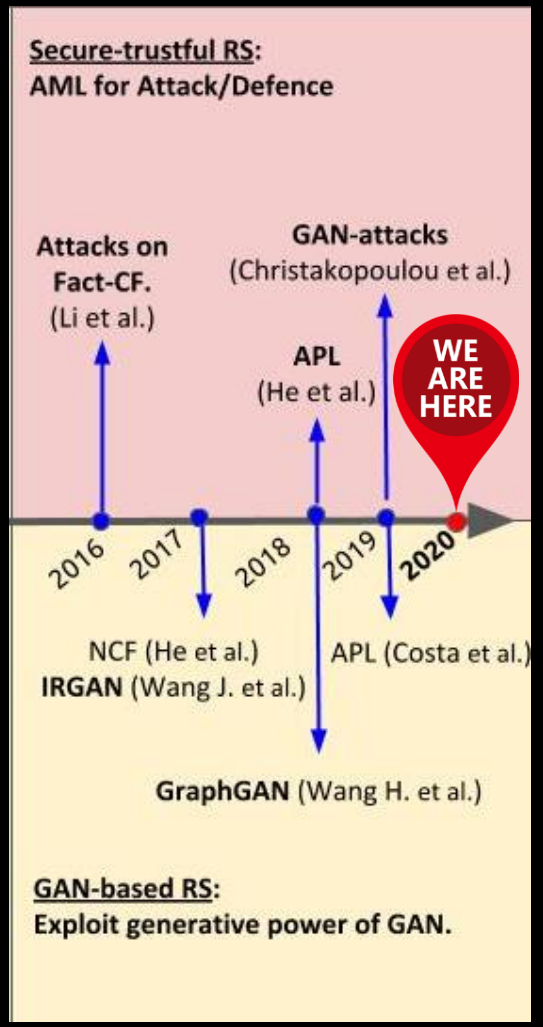
- (2016-2019) **Deep-Learning based models**
  - NFM (He)
  - Wide and Deep (Cheng)
  - Deep FM (Guo)
  - xDeep FM (Lian)
  - FNN (Zhang)
  - CrossNet (Wang)
  - TEM (Wang)

Politecnico di Bari

Deldjoo, Y., Noia, T. D., & Merra, F. A. (2021). A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.

ADVERSARIAL MACHINE LEARNING IN RS

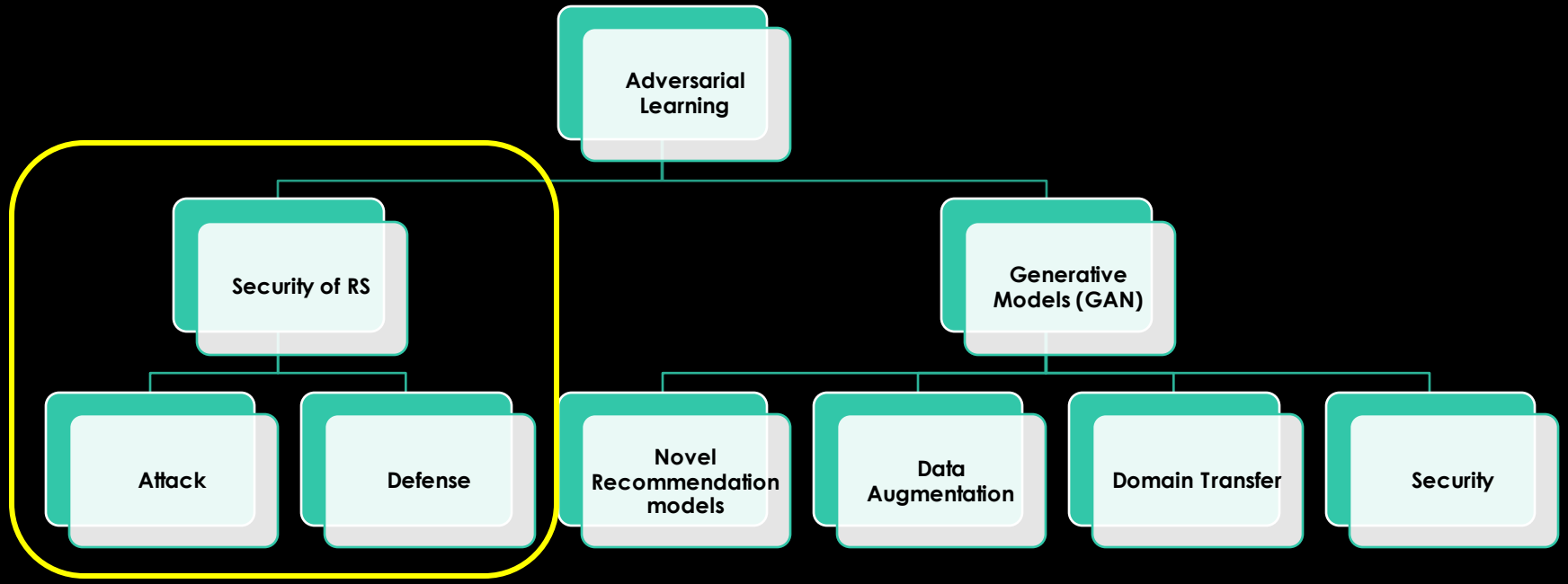
- RS based on adversarial machine learning
  - Adversarial examples (secruity of RS)
  - GAN-based models (generative models)

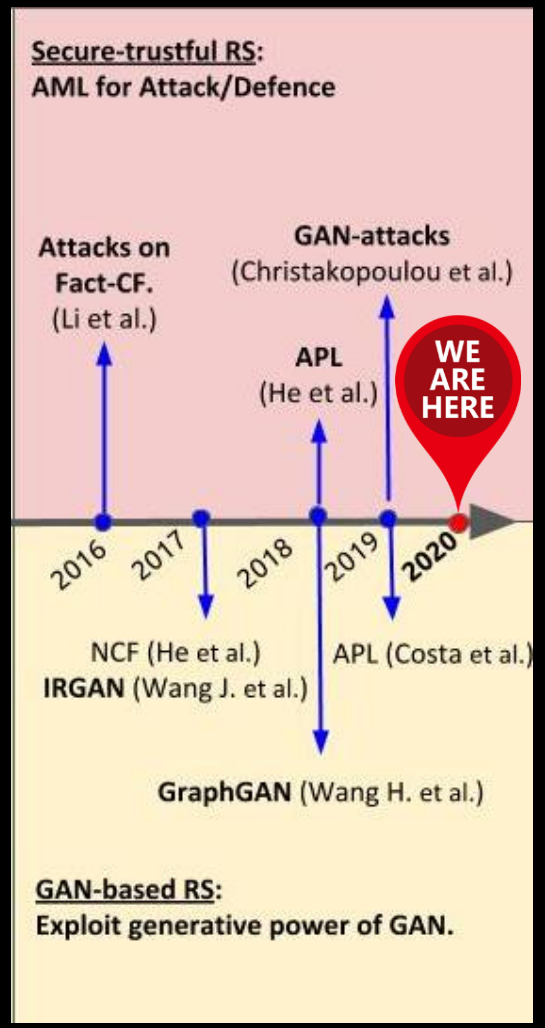


Politecnico di Bari

ADVERSARIAL MACHINE LEARNING IN RS

- RS based on adversarial machine learning
  - Adversarial examples (secruity of RS)
  - GAN-based models (generative models)





Politecnico di Bari

ATTACKS AGAINIST RECOMMENDER SYSTEMS

Adversarial Sample Generation

Hand-Engineered

- Fake users and items
  - Leveraging interaction data (Deldjoo'20)
  - Leveraging semantic data (Anelli'20)

Hand-eng. poisoning or Shilling attacks (2000-now)

Machine-Generated

- Fake users and items
  - Factorization-based models (Li'16)
  - RL-based models (Zhang'20)

MG poisoning attacks (2016-now)

- Adversarial noise/example
  - Attacks on CF model params (He'18)
  - Attacks based on content (Di Noia'20, Tang'19)

Adversarial attacks (2016-now)

Deldjoo, Y., Di Noia, T., Di Sciascio, E., & Merra, F. A. How dataset characteristics affect the robustness of collaborative recommendation models. SIGIR'20.

Anelli, V. W., Deldjoo, Y., Di Noia, T., Di Sciascio, E., & Merra, F. A. Sasha: Semantic-aware shilling attacks on recommender systems exploiting knowledge graphs. ESWC'20.

Li, B., Wang, Y., Singh, A., & Vorobeychik, Y. Data poisoning attacks on factorization-based collaborative filtering. NIPS'16.

Zhang, H., Li, Y., Ding, B., & Gao, J. Practical Data Poisoning Attack against Next-Item Recommendation. WWW'20.

He, X., He, Z., Du, X., & Chua, T. S. (2018, June). Adversarial personalized ranking for recommendation. SIGIR'18.

Di Noia, T., Malitesta, D., & Merra, F. A. Taamr: Targeted adversarial attack against multimedia recommender systems. DSNW'20.

ATTACKS AGAINIST RECOMMENDER SYSTEMS

Adversarial Sample Generation

Hand-Engineeringed

- Fake users and items
  - Leveraging interaction data (Deldjoo'20)
  - Leveraging semantic data (Anelli'20)

Hand-eng. poisoning or Shilling attacks (2000-now)

Machine-Generated

- Fake users and items
  - Factorization-based models (Li'16)
  - RL-based models (Zhang'20)

MG poisoning attacks (2016-now)

- Adversarial noise/example
  - Attacks on CF model params (He'18)
  - Attacks based on content (Di Noia'20, Tang'19)

Adversarial attacks (2016-now)

Adversarial component

Input Data

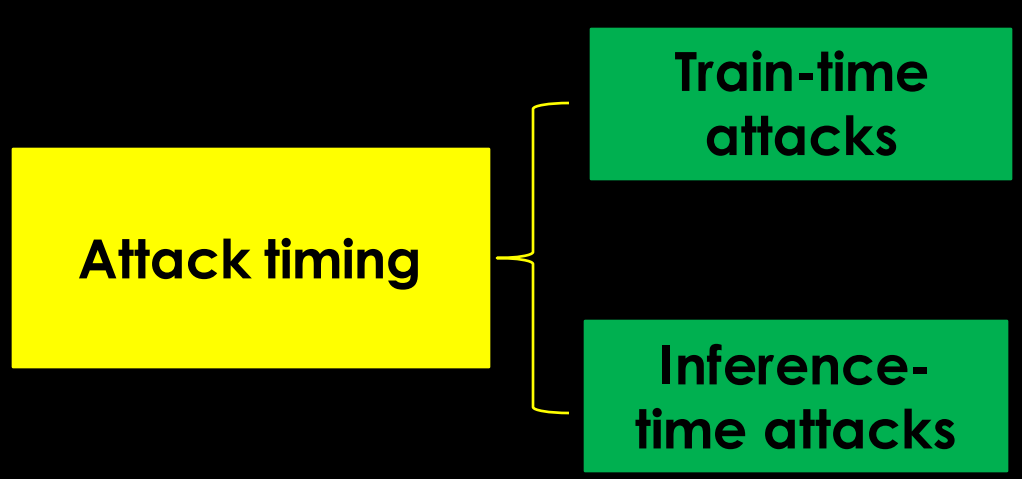
- U-I interaction
- User content/attributes (reviews, demographic)
- Item content
- Fake users and items

Rec. Model

- User or item embedding

Politecnico di Bari

ATTACKS AGAINIST RECOMMENDER SYSTEMS



- Poisoning attacks (profile injection)
  - Hand-crafted shilling attacks
  - MG-data poisoning attacks

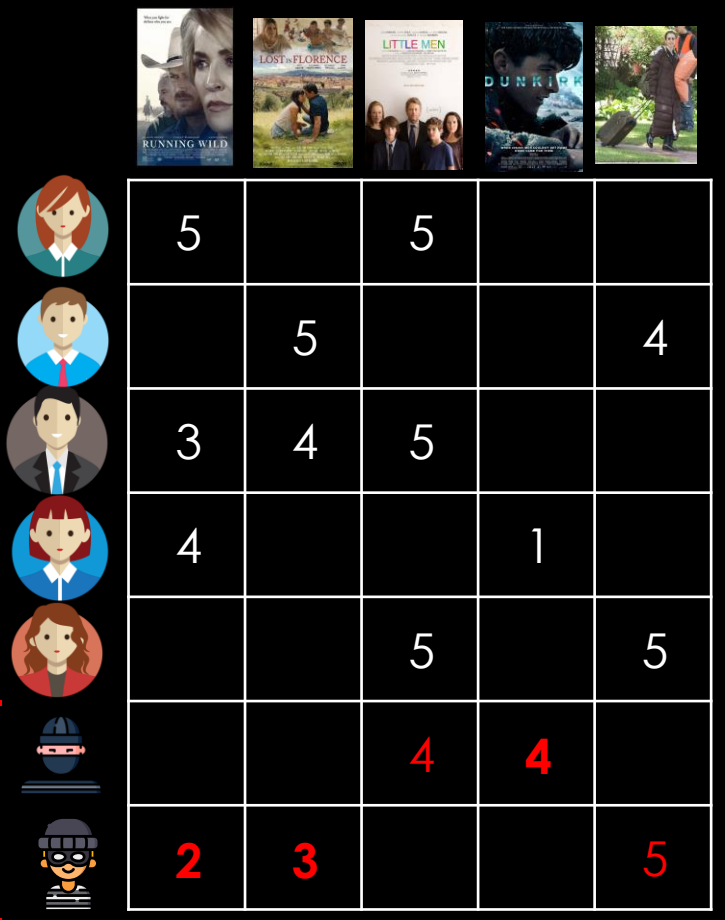
- Evasive attacks: common e.g., adversarial attacks (profile pollution)

HAND-CRAFTED SHILLING ATTACKS

Problem: Given a U-I matrix, the goal is to add a small number of fake users, where each new profile can have maximum 'C' ratings.

Different attack types: Constructed based on the composition a of user profile.
(e.g, random, popular, bandwagon, love-hate)

I_S			I_F			$I_\emptyset$			I_T
$i_s^{(1)}$	...	$i_s^{(\alpha)}$	$i_f^{(1)}$	...	$i_f^{(\phi)}$	$i_\emptyset^{(1)}$	...	$i_\emptyset^{(\chi)}$	i_t



5		5		
	5			4
3	4	5		
4			1	
		5		5
		4	**4**	
2	**3**			5

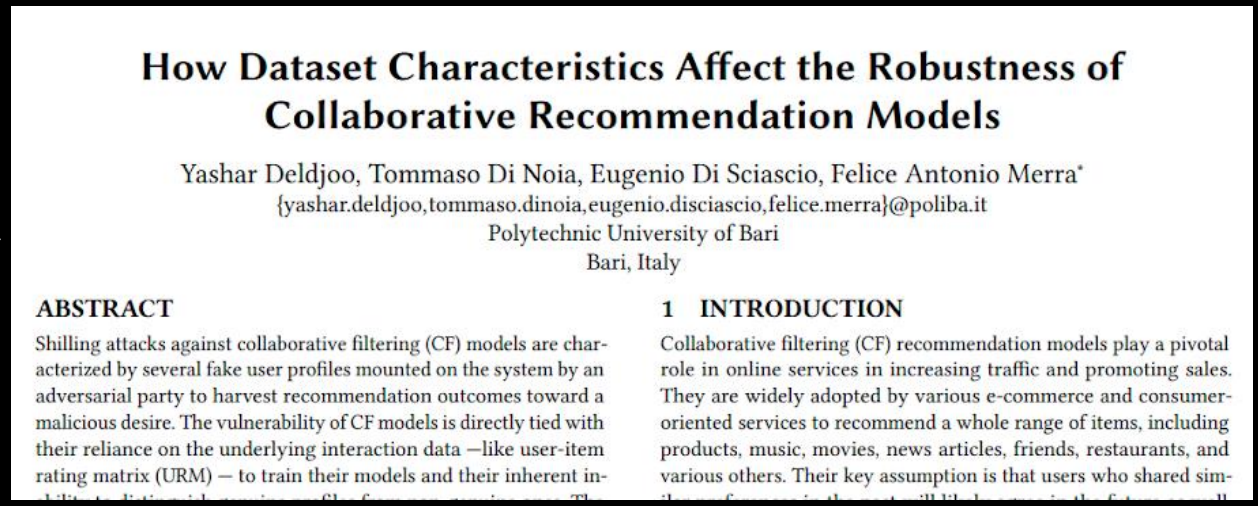
Gunes, I., Kaleli, C., Bilge, A., & Polat, H. (2014). Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review*,'14.

Politecnico di Bari

HAND-CRAFTED SHILLING ATTACKS AGAINIST RS

Recent advances focuses on:

Goal (attack): Study the Impact of Dataset Characteristics on the efficacy of most popular CF shilling attacks



How Dataset Characteristics Affect the Robustness of Collaborative Recommendation Models

Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, Felice Antonio Merra*
{yashar.deldjoo,tommaso.dinoia,eugenio.disciascio,felice.merra}@poliba.it
Polytechnic University of Bari
Bari, Italy

ABSTRACT

Shilling attacks against collaborative filtering (CF) models are characterized by several fake user profiles mounted on the system by an adversarial party to harvest recommendation outcomes toward a malicious desire. The vulnerability of CF models is directly tied with their reliance on the underlying interaction data —like user-item rating matrix (URM) — to train their models and their inherent in-

1 INTRODUCTION

Collaborative filtering (CF) recommendation models play a pivotal role in online services in increasing traffic and promoting sales. They are widely adopted by various e-commerce and consumer-oriented services to recommend a whole range of items, including products, music, movies, news articles, friends, restaurants, and various others. Their key assumption is that users who shared sim-

$$\mathbf{y} = \epsilon + \theta_0 + \theta_d \mathbf{X}_d + \theta_c \mathbf{X}_c$$

$$\mathbf{y} \rightarrow \Delta_{HR@k} = \hat{HR}@k - HR@k$$

$$x \rightarrow \text{data characteristics}$$

$$x_1 = \log_{10}\left(\frac{|\mathcal{U}| \cdot |\mathcal{I}|)}{sc}\right) \qquad x_4 = 1 - 2\sum_{i=1}^{|\mathcal{I}|}\left(\frac{|\mathcal{I}| + 1 - i}{|\mathcal{I}| + 1}\right) \times \left(\frac{|\mathcal{K}_i|}{|\mathcal{K}|}\right)$$

$$x_2 = \log_{10}\left(\frac{|\mathcal{U}|}{|\mathcal{I}|}\right) \qquad x_5 = 1 - 2\sum_{u=1}^{|\mathcal{U}|}\left(\frac{|\mathcal{U}| + 1 - u}{|\mathcal{U}| + 1}\right) \times \left(\frac{|\mathcal{K}_u|}{|\mathcal{K}|}\right)$$

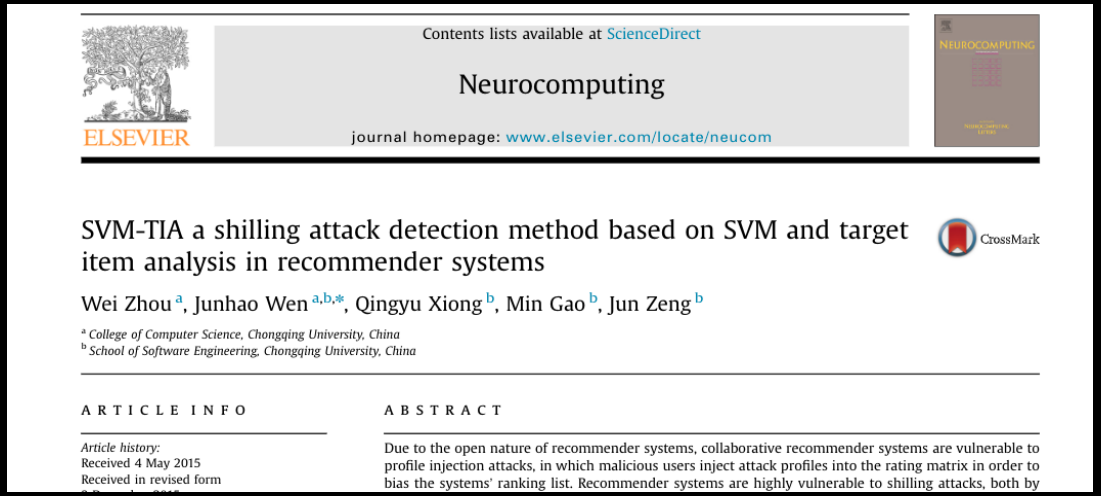
$$x_3 = \log_{10}\left(\frac{|\mathcal{K}|}{|\mathcal{U}| \times |\mathcal{I}|}\right) \qquad x_6 = \sqrt{\frac{\sum_{i=1}^{|\mathcal{K}|}(r_i - \bar{r})^2}{|\mathcal{K}| - 1}}$$

Politecnico di Bari

HAND-CRAFTED SHILLING ATTACKS AGAINIST RS

Recent advances focuses on:

Goal (defence): Focuses on detecting malicious users from normal users



Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems

Wei Zhou [a], Junhao Wen [a,b,*], Qingyu Xiong [b], Min Gao [b], Jun Zeng [b]

[a] College of Computer Science, Chongqing University, China
[b] School of Software Engineering, Chongqing University, China

ARTICLE INFO

Article history:
Received 4 May 2015
Received in revised form

ABSTRACT

Due to the open nature of recommender systems, collaborative recommender systems are vulnerable to profile injection attacks, in which malicious users inject attack profiles into the rating matrix in order to bias the systems' ranking list. Recommender systems are highly vulnerable to shilling attacks, both by

HAND-CRAFTED SHILLING ATTACKS AGAINIST RS

Recent advances focuses on:

Goal (defence): Focuses on detecting malicious users from normal users

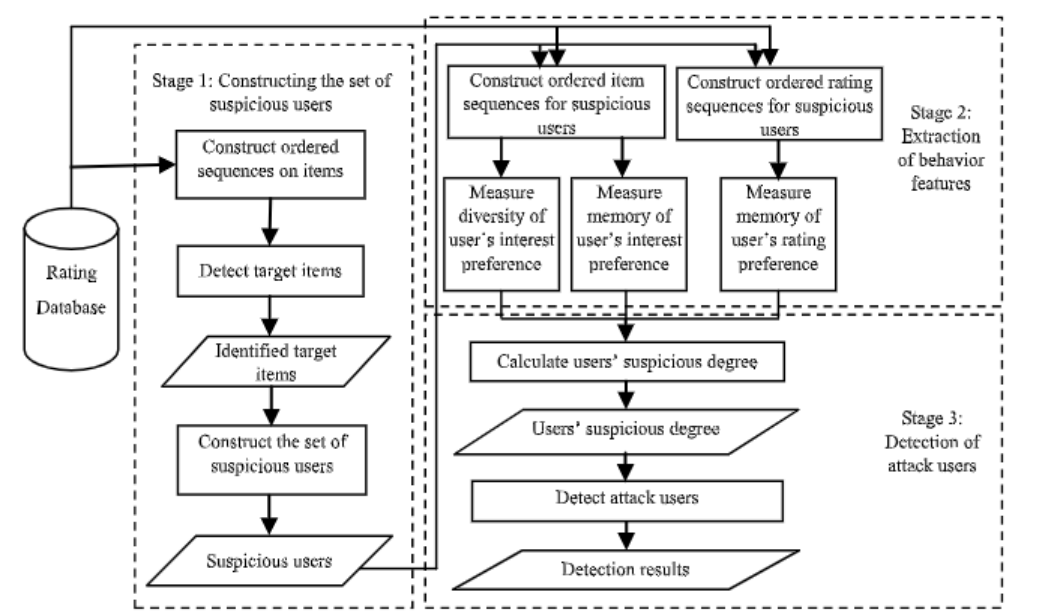


Fig. 1. Framework of DSA-AURB.

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

SVM-TIA a shilling attack detection method based on SVM and target item analysis in recommender systems

Wei Zhou [a], Junhao Wen [a,b,*], Qingyu Xiong [b], Min Gao [b], Jun Zeng [b]

[a] College of Computer Science, Chongqing University, China
[b] School of Software Engineering, Chongqing University, China

ARTICLE INFO

Article history:
Received 4 May 2015
Received in revised form

ABSTRACT

Due to the open nature of recommender systems, collaborative recommender systems are vulnerable to profile injection attacks, in which malicious users inject attack profiles into the rating matrix in order to bias the systems' ranking list. Recommender systems are highly vulnerable to shilling attacks, both by

Politecnico di Bari

DATA POISONING OPTIMIZATION ATTACKS AGAINST RS

- Main limitations of Hand-Engineered attacks:
  - **No optimization procedure** to maximize the attacker's utility
  - Empirical techniques
  - Heuristic
- In data-poising attacks against RS, a machine-learned optimization framework is built to learn an optimal user profile composition based on the attacker utility.

DATA POISONING OPTIMIZATION

Which recommendation models can be optimized?

In principle, every model could be optimized for data poisoning attacks

- In the literature, we have found:
  1. Factorization-based models
  2. Reinforcement Learning-based models
  3. Graph-based models
  4. K-NN models
  5. others

DATA POISONING OPTIMIZATION

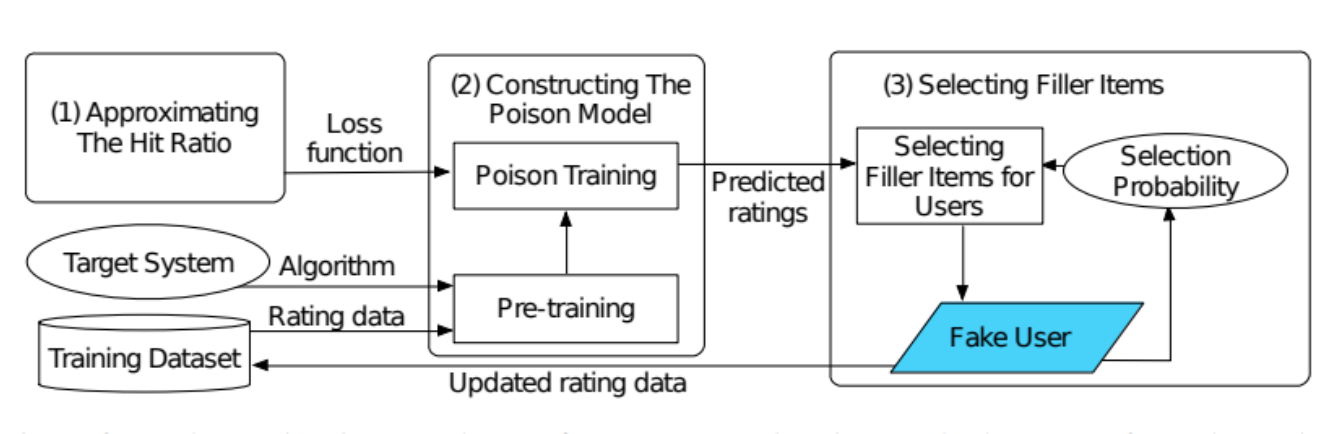
Which recommendation models can be optimized?

- Data Poisoning Optimization is DIFFERENT from the classic optimization for a recommendation utility

- What we need are:
  - the attacker utility;
  - the target recommendation model;
  - an optimization procedure for that utility.

DATA POISONING OPTIMIZATION

Goal: To manipulate a deep-RS such that the attacker-chosen target items are recommended to many users



Data Poisoning Attacks to Deep Learning Based Recommender Systems

Hai Huang[1], Jiaming Mu[1], Neil Zhenqiang Gong[2], Qi Li[1], Bin Liu[3], Mingwei Xu[1]
[1]Institute for Network Sciences and Cyberspace & Department of Computer Science and Technology, Tsinghua University
[1]Beijing National Research Center for Information Science and Technology (BNRist)
[2]Duke University
[3]Department of Management Information Systems, West Virginia University

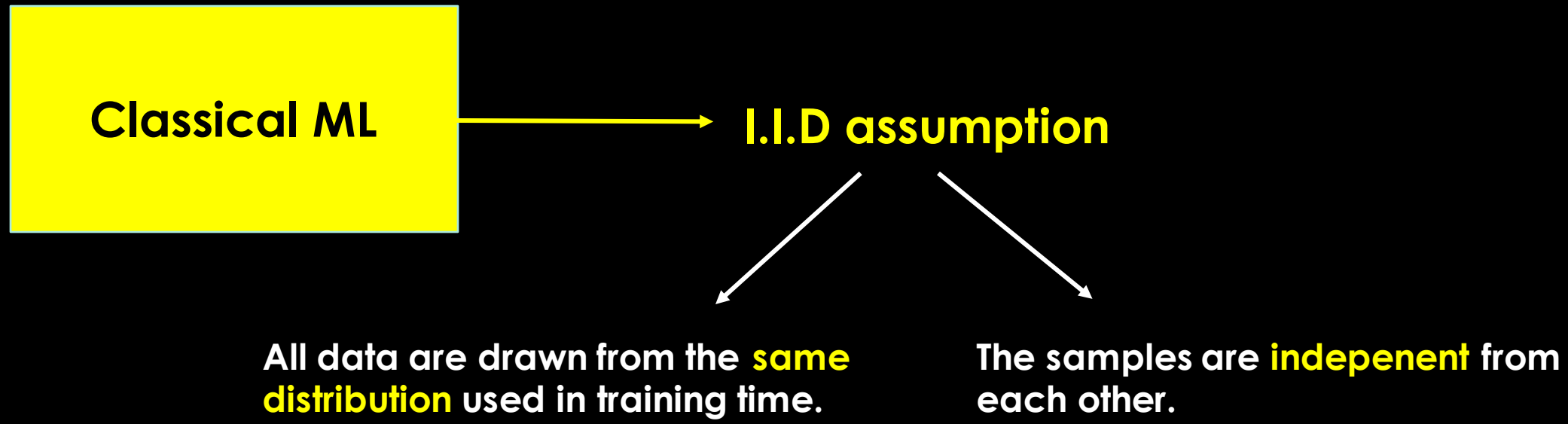
***Abstract*—Recommender systems play a crucial role in helping users to find their interested information in various web services such as Amazon, YouTube, and Google News. Various recommender systems, ranging from neighborhood-based, association-rule-based, matrix-factorization-based, to deep learning based, have been developed and deployed in industry. Among them, deep learning based recommender systems become increasingly popular due to their superior performance.**

In this work, we conduct the first systematic study on data poisoning attacks to deep learning based recommender systems. An attacker's goal is to manipulate a recommender system such

behavior (e.g., ratings or clicks) to model their preferences and make personalized recommendations for each user [37]. In a typical personalized recommender system setting, we are given a set of users, a set of items, and a log of the users' historical interactions (e.g., ratings) with the items, and the goal is to recommend each user a list of top ranked items based on user preferences learned from the historical interactions. Traditional recommender systems include *neighborhood-based* [38], *association-rule-based* [10], *matrix-factorization-based* (a.k.a *latent factor model*) [26], and *graph-based* [14]. Recently, with the rapid development of deep learning techniques, deep

ADVERSARIAL LEARNING FOR SECURITY OF RS

ADVERSARIAL LEARNING FOR SECURITY OF RS



ADVERSARIAL LEARNING FOR SECURITY OF RS

Adversarial examples taught us that simple real-world modifications are sufficient to break the i.i.d. assumtion.

"panda"
57.7% confidence

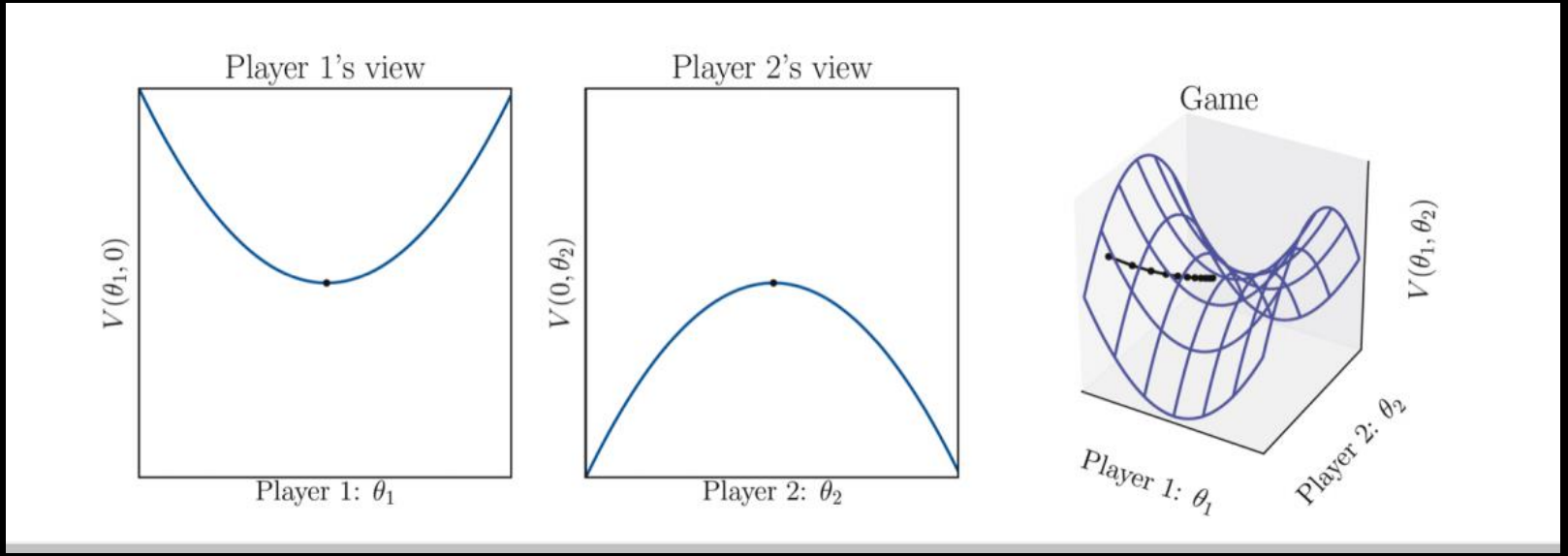
+ 0.007 ×

adversarial noise

=

"gibbon"
99.3% confidence

ADVERSARIAL TRAINING



ML model learns paramters

Attacker chooses input point

ADVERSARIAL NOISE

- **Definition of ADV. Noise**

$$\Delta_{adv} = \arg \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

- **Approximate calculation**

$$\Delta_{adv} = \epsilon \frac{\Pi}{\|\Pi\|} \quad \text{where} \quad \Pi = \frac{\partial J(X|\Omega + \Delta)}{\partial \Delta}$$

CHALLENGES

1. Unlike **images** composed of **continous features**, the input to RS are discrete (rating, (u,i,j) in BPR)

   Gradient-based apporaches to calculate adversarial noise are NOT suited!

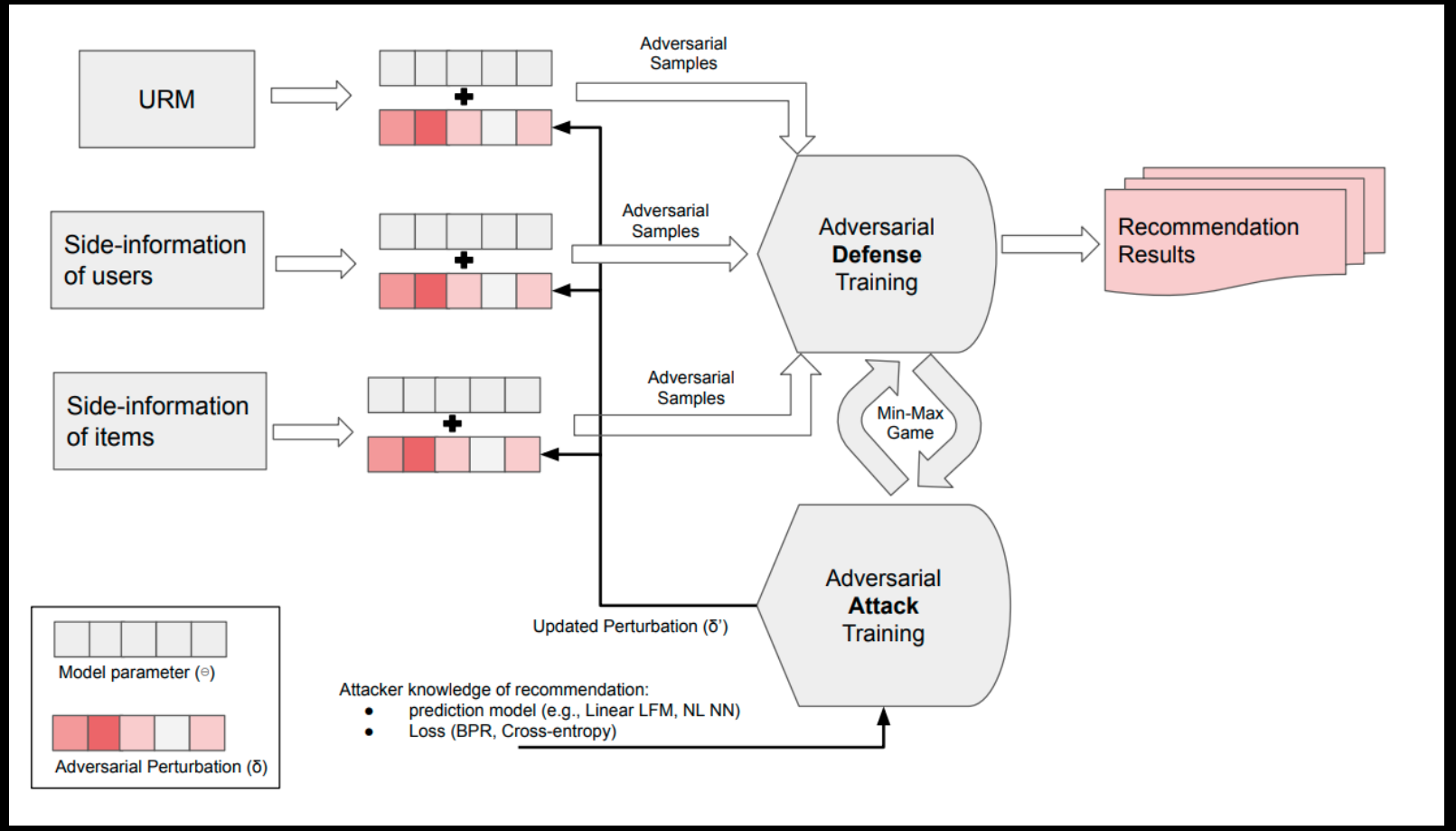
2. Adversarial examples on images aim to be **UNNOTICEABLE**.

   How can we model the notion of **'unoticable'** in (binary, discerete) RS?

ADVERSARIAL NOISE

Where can we add adversarial noise?

ADVERSARIAL NOISE



Deldjoo, Y., Noia, T. D., & Merra, F. A. (2021). A survey on adversarial recommender systems: from attack/defense strategies to generative adversarial networks. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.

Politecnico di Bari

ADVERSARIAL NOISE

Adding adversarial noise on CF model paramters:

- Adds adversarial noise to the model paramters of MF-BPR
- Compares adversarial v.s. random noise
- Applies adversarial training as a defense mechnasim

Adversarial Personalized Ranking for Recommendation*

Xiangnan He
National University of Singapore
xiangnanhe@gmail.com

Zhankui He
Fudan University
zkhe15@fudan.edu.cn

Xiaoyu Du
Chengdu University of Information Technology
duxy.me@gmail.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg

ABSTRACT

Item recommendation is a personalized ranking task. To this end, many recommender systems optimize models with pairwise ranking objectives, such as the Bayesian Personalized Ranking (BPR). Using matrix Factorization (MF) — the most widely used model in recommendation — as a demonstration, we show that optimizing it with BPR leads to a recommender model that is not robust. In particular, we find that the resultant model is highly vulnerable to adversarial perturbations on its model parameters, which implies the possibly large error in generalization.

KEYWORDS

Personalized Ranking, Pairwise Learning, Adversarial Training, Matrix Factorization, Item Recommendation

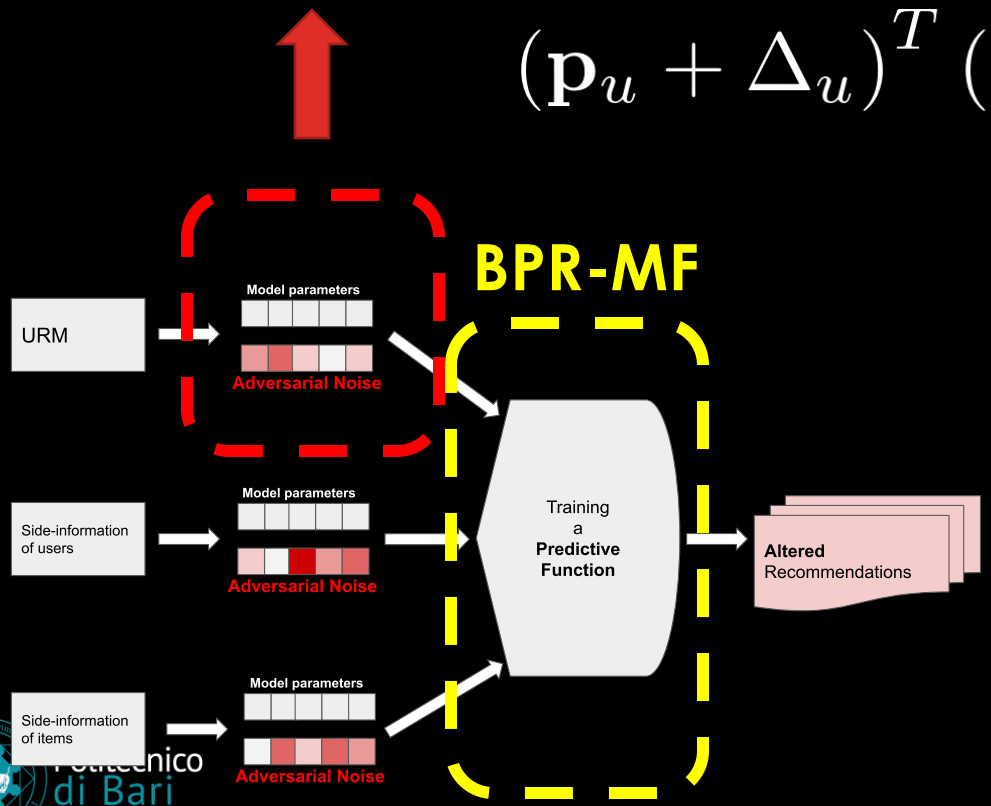
ACM Reference Format:

Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommendation. In *SIGIR '18: 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8-12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3209981

ADVERSARIAL PERSONALIZED RANKING

Adversarial Perturbation on each **embedding** vector of user and item

$$(\mathbf{p}_u + \Delta_u)^T (\mathbf{q}_i + \Delta_i)$$

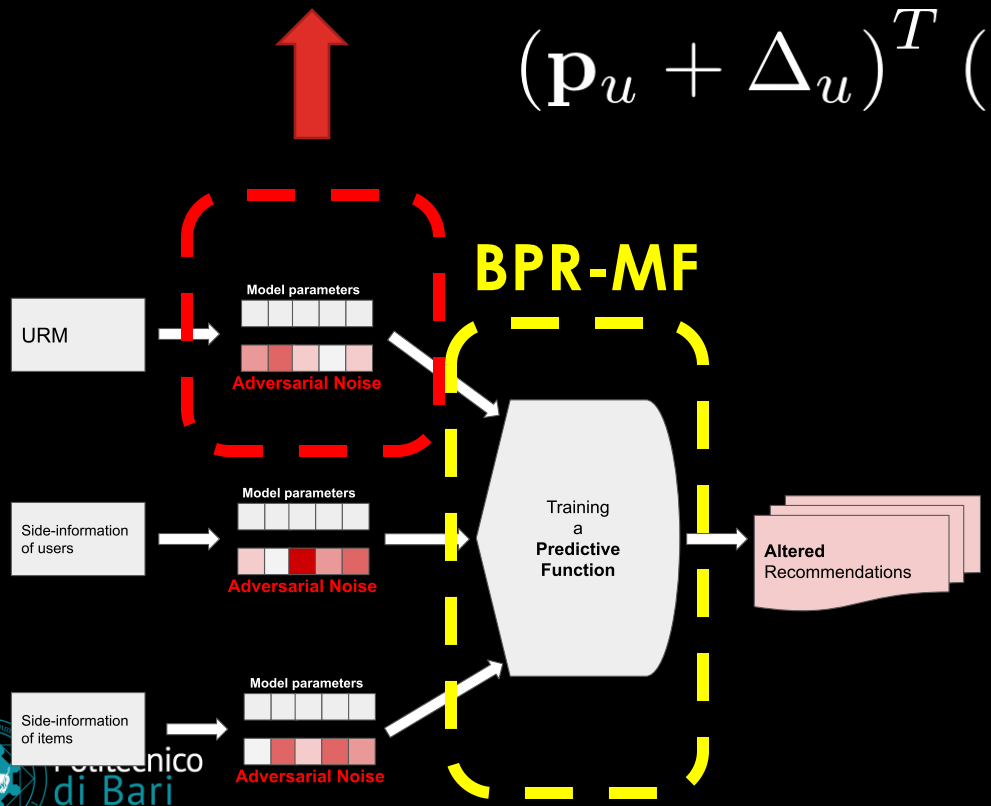


Politecnico di Bari

ADVERSARIAL PERSONALIZED RANKING

Adversarial Perturbation on each **embedding** vector of user and item

$$(\mathbf{p}_u + \Delta_u)^T (\mathbf{q}_i + \Delta_i)$$



The impact of applying adversarial perturbation

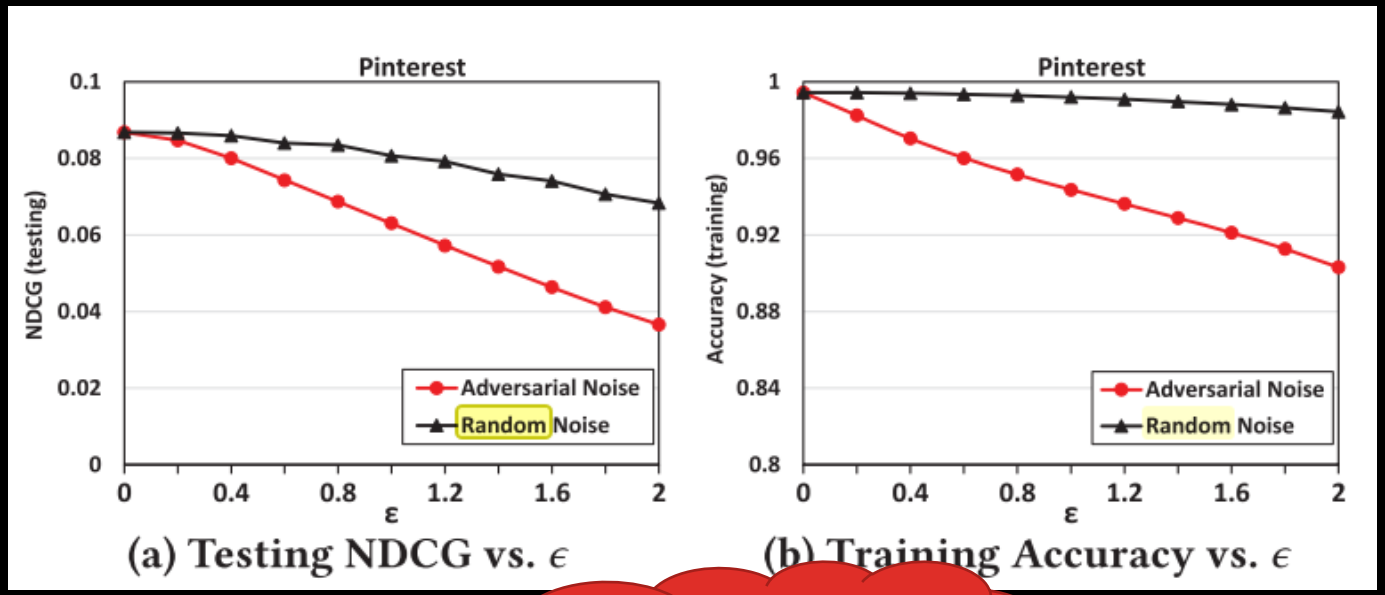
reduction of NDCG@100			
	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$
Dataset	BPR-MF	BPR-MF	BPR-MF
Yelp	-22.1%	-42.7%	-63.8%
Pinterest	-9.5%	-25.1%	-55.7%
Gowalla	-26.3%	-53.0%	-78.0%

Politecnico di Bari

ADVERSARIAL PERSONALIZED RANKING

The impact of adversarial v.s. random noise on BPR-MF:

- adversarial perturbations: NDCG decreases -21.2%
- random perturbations: NDCG decreases -1.6%



13 times difference!

Politecnico di Bari

DEFENSE AGAINST ADVERSARIAL SAMPLES

- **Goal**: Build ML models that can make robust prediction even in prescence of adversial examples.
- Main defensive pproaches:
  - (i) increasong robustness,
    - Robust optimization
      - Adversarial training (regularization)
      - Robust gradient decent
      - Certified robustness
    - Defenise destillation
  - (ii) detection

Most Popular in RecSys

Vorobeychik, Y., & Kantarcioglu, M. (2018). Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 12(3), 1-169.

ADVERSARIAL TRAINING: **DEFENSE**

- **Defender Goal?** **Minimize** the attack influence
- **Defence Method?** **Adversarial training**

$$\arg\min_{\Omega} J(X|\Omega) + \lambda J\left(X|\Omega + \Delta_{adv}\right)$$

$$where \quad \Delta_{adv} = \arg\max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

MINIMAX GAME

The training process is a **MINIMAX GAME**

$$\arg\min_{\Omega} \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega) + \lambda J(X|\Omega + \Delta)$$

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Do **Adversarial training** improve the **robustness**?

	NDCG@100					
	$\epsilon = 0.5$		$\epsilon = 1$		$\epsilon = 2$	
Dataset	BPR-MF	APR	BPR-MF	APR	BPR-MF	APR
Yelp	-22.1%	-4.7%	-42.7%	-12.5%	-63.8%	-31.0%
Pinterest	-9.5%	-2.6%	-25.1%	-7.2%	-55.7%	-23.4%
Gowalla	-26.3%	-2.9%	-53.0%	-13.2%	-78.0%	-29.2%

ADVERSARIAL PERSONALIZED RANKING

[XIANGNAN HE ET AL., SIGIR '18]

Can **Adversarial (re)training** improve the **recommendation performance**?

	Yelp, HR		Yelp, NDCG		Pinterest, HR		Pinterest, NDCG		Gowalla, HR		Gowalla, NDCG		RI
	K=50	K=100	K=50	K=100	K=50	K=100	K=50	K=100	K=50	K=100	K=50	K=100	
ItemPop	0.0405	0.0742	0.0114	0.0169	0.0294	0.0485	0.0085	0.0116	0.1183	0.1560	0.0367	0.0428	+416%
MF-BPR	0.1053	0.1721	0.0312	0.0420	0.2226	0.3403	0.0696	0.0886	0.4061	0.5072	0.1714	0.1878	+11.2%
CDAE [35]	0.1041	0.1733	0.0293	0.0405	0.2254	0.3495	0.0672	0.0873	0.4435	0.5483	0.1837	0.2007	+9.5%
IRGAN [31]	0.1119	0.1765	**0.0361***	**0.0465***	0.2254	0.3363	0.0724	0.0904	0.4157	0.518	0.1853	0.2019	+5.9%
NeuMF [17]	0.1135	0.1817	0.0335	0.0445	0.2342	0.3526	0.0734	0.0925	0.4558	0.5642	0.1962	0.2138	+2.9%
AMF	**0.1176***	**0.1885***	0.0350	**0.0465***	**0.2375***	**0.3595***	**0.0741***	**0.0938***	**0.4693***	**0.5763***	**0.2039***	**0.2212***	-

ITERATIVE ADVERSARIAL NOISE

Adding **iterative** adversarial noise on CF model paramters:

$$\Theta_0^{adv} = \Theta + \Delta_0$$

$$\Theta_1^{adv} = Clip_{\Theta,\epsilon}\left\{\Theta_0^{adv} + \alpha\frac{\Pi}{\|\Pi\|}\right\} \text{ where } \Pi = \frac{\partial\mathcal{L}(\Theta + \Delta_0)}{\partial\Delta_0}$$

MSAP: Multi-Step Adversarial Perturbations on Recommender Systems Embeddings

Vito Walter Anelli[†], Alejandro Bellogín[‡], Yashar Deldjoo[†], Tommaso Di Noia[†], Felice Antonio Merra[†*]
[†]Politecnico di Bari, Italy, firstname.lastname@poliba.it,
[‡]Universidad Autónoma de Madrid, Spain, firstname.lastname@uam.es

Abstract

Recommender systems (RSs) have attained exceptional performance in learning users' preferences and finding the most suitable products. Recent advances in adversarial machine learning (AML) in computer vision have raised interests in recommenders' security. It has been demonstrated that widely adopted model-based recommenders, e.g., BPR-MF, are not robust to adversarial perturbations added on the learned parameters, e.g., users' embeddings, which can cause drastic reduction of recommendation accuracy. However, the state-of-the-art adversarial method, named fast gradient sign method (FGSM), builds the perturbation with a single-step procedure.

In this work, we extend the FGSM method proposing multi-step adversarial perturbation (MSAP) procedures to study the recommenders' robustness under powerful methods. Letting fixed the perturbation magnitude, we illustrate that MSAP is much more harmful than FGSM in corrupting the recommendation performance of BPR-MF. Then, we assess the MSAP efficacy on a robustified version of BPR-MF, i.e., AMF. Finally, we analyze the variations of fairness measurements on each perturbed recommender. Code and data are available at https://github.com/sisinflab/MSAP.

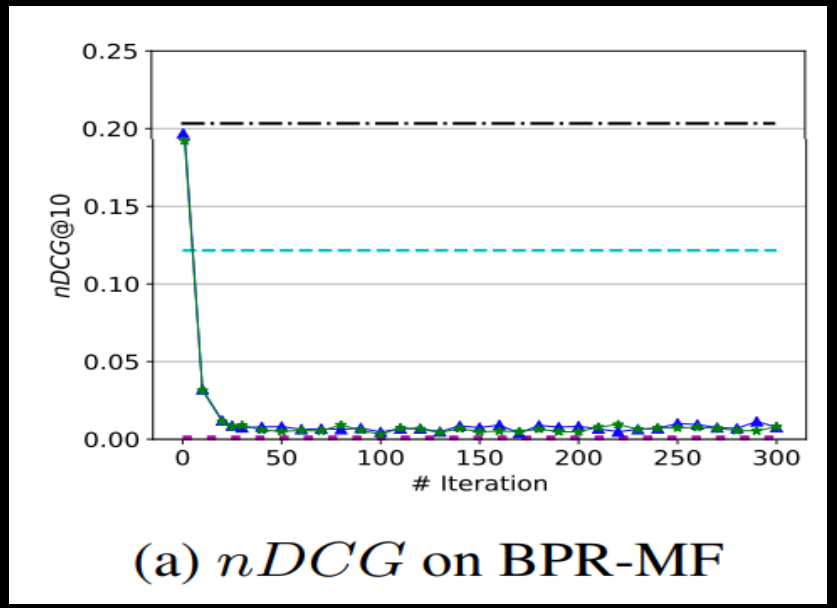
2018; Carlini and Wagner 2017), as well as defenses (Goodfellow, Shlens, and Szegedy 2015), have been studied in CV domain with the goal to make reliable ML models.

(He et al. 2018) proposed the pioneering work of AML for RSs. The authors clarified that attacks and defenses should be treated differently in the CV and RS tasks since image data are continuous-valued matrices, while recommender data are discrete interactions (0/1 feedback). For this reason, they tested adversarial perturbations on the model parameters, e.g., the embedding matrices of matrix-factorization (MF) models. They discovered that the fast gradient sign method (FGSM) (Goodfellow, Shlens, and Szegedy 2015), a *single-step adversarial perturbation* procedure, leads to five times larger deterioration of recommendation accuracy than the one caused by random variation. This finding shows the weaknesses of model-based recommenders in learning embeddings that will cause drastic performance degradation when subjected to small changes. For instance, this change can be caused when new users, or items, are added to the system. Furthermore, they successfully applied an *adversarial training* procedure (Goodfellow, Shlens, and

Politecnico di Bari

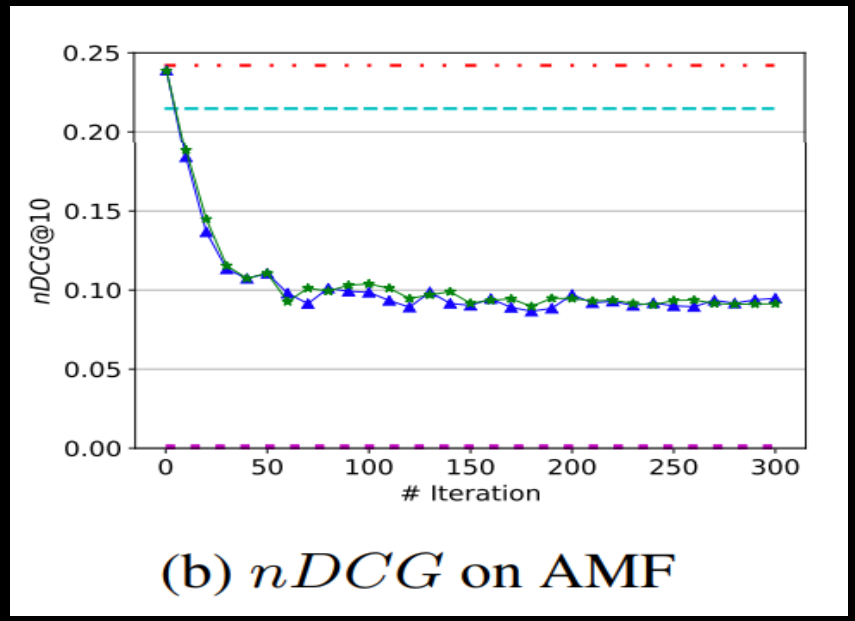
ITERATIVE ADVERSARIAL NOISE

- Iterative Perturbation can make the recommendation model worse than a random model



(a) $nDCG$ on BPR-MF

- The APR defense strategy limitates but does not protect from MSAP



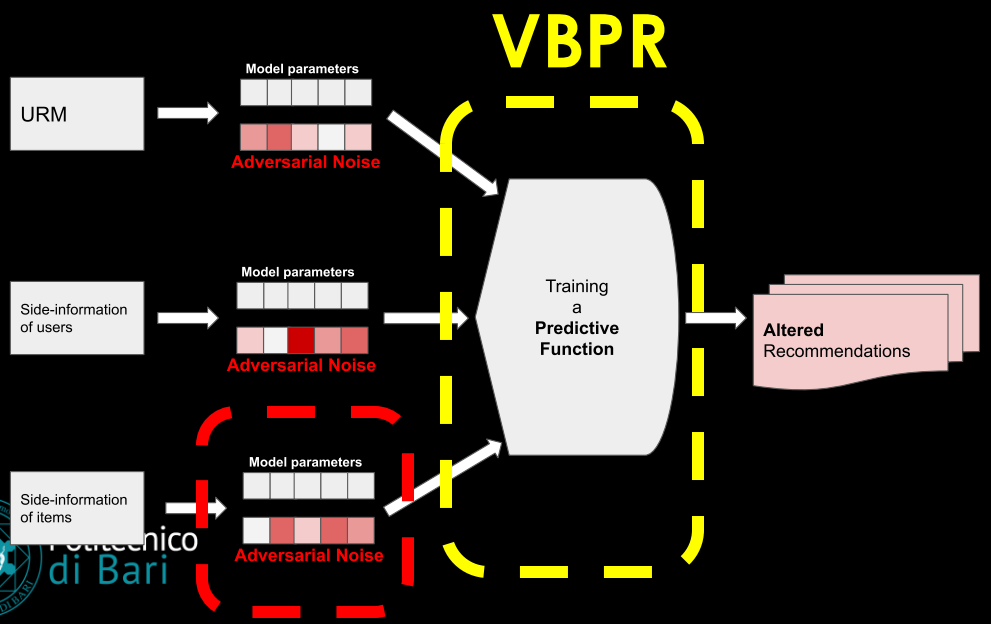
(b) $nDCG$ on AMF

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Adversarial Perturbation on each **CONTENT** embedding.

$$\mathbf{p}_u^T(\mathbf{q}_i + \boldsymbol{E} \cdot (\boldsymbol{c}_i + \boldsymbol{\Delta}_i))$$



The impact of applying adversarial perturbation

reduction of NDCG@10			
	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.2$
Dataset	VBPR	VBPR	VBPR
Amazon	-8.7%	-30.4%	-67.7%
Pinterest	-4.2%	-11.9%	-31.8%

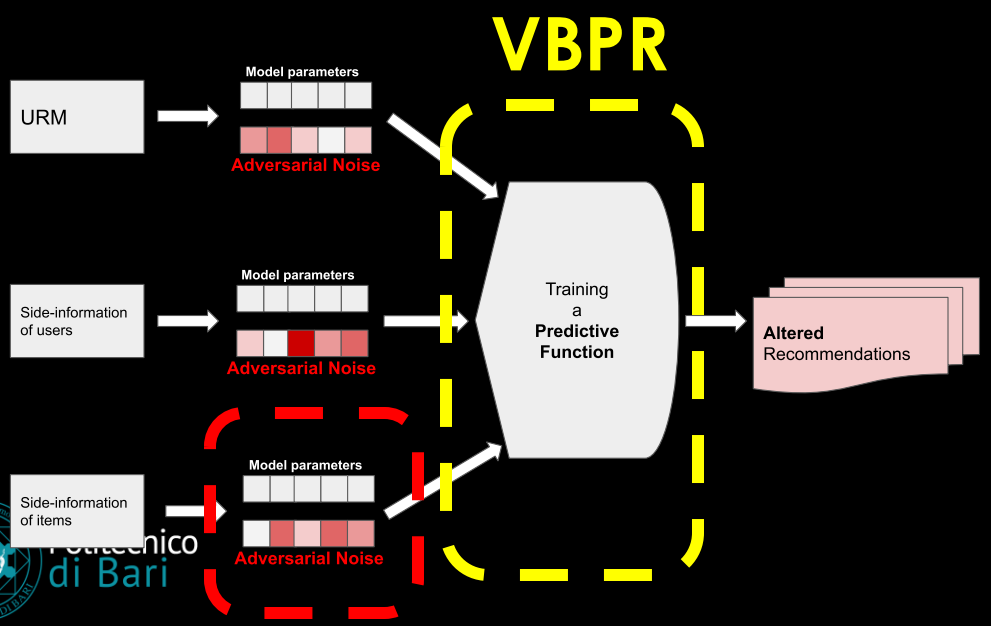
Politecnico di Bari

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Adversarial Perturbation on each **CONTENT** embedding.

$$\mathbf{p}_u^T(\mathbf{q}_i + \boldsymbol{E} \cdot (\boldsymbol{c}_i + \boldsymbol{\Delta}_i))$$



Apply Adversarial Training

Adversarial Regularization

$$\arg\min_{\Omega} \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega) + \lambda J(X|\Omega + \Delta)$$

where $J = J_{VBPR}$

ADVERSARIAL MULTIMEDIA RECOMMENDATION

[XIANGNAN HE ET AL., TKDE'19]

Do **Adversarial (re)training** improve the **robustness**?

	$\epsilon = 0.05$		$\epsilon = 0.1$		$\epsilon = 0.2$	
Dataset	**VBPR**	**AdvReg**	**VBPR**	**AdvReg**	**VBPR**	**AdvReg**
Amazon	-8.7%	-1.4%	-30.4%	-5.3%	-67.7%	-20.2%
Pinterest	-4.2%	-2.6%	-11.9%	-6.2%	-31.8%	-18.4%

Adversarial Perturbation on Content Data:

A focus on Multimedia Recommender Models

Politecnico
di Bari

ATTACK TIMIMIG

TRAINING TIME (Poisoning)

- Image samples are perturbed and injected in the VRSs before the training.

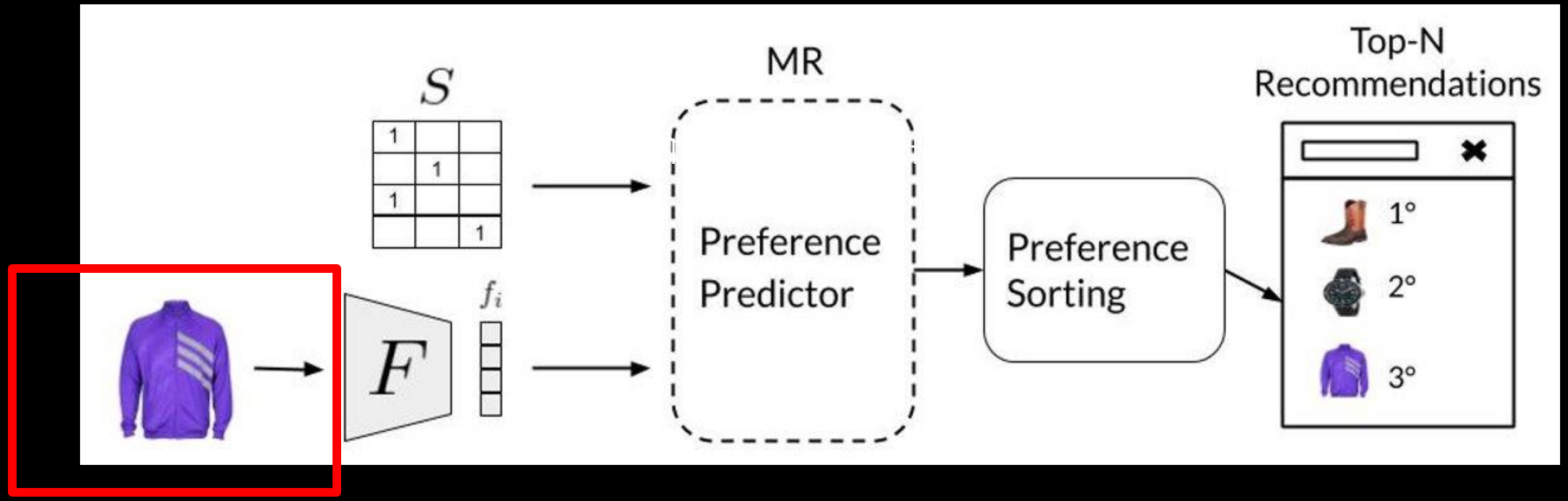
- **WORKS**
  - TAaMR **[Di Noia et al, 2020]**
  - VAR **[Anelli et al, 2020]**

TESTING TIME (Evasion)

- Images are perturbed at **inference** time

- **WORKS**
  - BlackBox-Model **[Cohen et al, 2021]**
  - Adv. Item Promotion **[Zhouran et al, 2021]**

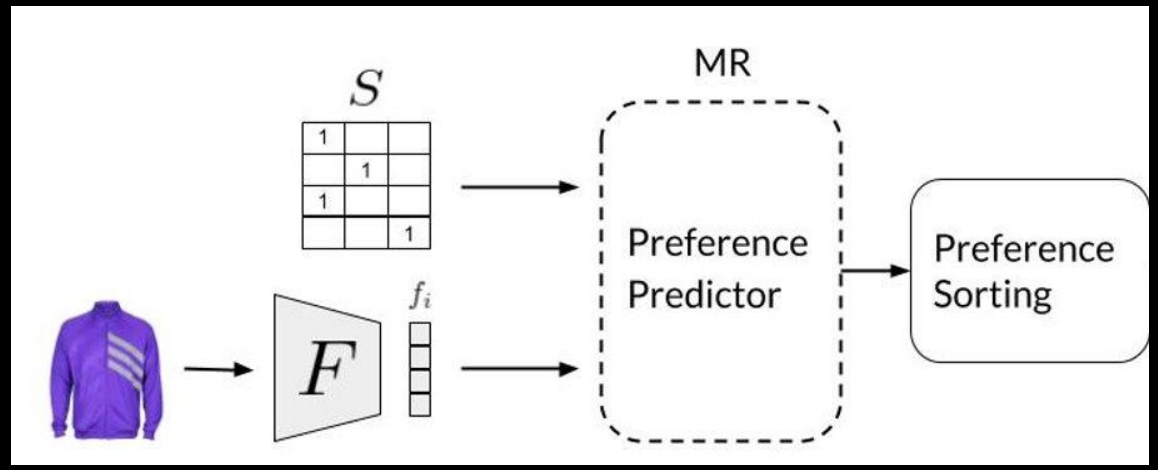
WHAT IS THE FAILURE POINT?



THE ADVERSARY CAN PERTURB THE PRODUCT IMAGES

Politecnico di Bari

ADVERSARIAL ATTACKS AGAINST VISUAL-AWARE RS

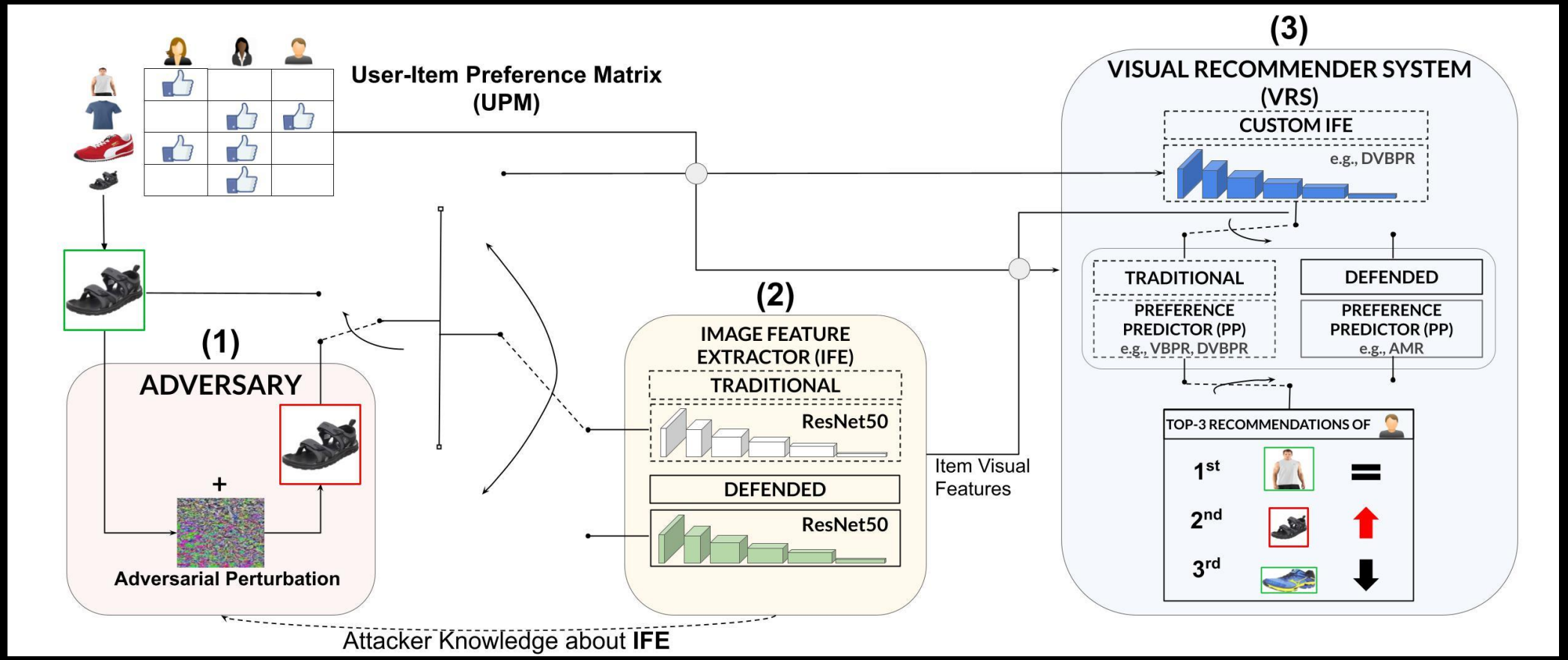


VRS		Image Feature Extractor			
		Extraction Layer		Training	
Model	*Reference*	*FC*	*FM*	*Pretrained*	*End-to-End*
FM	Rendle [39]	✓		✓	
VBPR	He and McAuley [21]	✓		✓	
AMR	Tang et al. [45]	✓		✓	
ACF	Chen et al. [9]		✓	✓	
DVBPR	Kang et al. [24]	✓			✓

[Tommaso Di Noia, Daniele Malitesta, Felice Antonio Merra: TAaMR: Targeted Adversarial Attack against Multimedia Recommender Systems. DSN Workshops 2020]

Politecnico di Bari

TRAINING TIME ATTACK
VISUAL ADVERSARIAL RECOMMENDATION FRAMEWORK



[Anelli, Deldjoo, Di Noia, Malitesta and Merra, *A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images*, Under Review]

TRAINING TIME ATTACK
VISUAL ADVERSARIAL RECOMMENDATION FRAMEWORK

- **Adversarial Attacks**
  - FGSM
  - PGD
  - Carlini&Wagner
- **Adversarial Defense**
  - Adversarial Regularization of the Rec. Engine (APR)
  - Adversarial Training of the IFE
  - Free Adversarial Training of the IFE
  - Personalized Training (**DVBPR**)

WHITE BOX wrt the IFE

BLACK BOX wrt the Recommender

Politecnico di Bari

[Anelli, Deldjoo, Di Noia, Malitesta and Merra, *A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images,* Under Review]

TRAINING TIME ATTACK
VISUAL ADVERSARIAL RECOMMENDATION FRAMEWORK

Data	VRS	Att.	Image Feature Extractor					
			Traditional		Adv. Train.		Free Adv. Train.	
			CHR	*CnDCG*	*CHR*	*CnDCG*	*CHR*	*CnDCG*
Amazon Men	FM	*Base*	*0.4960*	*0.0246*	*0.4082*	*0.0204*	*0.4048*	*0.0202*
		FGSM	**0.5309** *	**0.0266***	0.3886	0.0198*	0.3821*	0.0194*
		PGD	0.5293*	**0.0266***	0.3795*	0.0193*	0.3811*	0.0193*
		C&W	0.5258*	0.0263*	0.3837*	0.0194*	0.3871*	0.0194*
	VBPR	*Base*	*0.6531*	*0.0293*	*0.3074*	*0.0141*	*0.3775*	*0.0159*
		FGSM	0.5824*	0.0299	0.6164*	0.0323*	0.5860*	0.0283*
		PGD	**1.1480**	**0.0538***	0.6410*	0.0324*	0.5918*	0.0286*
		C&W	0.6132*	0.0290	**0.6880***	**0.0336***	**0.6642***	**0.0348***
	AMR	*Base*	*0.3944*	*0.0196*	*0.5037*	*0.0232*	*0.1076*	*0.0038*
		FGSM	0.3347*	0.0150*	0.4426*	0.0235	0.4178*	0.0187*
		PGD	**0.8365**	**0.0418***	0.4519*	**0.0242**	0.4263*	0.0193*
		C&W	0.3678	0.0170*	0.4371*	0.0230	**0.4451***	**0.0202***
	ACF	*Base*	*0.5574*	*0.0278*	*0.3560*	*0.0176*	*0.3565*	*0.0176*
		FGSM	**0.5692***	**0.0282***	**0.3773***	**0.0185***	0.3517	0.0172*
		PGD	0.5610	0.0280	0.3731*	0.0183*	0.3521	0.0172*
		C&W	0.5628	0.0279	0.3690*	0.0181*	0.3471*	0.0169*
	DVBPR	*Base*	*0.6945*	*0.0359*	–	–	–	–
		FGSM	0.6579*	0.0329*	–	–	–	–
		PGD	0.5549*	0.0281*	–	–	–	–
		C&W	0.6414*	0.0306*	–	–	–	–

Politecnico di Bari

[Anelli, Deldjoo, Di Noia, Malitesta and Merra, *A Study of Defensive Methods to Protect Visual Recommendation Against Adversarial Manipulation of Images*, Under Review]

TRAINING TIME ATTACK HUMAN IMPERCEPTIBILITY

a. Clean
Rec. Position: 68th

b. Attack + **T**
Rec. Position: 10th
LPIPS: 0.5484

c. Attack + **AT**
Rec. Position: 27th
LPIPS: 0.5347

d. Attack + **FAT**
Rec. Position: 40th
LPIPS: 0.3447

Politecnico di Bari

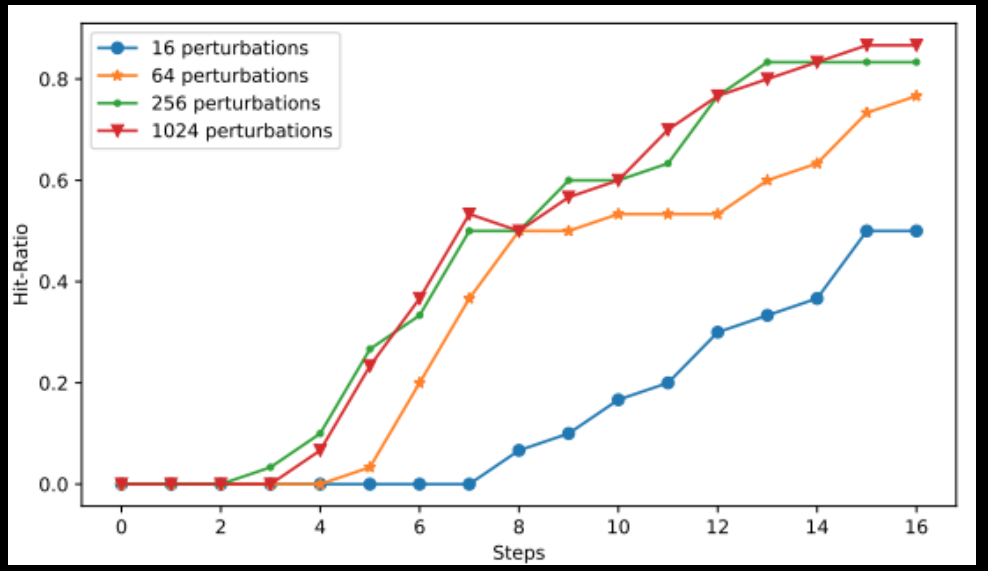
[Vito Walter Anelli, Tommaso Di Noia, Daniele Malitesta, Felice Antonio Merra:Assessing Perceptual and Recommendation Mutation of Adversarially-Poisoned Visual Recommenders. WDCS@NeurIPS2020: 49-56]

TEST TIME ATTACK
BLACK-BOX ATTACK MODEL FOR VRS

- **Black-box** attack on
  - Preference Scores
  - Ranking list

- **Adversarial Goal**?
  - Specific User (Grey-box settings)
  - Segment Users with Similar Taste
  - General population

- Approach?
  - Solve an optimization problem while with a pixel-by-pixel perturbation

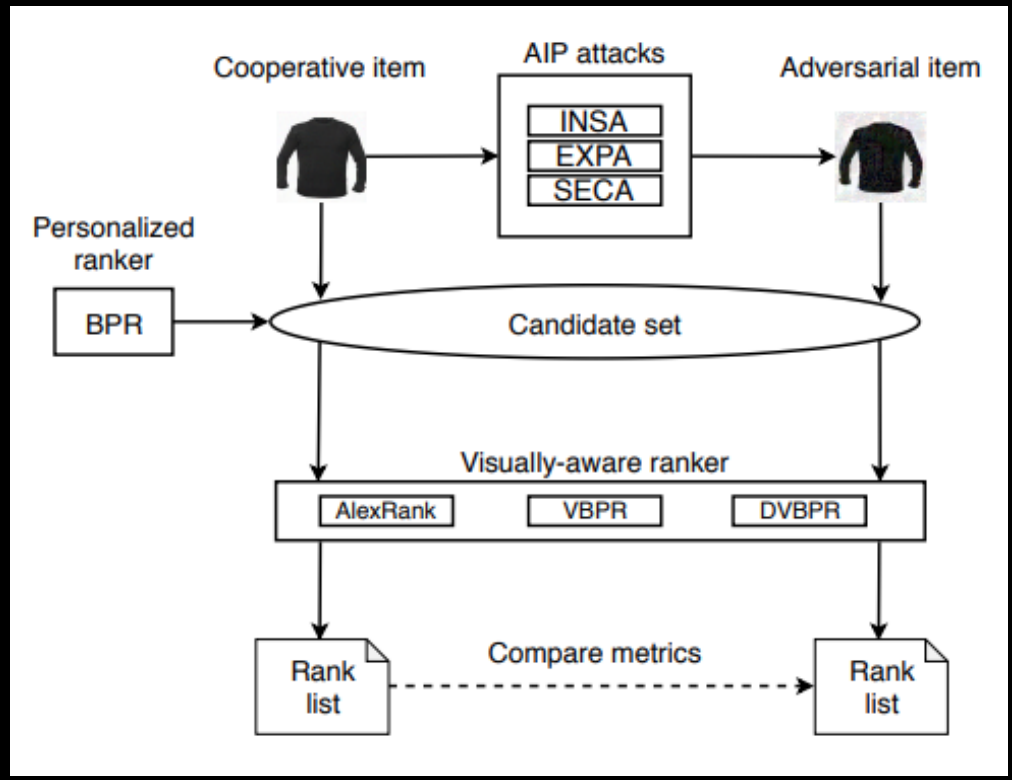


Politecnico di Bari

[Rami Cohen, Oren Sar Shalom, Dietmar Jannach, Amihood Amir: A Black-Box Attack Model for Visually-Aware Recommender Systems. WSDM 2021]

TEST TIME ATTACK
ADVERSARIAL ITEM PROMOTION

- **Three-attack types**
  - **White**
  - **Grey**
  - **Black**
- **Adversarial Goal?**
  - Promote a specific item by perturbing its image
- **Approach?**
  - Directly optimize the perturbation with respect to a BPR-based loss function



Politecnico di Bari

[Zhuoran Liu, Martha Larson:Adversarial Item Promotion: Vulnerabilities at the Core of Top-N Recommenders that Use Images to Address Cold Start WWW 2021]

OTHER APPLICATIONS

[Yuan, F. et al., 2019]

- attack/defense on **Deep Neural Model/ Auto-Encoder**

[Chen and Li, 2019]

- attack/defense on Factorization Machine

[Li R. et al., 2019]

- adv. regularization to improve **Sequential recommendations**

[Park et al., 2019]

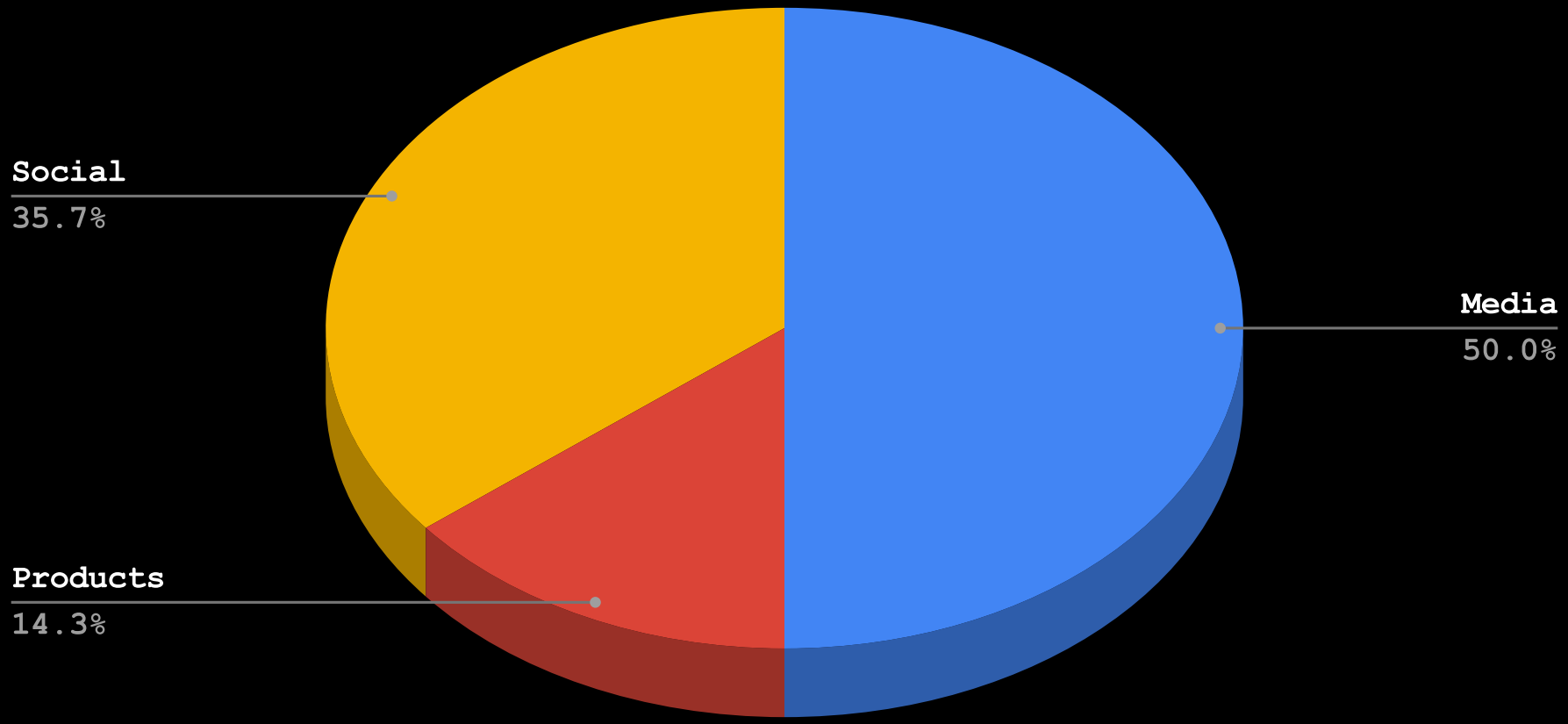
- perturb **user-continuous input**

[Du et al., 2019]

- attack with **C&W**, defense with **Defensive Distillation**

For further study check our Survey!

DOMAINS



Politecnico di Bari

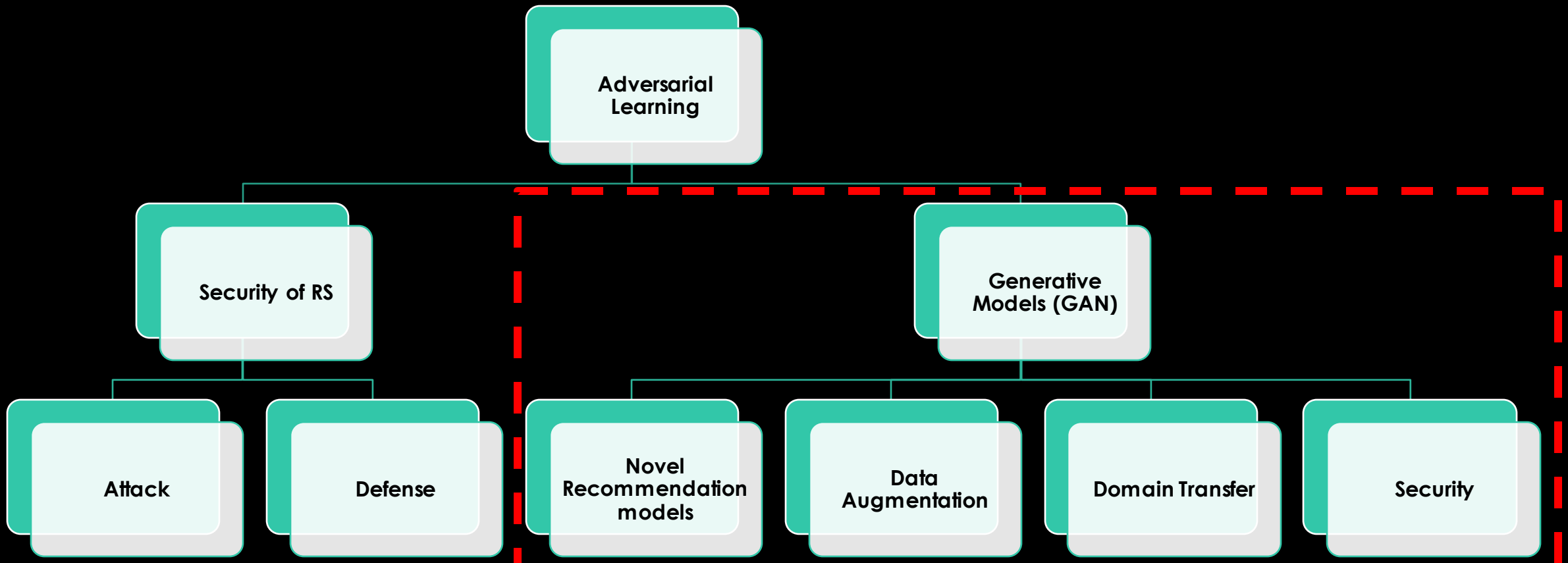
OPEN DIRECTIONS

- Other **attacks strategies**
  - Use state-of-the-art adv. Attack strategies
  - Implement perturbation direct on the input:
    - user-rating profile
    - Imitation of implicit feedback
    - images, audio, videos
- Other **defence approaches**
- Verify and Extend the **AVD-RF** on other recomenders
- Other **domains**

ADVERSARIAL LEARNING FOR GAN-BASED RECOMMENDATION

Politecnico
di Bari

ADVERSARIAL LEARNING FOR RS



PIONEERING WORKS

- **IRGAN** [Wang J. et al., *SIGIR'17*]
  - Solution for CF based on **MF**
  - Combine **generative** and **discriminative** **IR** schools of thinking under a GAN-framework
- **GraphGAN** [Wang H. et al., *AAAI'18*]
  - Solution for CF based on **graph-representation learning**
  - Combine **generative** and **discriminative** **graph-representation learning** models under a GAN-framework

IRGAN: A MINIMAX GAME FOR UNIFYING GENERATIVE AND DISCRIMINATIVE INFORMATION RETRIEVAL MODELS

[WANG J. ET AL., SIGIR'17]

Generative model

- **Assumption:** Exists a stochastic generative process between i and u

$$u \longrightarrow i$$

- **Model function:** $p_\theta(i|u,r)$ tries to generate **relevant items**, from the candidate pool for the given **user**

IRGAN: A MINIMAX GAME FOR UNIFYING GENERATIVE AND DISCRIMINATIVE INFORMATION RETRIEVAL MODELS

[WANG J. ET AL., SIGIR'17]

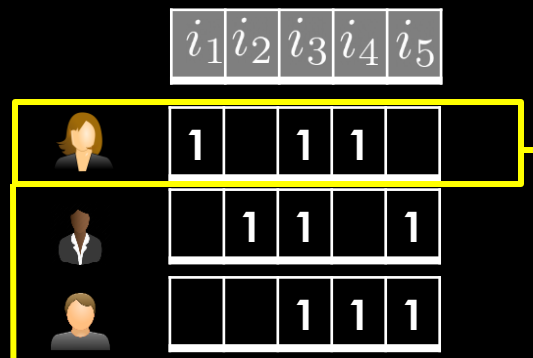
Discriminative model

- **Assumption:** The rating is predicted as a label given i and u

$$u + i \rightarrow r$$

- **Model function:** $p_\phi(u, i)$ tries to discriminate well-matched user-item pairs from ill-matched ones

GAN-RF



u

Real item

i_1

$\arg\max_{\phi} J(\theta, \phi)$

D

Distinguish the true and generated relevant item

G is user-conditioned

u

G

$p_\theta(i|u, r)$

Generated item

i_7

$\arg\min_{\theta} J(\theta, \phi)$

Politecnico di Bari

ADV-RF: **ALGORITHM**

MINIMAX

Input: Training data **X**
Initialize G and D
Pretrain G and D
While *stopping-criteria* do:
For g-steps do:
Generate K relevant items for each user u
Update G parameters
end for
For d-steps do:
Combine Real relevant items with Generated Relevant items.
Update D parameters
end for

Politecnico di Bari

GAN-RecSys Applications

CATEGORIZATION

1. **Collaborative** Recommendation
    1. Pure Collaborative Filtering
    2. Graph-based Recommendation
    3. Hybrid
2. **Contextual** Recommendation
    1. Time-aware
    2. Geographical
3. **Cross-domain** Recommendation
4. **Complementary** Recommendation
5. Other applications

COLLABORATIVE RECOMMENDATION - **PURE**:

CFGAN

[DONG-KYU CHAE, CIKM '18]

PROBLEM

1. Generator's **performance degredation** as training progresses
2. Difficulties in capturing **users's preference** on **0/1-sparse-vector**

PROPOSAL

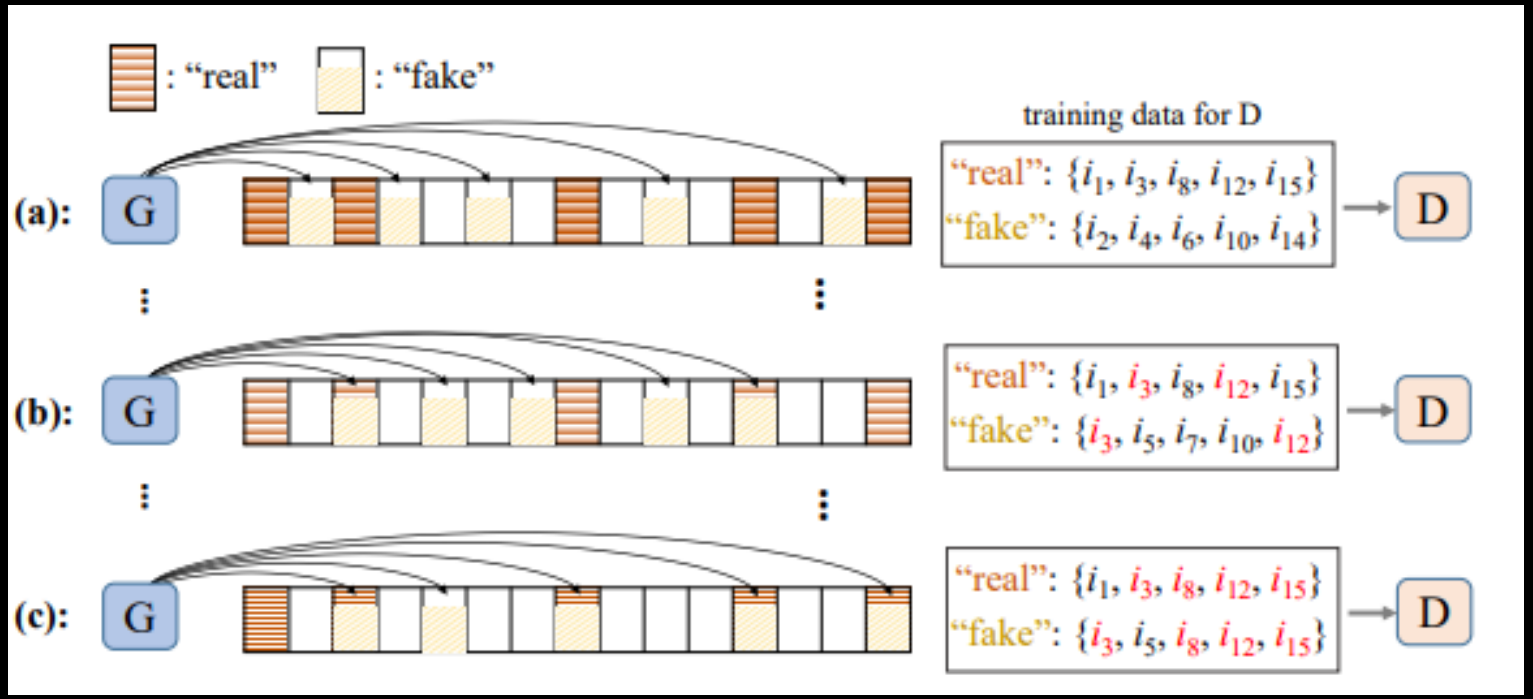
1. **Vector-wise adversarial training** where G generates ***real-valued vectors***
2. Novel **negative-sampling methods to capture the** user's relative preferences

COLLABORATIVE RECOMMENDATION - **PURE:**

CFGAN

[DONG-KYU CHAE, CIKM '18]

Generator's performance degredation because it samples 'real' items with **contradicting labels**

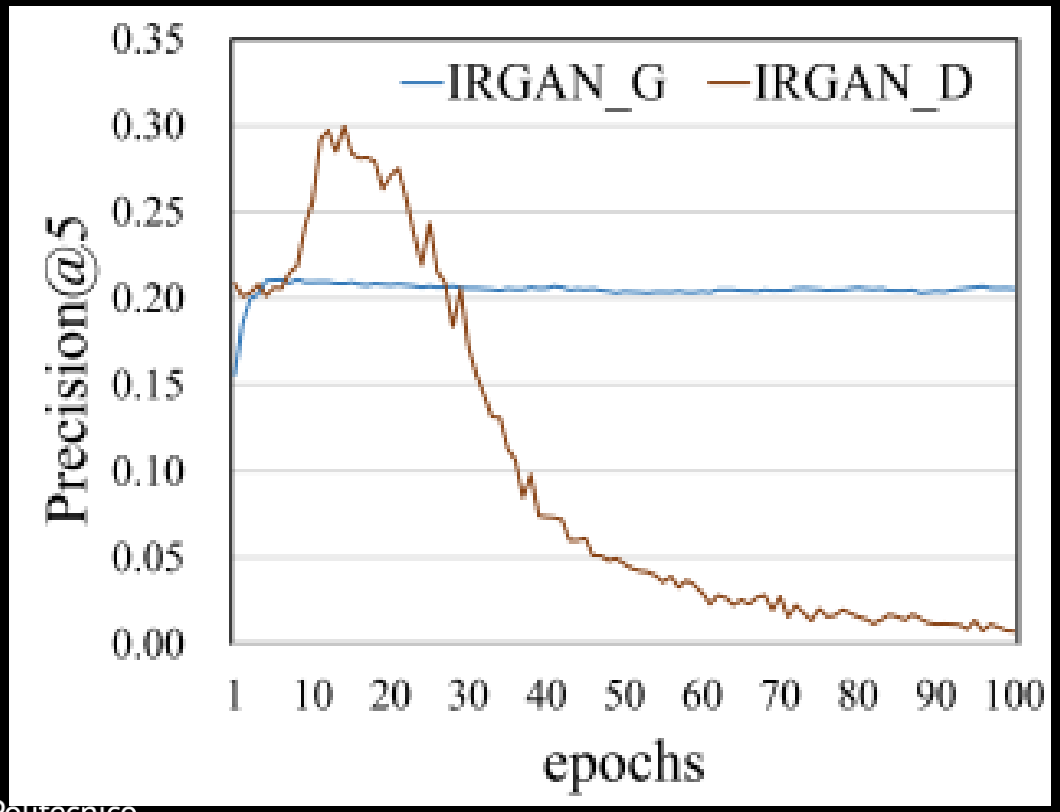


COLLABORATIVE RECOMMENDATION - **PURE:**

CFGAN

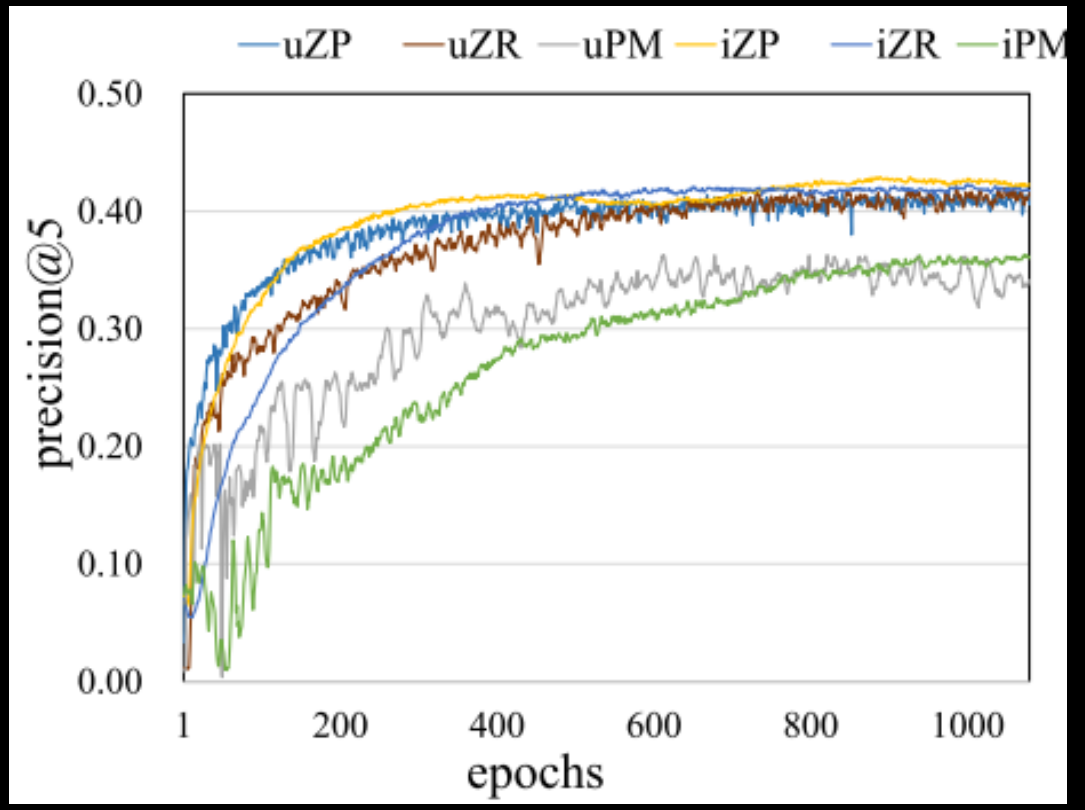
[DONG-KYU CHAE, CIKM '18]

Single-item IRGAN implementation



Results on Movielens 100K

Vector-wise CF-GAN implementation



Results on Movielens 100K

Politecnico di Bari

COLLABORATIVE RECOMMENDATION - **PURE:**

CFGAN

[DONG-KYU CHAE, CIKM '18]

datasets	Movielens 100K								Movielens 1M							
metrics	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20	P@5	P@20	R@5	R@20	G@5	G@20	M@5	M@20
ItemPop	.181	.138	.102	.251	.163	.195	.254	.292	.157	.121	.076	.197	.154	.181	.252	.297
BPR	.348	.236	.116	.287	.370	.380	.556	.574	.341	.252	.077	.208	.349	.362	.537	.556
FISM	.426	.285	.140	.353	.462	.429	.674	.685	.420	.302	.107	.270	.443	.399	.637	.651
CDAE	.433	.287	.144	.353	.465	.425	.664	.674	.419	.307	.108	.272	.439	.401	.629	.644
GraphGAN	.212	.151	.102	.260	.183	.249	.282	.312	.178	.194	.070	.179	.205	.184	.281	.316
IRGAN	.312	.221	.107	.275	.342	.368	.536	.523	.263	.214	.072	.166	.264	.246	.301	.338
Ours	**.444**	**.294**	**.152**	**.360**	**.476**	**.433**	**.681**	**.693**	**.432**	**.309**	**.108**	**.272**	**.455**	**.406**	**.647**	**.660**

G@N = normalized Discounted Cumulative Gain

M@N = Mean Reciprocal Rank

+ 2.8% against FISM [Kabbur et al., KDD'13]

SECURITY APPLICATION

ADVERSARIAL ATTACKS ON AN OBLIVIOUS RECOMMENDER

[CHRISTAKOPOULOU AND BANERJEE, RECSYS'19]

PROBLEM

1. user-rating matrix **poisoning attack** in an optimized way
2. **Unnoticeability** of fake profiles

PROPOSAL

1. **Zero-order optimization techniques** to overcome the challenge that the adversary does not have access to the recommender's gradient
2. **Use GANs for generating realistic fake-user samples**

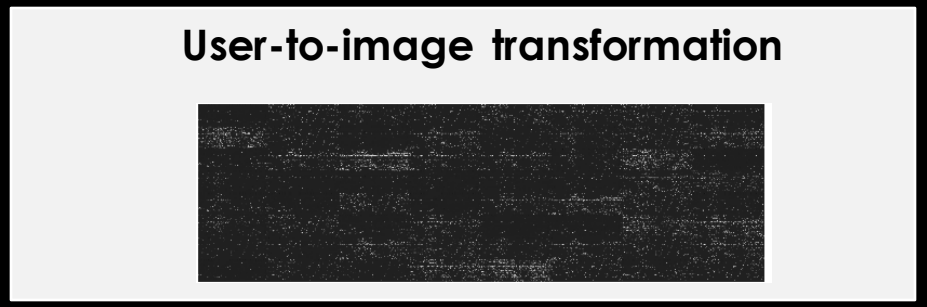
SECURITY APPLICATION

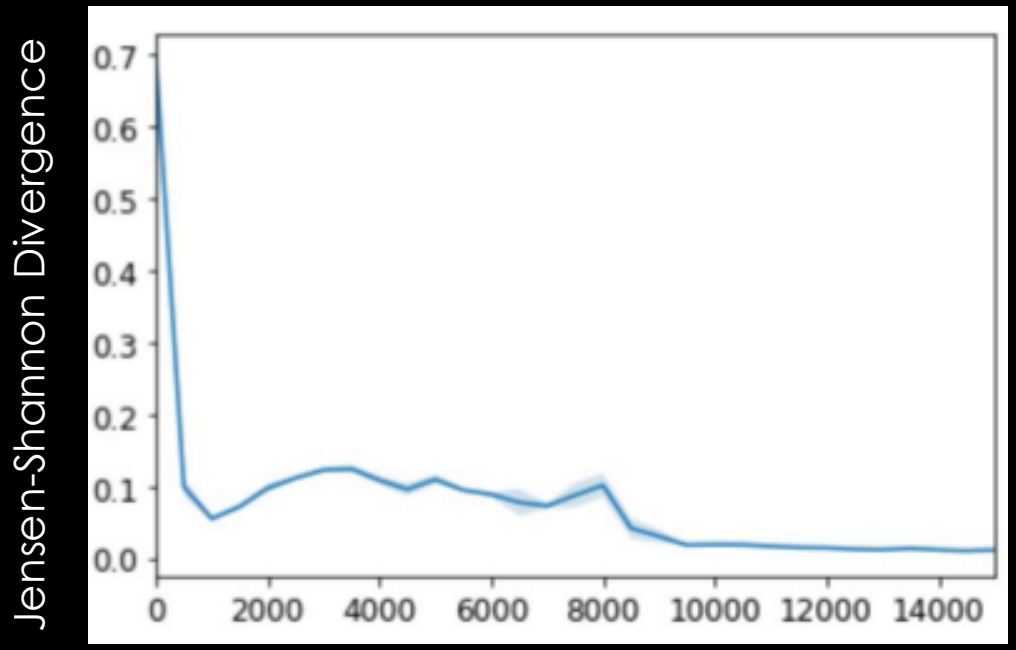
ADVERSARIAL ATTACKS ON AN OBLIVIOUS RECOMMENDER

[CHRISTAKOPOULOU AND BANERJEE, RECSYS'19]

METHOD:

Transform each user profile in an image to work with standard GAN for computer vision.





"GANs can produce fake user samples whose distribution is close to the real user distribution."

[CHRISTAKOPOULOU AND BANERJEE, RECSYS'19]

Politecnico di Bari

Table 5. A Schematic Representation of GAN-based Approaches to Recommendation

Name	Year	Generator (G)							Discriminator (D)						Training	
		LFM	MLP	CNN	AE	VAE	LSTM	GRU	LFM	MLP	CNN	AE	LSTM	GRU	point	pair
Collaborative Rec.																
IRGAN [150]	2017	✓							✓						✓	✓
CFGAN [28]	2018		✓							✓					✓	
Chae et al. [29]	2018				✓				✓							✓
AVAE [174]	2018					✓				✓					✓	
GAN-VAE-CF [84]	2018				✓	✓				✓						✓
CAAE [30]	2019				✓				✓							✓
CGAN [141]	2019					✓				✓					✓	
CALF [42]	2019			✓							✓					✓
PD-GAN [160]	2019	✓							✓							✓
LambdaGAN [154]	2019	✓							✓							✓
VAEGAN [169]	2019					✓				✓					✓	
APL [136]	2019	✓							✓							✓
RsyGAN [167]	2019				✓					✓					✓	
GAN-PW/LSTM [35]	2019						✓			✓					✓	
CoFiGAN [96]	2020	✓							✓						✓	✓
GCF [173]	2020	✓		✓					✓	✓					✓	✓
Graph-based Collaborative Rec.																
GraphGAN [148]	2018	✓							✓						✓	
GAN-HBNR [20]	2018				✓							✓			✓	
VCGAN [177]	2018					✓			✓						✓	
UPGAN [65]	2020		✓							✓					✓	
Hybrid Collaborative Rec.																
VAE-AR [88]	2017					✓				✓					✓	
RGD-TR [93]	2018		✓							✓					✓	
aae-RS [166]	2018				✓					✓					✓	
SDNet [37]	2019	✓							✓						✓	
ATR [121]	2019							✓			✓				✓	
AugCF [153]	2019		✓						✓						✓	
RSGAN [168]	2019				✓				✓							✓
RRGAN [35]	2019	✓								✓					✓	
UGAN [155]	2019	✓							✓						✓	
LARA [135]	2020		✓							✓					✓	
CGAN [39]	2020				✓					✓					✓	
Context-aware Rec.																
Temporal-aware																
RecGAN [16]	2018							✓						✓	✓	
NMRN-GAN [152]	2018		✓										✓			✓
AAE [143]	2018				✓					✓					✓	
PLASTIC [180]	2018	✓					✓		✓				✓			✓
LSIC [179]	2019	✓					✓		✓				✓			✓
GAN-CDQN [36]	2019						✓			✓					✓	
AOS4Rec [178]	2020		✓					✓		✓				✓		✓
MFGAN [122]	2020		✓							✓						✓

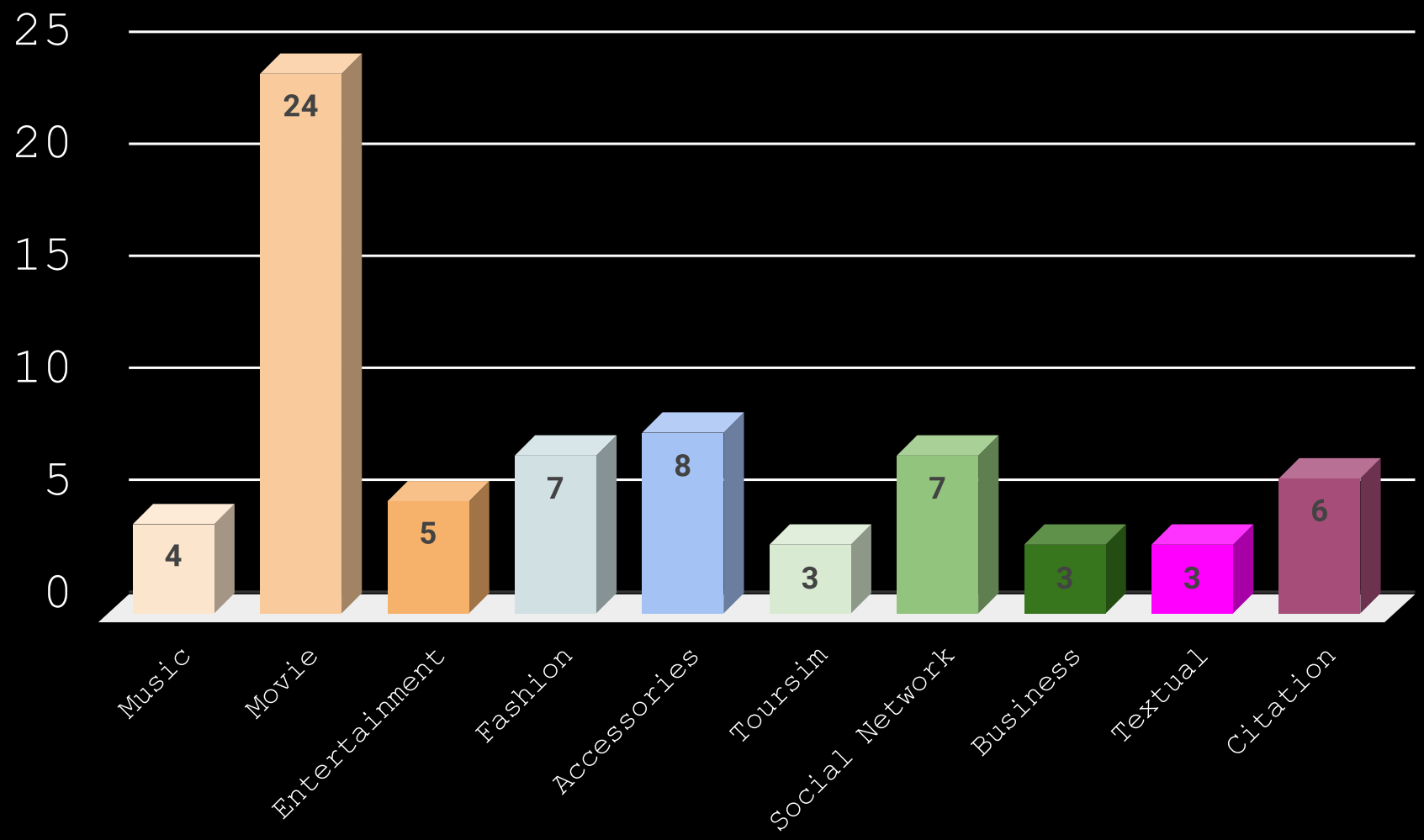
SUMMARY

- A fine-grained categorization of more the **50** GAN-based recommenders is available in our Survey
- Prevalence of **Linear** and **AutoEncoder** models
- Applications in **Cross Domain** and **Fashion Recommendation**

Geographical-aware																				
Geo-ALM [99]	2019	✓											✓						✓	
APOIR [181]	2019									✓			✓					✓		
Cross-domain Rec.																				
VAE-GAN-CC [112]	2018								✓				✓					✓		
RecSys-DAN [147]	2019				✓								✓					✓		
FR-DiscoGAN [75]	2019				✓									✓					✓	
DASO [55]	2019		✓										✓					✓		
CnGAN [119]	2019					✓							✓						✓	
Asr [83]	2019		✓			✓	✓						✓	✓	✓				✓	
ALTRec [92]	2020					✓							✓						✓	
Fashion Rec.																				
DVBPR [79]	2017				✓									✓				✓		
CRAFT [73]	2018		✓										✓					✓		
MrCGAN [132]	2018				✓									✓					✓	
Yang et al. [165]	2018				✓									✓					✓	
c^+GAN [85]	2019				✓									✓					✓	

Domains and open directions

DOMAIN



Politecnico di Bari

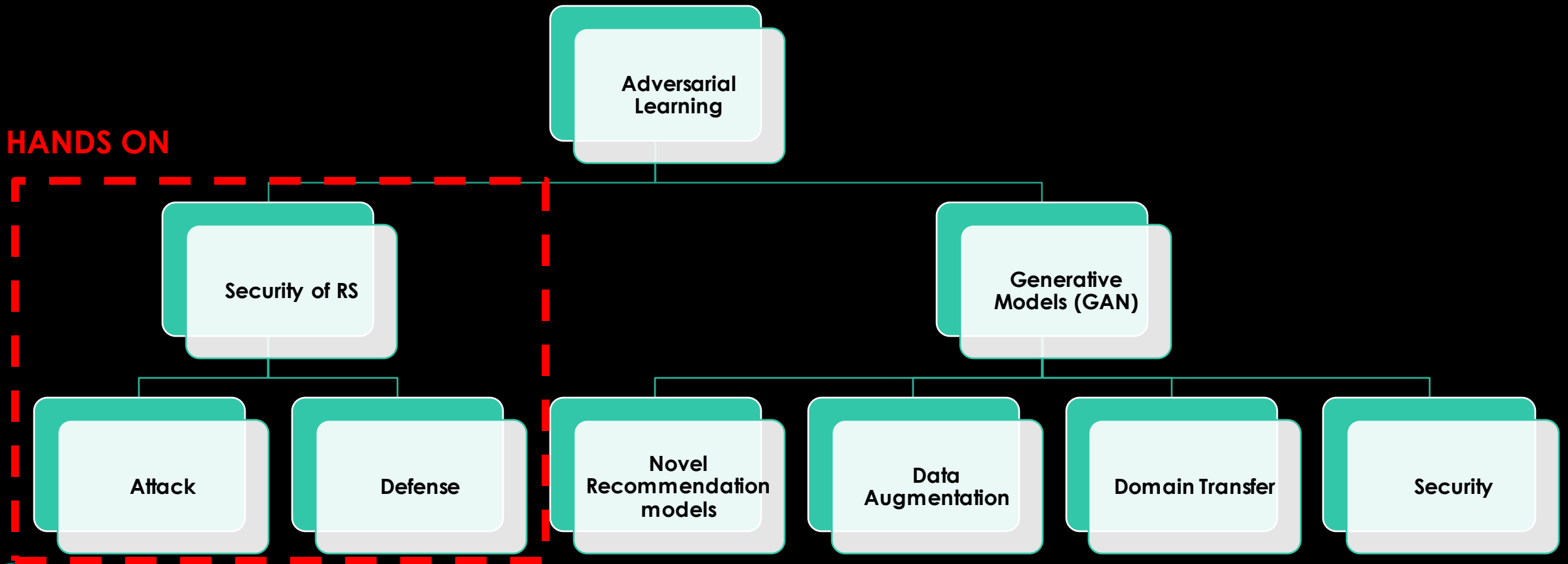
OPEN DIRECTIONS

- Novel solution for the discrete sampling non-differentiability problems
- Integrate user-personalization in complementary recommendation
- Extend GAN-RF to Knowledge-aware recommender systems
- Explore GAN-based attacks to Deep-CF RS
- Verify the effectiveness of GAN-RF with an online evaluation

HANDS-ON SESSION
ADVERSARIAL REGULARIZATION

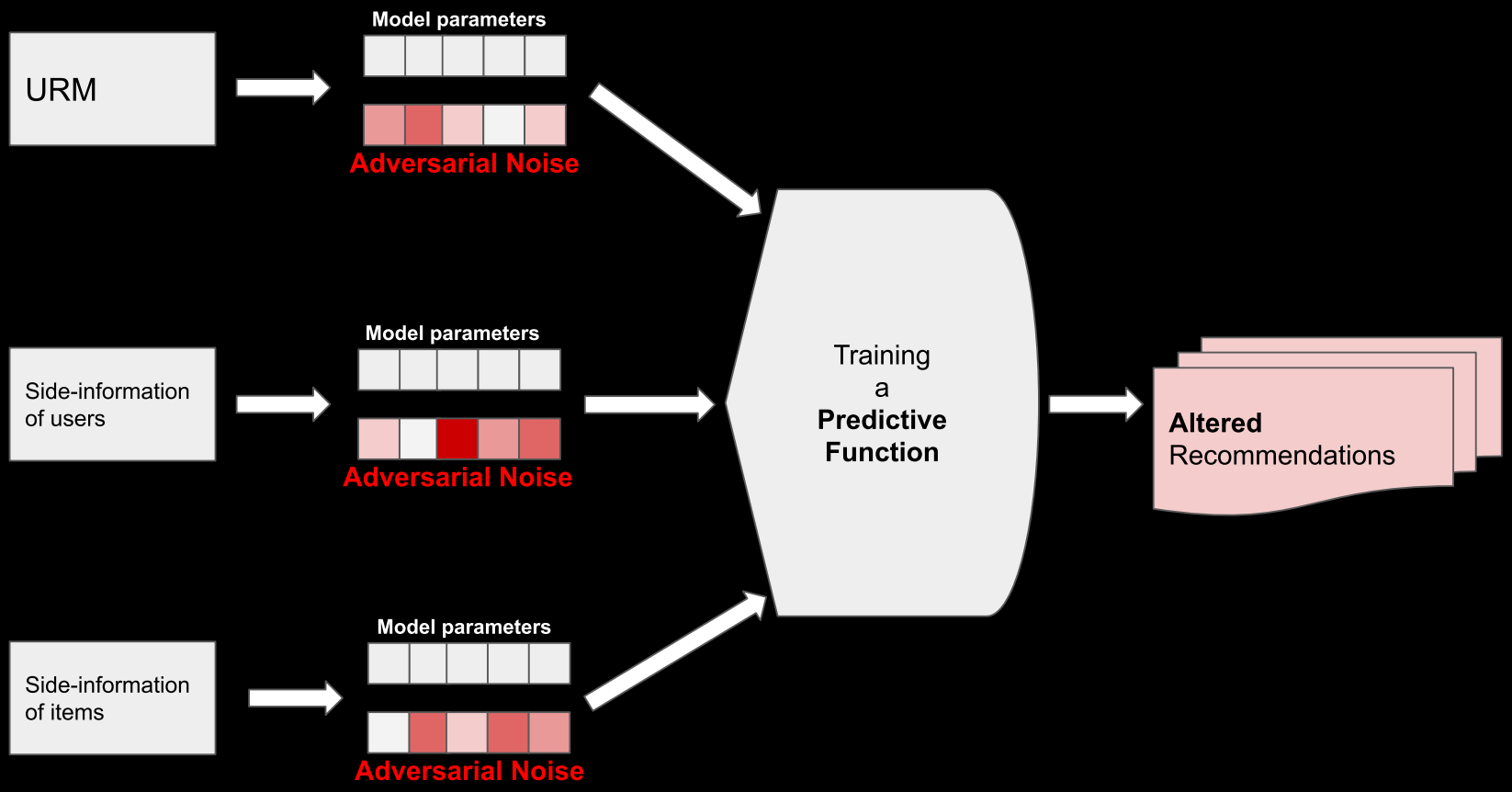
Presenter: Felice Antonio Merra

ADVERSARIAL LEARNING FOR RS



Politecnico di Bari

ADVERSARIAL ROBUST RECOMMENDATION



Politecnico di Bari

HOW TO ACCESS THE JUPYTER NOTEBOOK

https://github.com/sisinflab/HandsOn-ECIR2021

README.md

Adversarial Learning for Recommendation: Applications for Security and Generative Tasks – Concept to Code

To start the ***Hands-on Session*** access the Jupyter Notebook: launch binder

Source: 🤖Adversarial Learning for Recommendation: Applications for Security and Generative Tasks – Concept to Code repository_

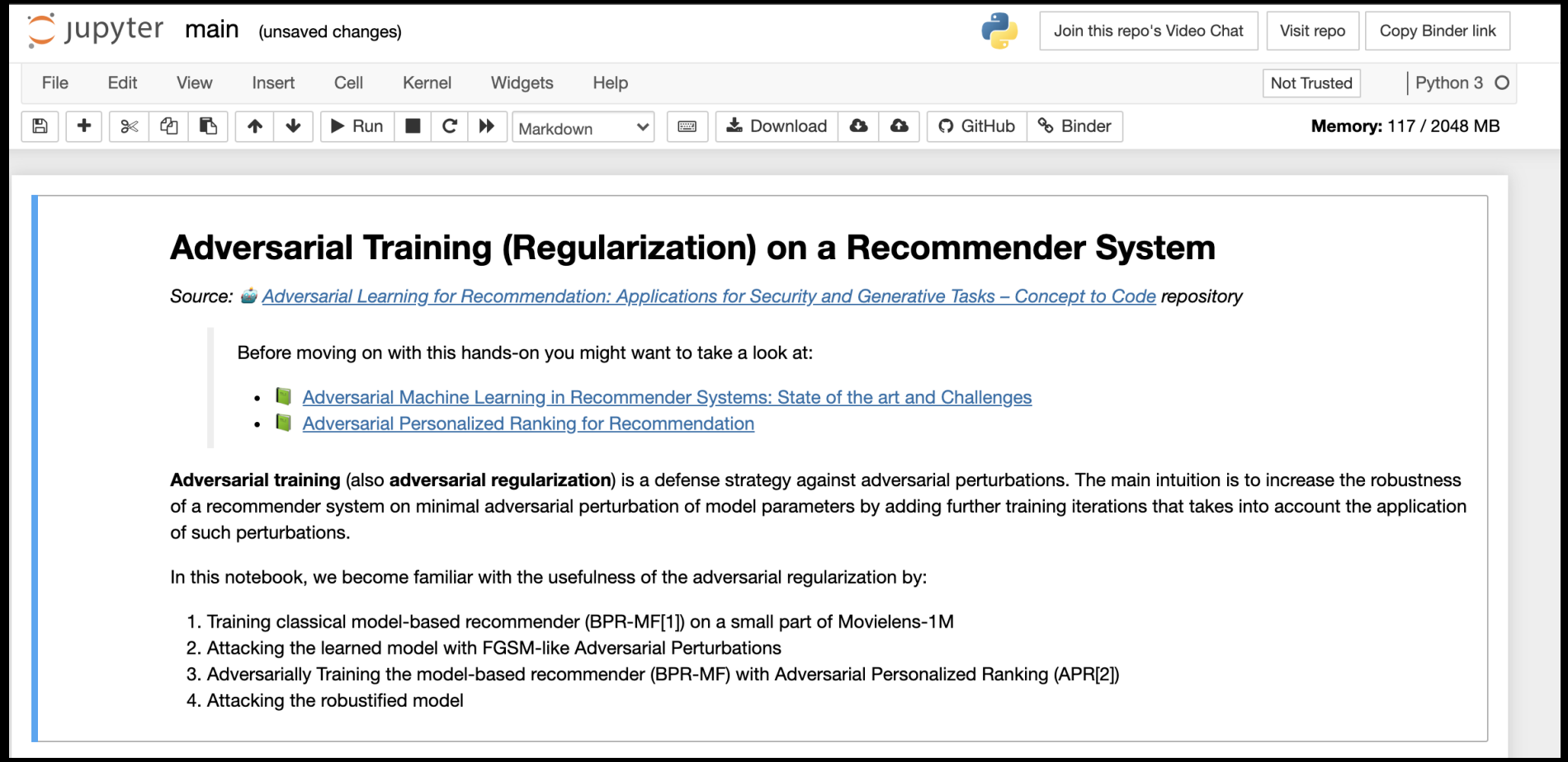
> Before moving on with this hands-on you might want to take a look at:
>
> - 📗 Adversarial Machine Learning in Recommender Systems: State of the art and Challenges
> - 📗 Adversarial Personalized Ranking for Recommendation

Click The Binder Widget

**It takes few minutes to build the interactive environment.*

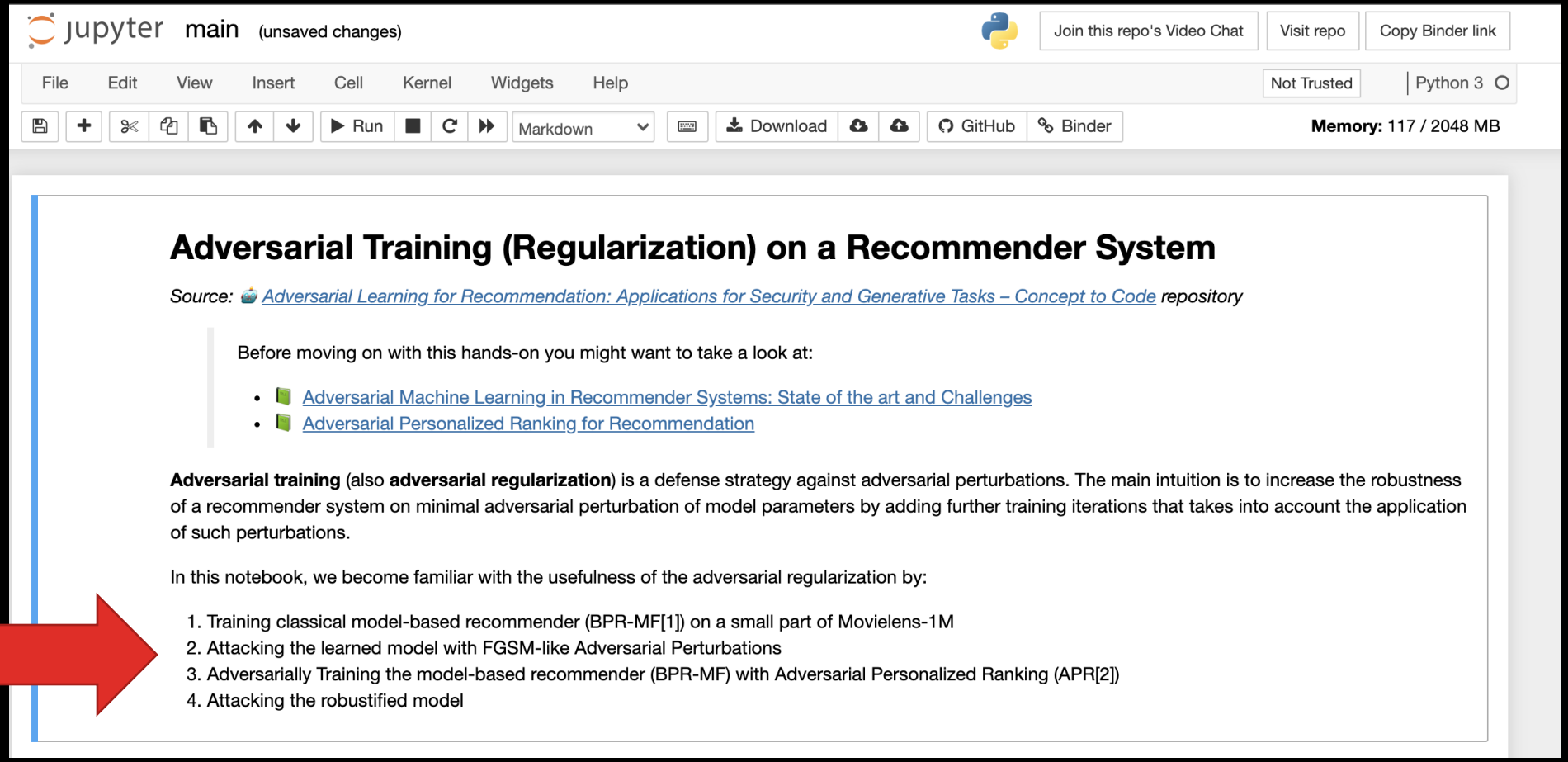
Politecnico di Bari

JUPYTER NOTEBOOK

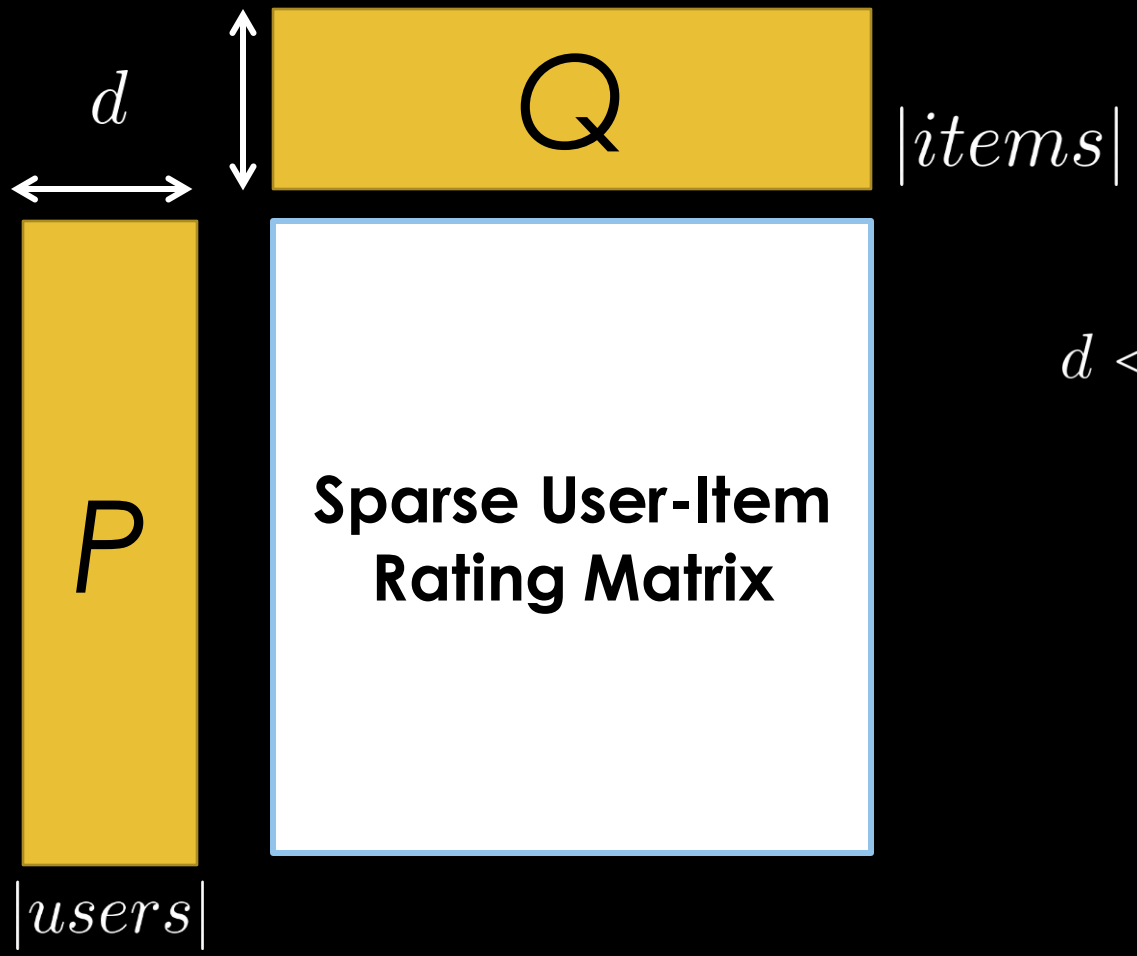


Politecnico di Bari

JUPYTER NOTEBOOK



THE RECOMMENDER MODEL
BPR-MF



$$d << |users|, |items|$$

IMPLEMENTATION BPR-MF

Define The Model

We will define a new Tensorflow 2 model class to define the model (BPR-MF). For a matter of simplicity we have also implemented the adversarial attack and defense strategies,, that will be used in the later sections.

In [ ]:

```
from src.recommender.RecommenderModel import RecommenderModel

TOPK = 100 # Top-K

class BPRMF(RecommenderModel):
    def __init__(self, data_loader, path_output_rec_result, path_output_rec_weight):
        super(BPRMF, self).__init__(data_loader, path_output_rec_result, path_output_rec_weight, 'bprmf')
        self.embedding_size = 64
        self.learning_rate = 0.05
        self.reg = 0
        self.epochs = 5
        self.batch_size = 512
        self.verbose = 1
        self.evaluator = Evaluator(self, data, TOPK)

        self.initialize_model_parameters()
        self.initialize_perturbations()
        self.initialize_optimizer()

    def initialize_model_parameters(self):
        """
            Initialize Model Parameters
        """
        self.embedding_P = tf.Variable(tf.random.truncated_normal(shape=[self.num_users, self.embedding_size], mean=0.0
        self.embedding_Q = tf.Variable(tf.random.truncated_normal(shape=[self.num_items, self.embedding_size], mean=0.0
```

Politecnico di Bari

TRAIN THE CLEAN MODEL

TRAIN STEP

```python
@timethis
def _train_step(self, batches):
    """ Apply a Single Training Step (across all the batches in the dataset). """
    user_input, item_input_pos, item_input_neg = batches

    for batch_idx in range(len(user_input)):
        with tf.GradientTape() as t:
            t.watch([self.embedding_P, self.embedding_Q])

            # Model Inference
            self.output_pos, embed_p_pos, embed_q_pos = self.get_inference(user_input[batch_idx],
                                                                           item_input_pos[batch_idx])
            self.output_neg, embed_p_neg, embed_q_neg = self.get_inference(user_input[batch_idx],
                                                                           item_input_neg[batch_idx])
            self.result = tf.clip_by_value(self.output_pos - self.output_neg, -80.0, 1e8)

            self.loss = tf.reduce_sum(tf.nn.softplus(-self.result))

            # Regularization Component
            self.reg_loss = self.reg * tf.reduce_mean(tf.square(embed_p_pos) + tf.square(embed_q_pos)
                                                      + tf.square(embed_q_neg))

            # Loss Function
            self.loss_opt = self.loss + self.reg_loss

        gradients = t.gradient(self.loss_opt, [self.embedding_P, self.embedding_Q])
        self.optimizer.apply_gradients(zip(gradients, [self.embedding_P, self.embedding_Q]))

@timethis
def train(self):
    for epoch in range(self.epochs):
        batches = self.data.shuffle(self.batch_size)
        self._train_step(batches)
        print('Epoch {0}/{1}'.format(epoch+1, self.epochs))
```

TRAIN LOOP

EVALUATE THE MODEL

Initialize and Train The Model

Now, we are ready to initialize and train the model.

```
recommender_model = BPRMF(data, '../rec_result/', '../rec_weights/')
recommender_model.train()
```

Evaluate The Model

The evaluation is computed on TOPK recommendation lists (default K = 100).

```
before_adv_hr, before_adv_ndcg, before_adv_auc = recommender_model.evaluator.evaluate()
```

ADV-RF: **ATTACKS**

- **ATTACKER Goal?** **Maximize** the recommender model objective

$$\Delta_{adv} = \arg \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

- **Attack Method?** **FGSM**

$$\Delta_{adv} = \epsilon \frac{\Pi}{\|\Pi\|} \quad \text{where} \quad \Pi = \frac{\partial J(X|\Omega + \Delta)}{\partial \Delta}$$

IMPLEMENTING THE ADVERSARIAL ATTACK

```python
def execute_adversarial_attack(self, epsilon):
    user_input, item_input_pos, item_input_neg = self.data.shuffle(len(self.data._user_input))
    self.initialize_perturbations()

    with tf.GradientTape() as tape_adv:
        tape_adv.watch([self.embedding_P, self.embedding_Q])
        # Evaluate Current Model Inference
        output_pos, embed_p_pos, embed_q_pos = self.get_inference(user_input[0],
                                                                  item_input_pos[0])
        output_neg, embed_p_neg, embed_q_neg = self.get_inference(user_input[0],
                                                                  item_input_neg[0])
        result = tf.clip_by_value(output_pos - output_neg, -80.0, 1e8)
        loss = tf.reduce_sum(tf.nn.softplus(-result))
        loss += self.reg * tf.reduce_mean(
            tf.square(embed_p_pos) + tf.square(embed_q_pos) + tf.square(embed_q_neg))
    # Evaluate the Gradient
    grad_P, grad_Q = tape_adv.gradient(loss, [self.embedding_P, self.embedding_Q])
    grad_P, grad_Q = tf.stop_gradient(grad_P), tf.stop_gradient(grad_Q)

    # Use the Gradient to Build the Adversarial Perturbations (https://doi.org/10.1145/3209978.3209981)
    self.delta_P = tf.nn.l2_normalize(grad_P, 1) * epsilon
    self.delta_Q = tf.nn.l2_normalize(grad_Q, 1) * epsilon
```

Politecnico di Bari

EVALUATING THE ATTACK

Adversarial Attack Against The Model

We can attack the model with adversarial perturbation and measure the performance after the attack. Epsilon is the perturbation budget.

```
epsilon = 0.5
print('Running the Attack with Epsilon = {0}'.format(epsilon))
recommender_model.execute_adversarial_attack(epsilon=epsilon)
print('The model has been Adversarially Perturbed.')
```

Evaluate the Effects of the Adversarial Attack

We will now evaluate the performance of the attacked model.

```
after_adv_hr, after_adv_ndcg, after_adv_auc = recommender_model.evaluator.evaluate()

print('HR decreases by %.2f%%' % ((1-after_adv_hr/before_adv_hr)*100))
print('nDCG decreases by %.2f%%' % ((1-after_adv_ndcg/before_adv_ndcg)*100))
print('AUC decreases by %.2f%%' % ((1-after_adv_auc/before_adv_auc)*100))
```

Politecnico di Bari

ADV-RF: **DEFENSE**

- **Defender Goal? Minimize** the attack influence
- **Defence Method? Adversarial (re)training**

$$\arg\min_{\Omega} J(X|\Omega) + \lambda J\left(X|\Omega + \Delta_{adv}\right)$$

$$where \quad \Delta_{adv} = \arg\max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega + \Delta)$$

MINIMAX GAME

The training process is a **MINIMAX GAME**

$$\arg\min_{\Omega} \max_{\Delta, \|\Delta\| \leq \epsilon} J(X|\Omega) + \lambda J(X|\Omega + \Delta)$$

IMPLEMENTING THE DEFENSE

BPR-MF Loss

Regularization

Adversarial Attack

Adversarial Regularizer

Adversarial Regularized Loss Function

```
self.loss = tf.reduce_sum(tf.nn.softplus(-self.result))

# Regularization Component
self.reg_loss = self.reg * tf.reduce_mean(tf.square(embed_p_pos) + tf.square(embed_q_pos) + tf.square(e

# Adversarial Regularization Component
##  Execute the Adversarial Attack on the Current Model (Perturb Model Parameters)
self.execute_adversarial_attack(epsilon)
##  Inference on the Adversarial Perturbed Model
self.output_pos_adver, _, _ = self.get_inference(user_input[batch_idx], item_input_pos[batch_idx],
self.output_neg_adver, _, _ = self.get_inference(user_input[batch_idx], item_input_neg[batch_idx])

self.result_adver = tf.clip_by_value(self.output_pos_adver - self.output_neg_adver, -80.0, 1e8)
self.loss_adver = tf.reduce_sum(tf.nn.softplus(-self.result_adver))

# Loss Function
self.adversarial_regularizer = adv_reg * self.loss_adver # AMF = Adversarial Matrix Factorization
self.bprmf_loss = self.loss + self.reg_loss

self.amf_loss = self.bprmf_loss + self.adversarial_regularizer
```

Politecnico di Bari

EVALUATING THE ATTACK ON THE DEFENDED MODEL

Evaluated The Adversarially Defended Model before the Attack

```
before_adv_hr, before_adv_ndcg, before_adv_auc = recommender_model.evaluator.evaluate()
```

Adversarial Attack Against The Defended Model

```
recommender_model.execute_adversarial_attack(epsilon=0.5)
```

Evaluate the Effects of the Adversarial Attack against the Defended Model

```
after_adv_hr, after_adv_ndcg, after_adv_auc = recommender_model.evaluator.evaluate()

print('HR decreases by %.2f%%' % ((1-after_adv_hr/before_adv_hr)*100))
print('nDCG decreases by %.2f%%' % ((1-after_adv_ndcg/before_adv_ndcg)*100))
print('AUC decreases by %.2f%%' % ((1-after_adv_auc/before_adv_auc)*100))
```

ELLIOT

THE REPRODUCIBILITY FRAMEWORK

TO TEST YOUR ADVERSARIAL MODEL

https://github.com/sisinflab/elliot

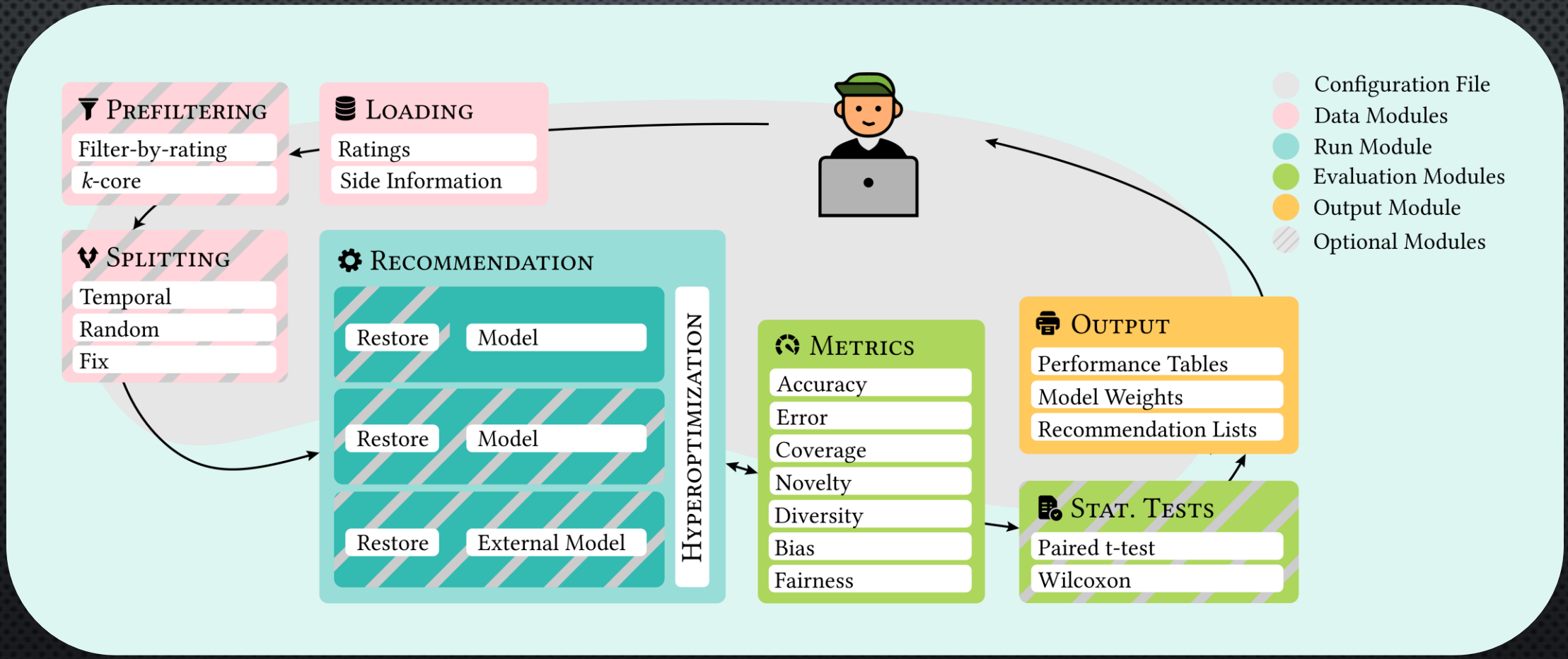
A FEW WORDS ABOUT REPRODUCIBILITY

- KEY-ELEMENTS: DATASET COLLECTION, DATA SPLITTING, RECOMMENDATION, CANDIDATE ITEM FILTERING, EVALUATION, AND STATISTICAL TESTING *
- METHODOLOGICAL ISSUES: CHOICE OF BASELINES, PROPAGATION OF WEAK BASELINES, LACK OF PROPER TUNING OF BASELINES **
- GUIDELINES AND BEST PRACTICES **:
  - METHODOLOGY: INCLUDE BASELINES ALGORITHMS FROM DIFFERENT FAMILIES, SYSTEMATICALLY OPTIMIZE ALL ALGORITHMS, CAREFULLY CHOOSE THE HYPERPARAMETER OPTIMIZATION STRATEGY
  - MINIMUM DETAILS TO PROVIDE: STATE UNDERLYING ASSUMPTIONS, BE CLEAR ABOUT DATA PRE-PROCESSING, REPORT HYPERPARAMETER OPTIMIZATION DETAILS
  - EXTENDED NEURIPS GUIDELINES: PUBLISH THE SOURCE CODE OF ALL MODELS, INCLUDING THE BASELINES, PROVIDE THE DATASETS, MAKE THE REPRODUCTION EASY FOR OTHERS, USE PERSISTENT REPOSITORIES

* (BELLOGÍN & SAID, 2021, UNDER REVIEW)

** (FERRARI DACREMA, BOGLIO, CREMONESI, JANNACH, 2021, TOIS)

ELLIOT'S PIPELINE



CONFIGURATION FILE

```
experiment:
  dataset: movielens_1m
  data_config:
    strategy: dataset
    dataset_path: ../data/movielens_1m/dataset.tsv
    splitting:
      test_splitting:
        strategy: random_subsampling
        test_ratio: 0.2
    models:
      ItemKNN:
        meta:
          hyper_opt_alg: grid
          save_recs: True
        neighbors: [50, 100]
        similarity: cosine
    evaluation:
      simple_metrics: [nDCG]
    top_k: 10
```

DATA LOADING

- DATASET
- FIXED
- HIERARCHY

```
experiment:
  data_config:
    strategy: dataset
    dataset_path: this/is/the/path.tsv
```

```
experiment:
  data_config:
    strategy: fixed
    train_path: this/is/the/path.tsv
    validation_path: this/is/the/path.tsv
    test_path: this/is/the/path.tsv
```

PREFILTERING: THRESHOLD-BASED

- GLOBAL THRESHOLD
- USER AVERAGE

```
experiment:
  prefiltering:
    strategy: global_threshold
    threshold: 3
```

```
experiment:
  prefiltering:
    strategy: global_threshold
    threshold: average
```

```
experiment:
  prefiltering:
    strategy: user_average
```

PREFILTERING: K-CORE

- User k-core
- Item k-core
- Iterative k-core
- N-rounds k-core
- Cold users

```
experiment:
  prefiltering:
    strategy: user_k_core
    core: 5
```

```
experiment:
  prefiltering:
    strategy: n_rounds_k_core
    core: 5
    rounds: 2
```

SPLITTING: TIME-AWARE

- FIXED TIMESTAMP
- TEMPORAL HOLD-OUT
- TEMPORAL LEAVE-N-OUT

```
experiment:
  splitting:
    test_splitting:
      strategy: fixed_timestamp
      timestamp: 1609786061
```

```
experiment:
  splitting:
    test_splitting:
      strategy: fixed_timestamp
      timestamp: best
```

```
experiment:
  splitting:
    test_splitting:
      strategy: temporal_hold_out
      test_ratio: 0.2
```

SPLITTING: K-FOLD VALIDATION

- K-FOLD CROSS VALIDATION
- NESTED K-FOLD CROSS VALIDATION

```
experiment:
  splitting:
    test_splitting:
        strategy: random_cross_validation
        folds: 5
```

SPLITTING: K RANDOM SUBSAMPLING

- HOLD-OUT
- LEAVE-N-OUT
- K-RANDOM SUBSAMPLING WITH HOLD-OUT
- K-RANDOM SUBSAMPLING WITH LEAVE-N-OUT

```
experiment:
  splitting:
    test_splitting:
      strategy: random_subsampling
      test_ratio: 0.2
```

```
experiment:
  splitting:
    test_splitting:
      strategy: random_subsampling
      test_ratio: 0.2
      folds: 5
```

```
experiment:
  splitting:
    test_splitting:
      strategy: random_subsampling
      leave_n_out: 1
      folds: 5
```

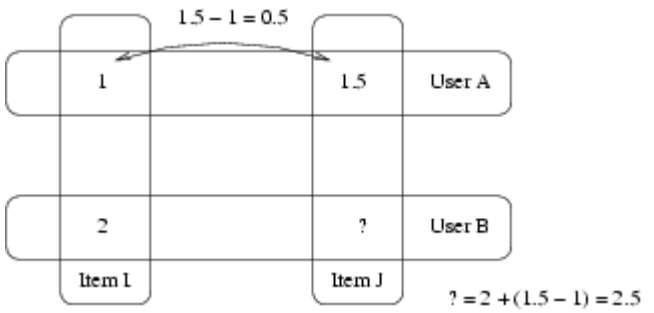
RECOMMENDATION: ADVERSARIAL LEARNING

- ADVERSARIAL MATRIX FACTORIZATION
- ADVERSARIAL MULTIMEDIA RECOMMENDER

```
models:
  AMF:
    meta:
      save_recs: True
    epochs: 10
    batch_size: 512
    factors: 200
    lr: 0.001
    l_w: 0.1
    l_b: 0.001
    eps: 0.1
    l_adv: 0.001
    adversarial_epochs: 10
```

RECOMMENDATION: ALGEBRIC

- SLOPE ONE PREDICTOR

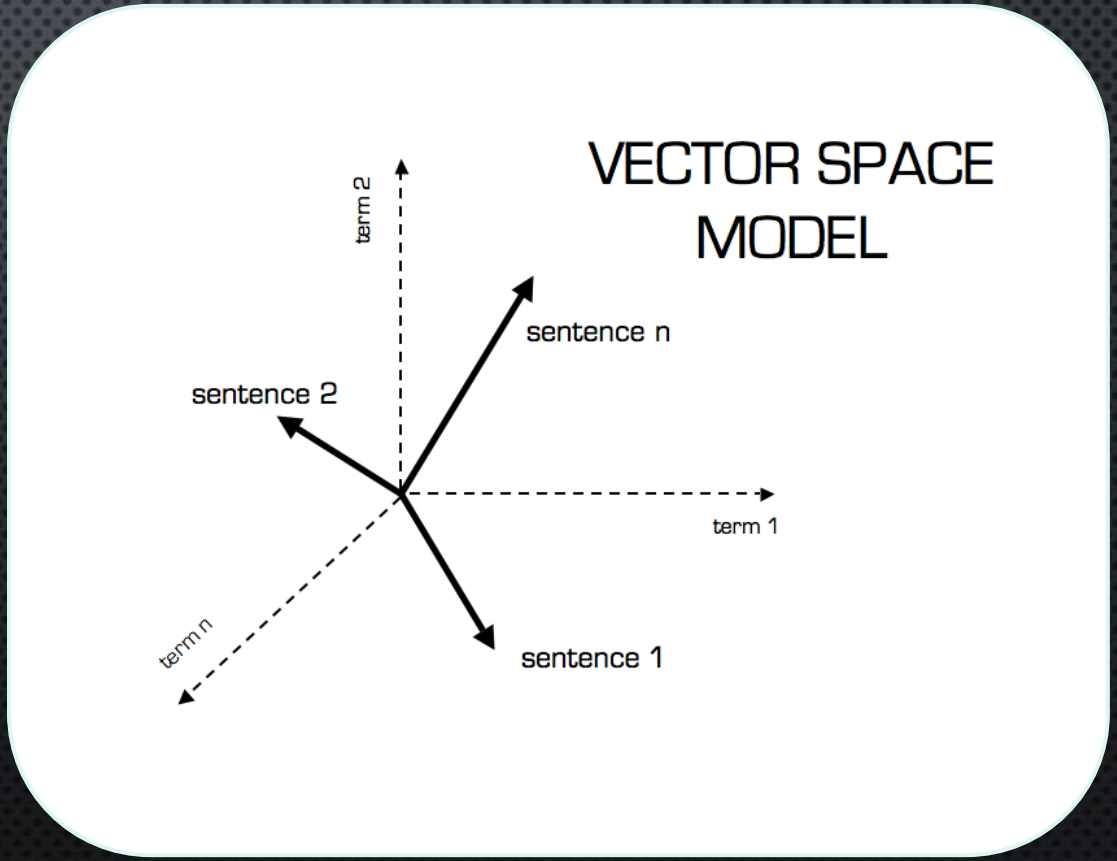


RECOMMENDATION: AUTOENCODERS

- COLLABORATIVE DENOISING AUTOENCODER
- VARIATIONAL AUTOENCODERS FOR COLLABORATIVE FILTERING

RECOMMENDATION: CONTENT-BASED

- VECTOR SPACE MODEL



Recommendation: Generative Adversarial Networks (GANs)

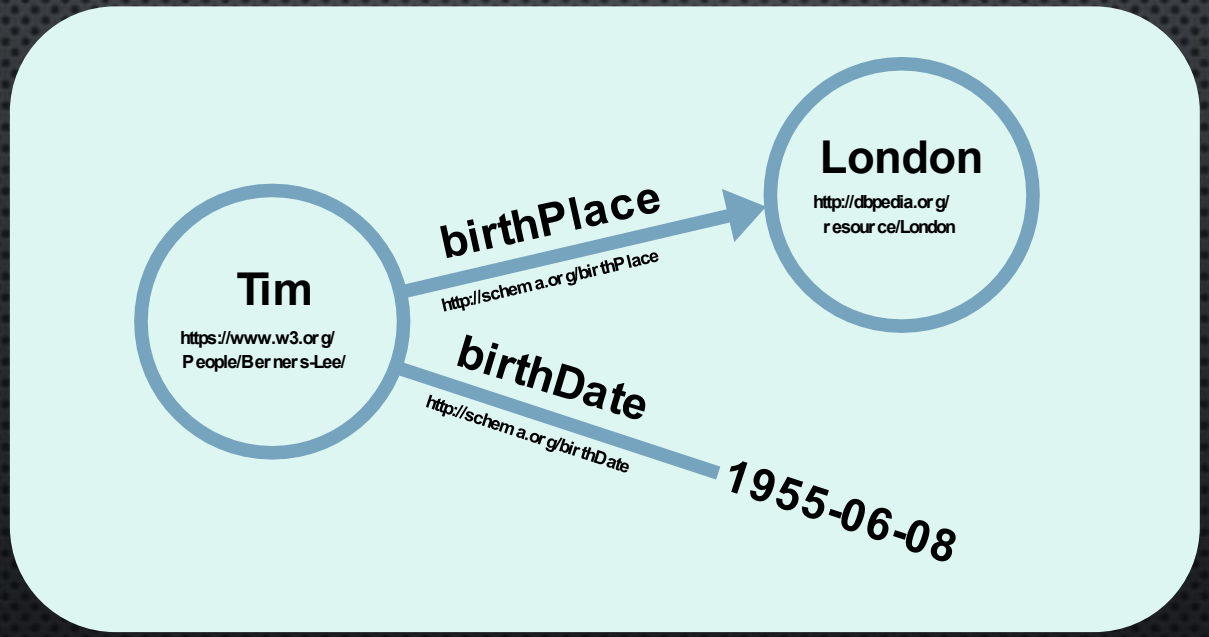
- IRGAN: A Minimax Game for Unifying Generative and Discriminative Information Retrieval Models
- CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks

RECOMMENDATION: GRAPH-BASED

- LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation
- Neural Graph Collaborative Filtering

RECOMMENDATION: KNOWLEDGE-AWARE

- KNOWLEDGE-AWARE HYBRID FACTORIZATION MACHINES



Recommendation: Latent Factor Models

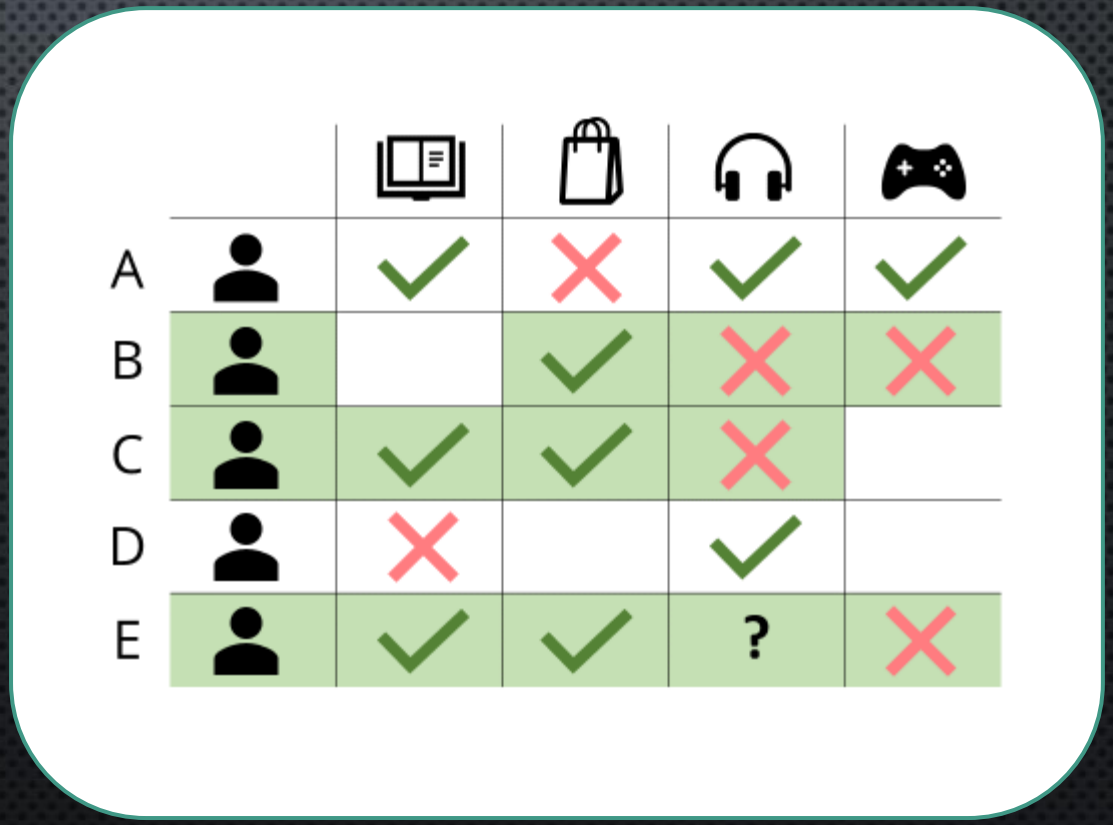
- Bayesian Personalized Ranking with Matrix Factorization
- BPR-SLIM: BPR Sparse Linear Methods
- Collaborative Metric Learning
- Field-aware Factorization Machines
- FISM: Factored Item Similarity Models
- Factorization Machines
- FunkSVD
- Logistic Matrix Factorization
- Matrix Factorization
- Non-Negative Matrix Factorization
- Probabilistic Matrix Factorization
- PureSVD
- SLIM: Sparse Linear Methods
- SVD++
- Weighted Regularized Matrix Factorization

Recommendation: Artificial Neural Networks

- Convolutional Matrix Factorization for Document Context-Aware Recommendation
- Outer Product-based Neural Collaborative Filtering
- DeepFM: A Factorization-Machine based Neural Network for CTR Prediction
- Deep Matrix Factorization Models for Recommender Systems
- Neural Collaborative Filtering
- AutoRec: Autoencoders Meet Collaborative Filtering
- NAIS: Neural Attentive Item Similarity Model for Recommendation
- Neural Collaborative Filtering
- Neural Factorization Machines for Sparse Predictive Analytics
- Neural Personalized Ranking for Image Recommendation
- Wide & Deep Learning for Recommender Systems

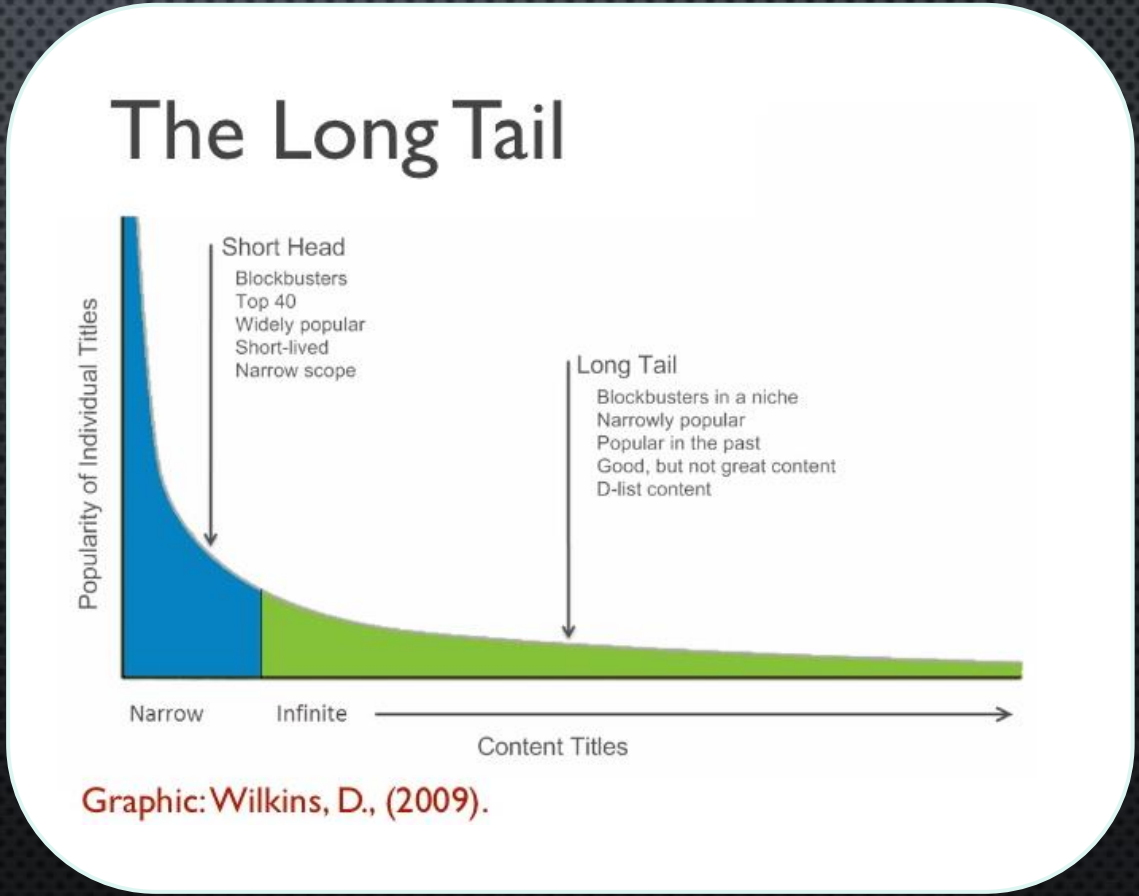
RECOMMENDATION: NEIGHBORHOOD-BASED MODELS

- ITEM K-NN
- USER K-NN
- ATTRIBUTE ITEM K-NN
- ATTRIBUTE USER K-NN



RECOMMENDATION: UNPERSONALIZED RECOMMENDERS

- MOST POPULAR
- RANDOM



Recommendation: visual models

- Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention
- DeepStyle: Learning User Preferences for Visual Recommendation
- Visually-Aware Fashion Recommendation and Design with Generative Image Models
- VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback
- Visual Neural Personalized Ranking for Image Recommendation
- Adversarial Multimedia Recommender

EVALUATION: ACCURACY

- Area Under the Curve
- Group Area Under the Curve
- Limited Area Under the Curve
- Sørensen–Dice coefficient
- F-Measure
- Extended F-Measure
- Hit Rate
- Mean Average Precision
- Mean Average Recall
- Mean Reciprocal Rank
- normalized Discounted Cumulative Gain

```
experiment:
  top_k: 50
  evaluation:
    cutoffs: [10, 5]
    simple_metrics: [ nDCG, Precision, Recall]
    relevance_threshold: 1
    paired_ttest: True
    wilcoxon_test: True
    complex_metrics:
    - metric: DSC
      beta: 2
    - metric: SRecall
      feature_data: this/is/the/path.tsv
```

EVALUATION: RATING

- MEAN ABSOLUTE ERROR
- MEAN SQUARED ERROR
- ROOT MEAN SQUARED ERROR

EVALUATION: COVERAGE

- Item Coverage
- Number of Recommendations Retrieved
- User Coverage
- User Coverage on Top-N

EVALUATION: NOVELTY

- EXPECTED FREE DISCOVERY (EFD)
- EXTENDED EFD
- EXPECTED POPULARITY COMPLEMENT (EPC)
- EXTENDED EPC

EVALUATION: DIVERSITY

- GINI INDEX
- SHANNON ENTROPY
- SUBTOPIC RECALL

EVALUATION: BIAS

- ACLT: Average coverage of long tail items
- APLT: Average percentage of long tail items
- ARP: Average Recommendation Popularity
- Popularity-based Ranking-based Equal Opportunity
- Popularity-based Ranking-based Statistical Parity

EVALUATION: FAIRNESS

- BIAS DISPARITY – STANDARD
- BIAS DISPARITY - BIAS RECOMMENDATIONS
- BIAS DISPARITY - BIAS SOURCE
- ITEM MAD RANKING-BASED
- ITEM MAD RATING-BASED
- USER MAD RANKING-BASED
- USER MAD RATING-BASED
- RANKING-BASED EQUAL OPPORTUNITY
- RANKING-BASED STATISTICAL PARITY

HYPERPARAMETER OPTIMIZATION: STRATEGIES

- GRID SEARCH
- TREE OF PARZEN ESTIMATORS
- ADAPTIVE TREE OF PARZEN ESTIMATORS
- RANDOM SEARCH
- MIXED
- SIMULATED ANNEALING

```
experiment:
  models:
    PMF:
      meta:
        hyper_max_evals: 20
        hyper_opt_alg: tpe
        validation_rate: 1
        verbose: True
        save_weights: True
        save_recs: True
        validation_metric: nDCG@10
      lr: 0.0025
      epochs: 2
      factors: 50
      batch_size: 512
      reg: 0.0025
      reg_b: 0
      gaussian_variance: 0.1
```

HYPERPARAMETER OPTIMIZATION: SEARCH SPACES

- Choice
- Randint
- Uniform
- Quniform
- Loguniform
- Qloguniform
- Normal
- Qnormal
- Lognormal
- qlognormal

```
experiment:
  models:
    PMF:
      meta:
        hyper_max_evals: 20
        hyper_opt_alg: tpe
      lr:                [loguniform, -10, -1]
      epochs:            2
      factors:           50
      batch_size:        512
      reg:               [0.0025, 0.005, 0.01]
      reg_b:             0
      gaussian_variance: 0.1
```

STATISTICAL SIGNIFICANCE OF RESULTS

- WILCOXON STATISTICAL TEST
- STUDENT'S PAIRED T-TEST

```
experiment:
  evaluation:
    paired_ttest: True
    wilcoxon_test: True
```

EXAMPLE: AN ADVERSARIAL MODEL

- Model: **Adversarial matrix factorization**
- Splitting: **random hold-out 80-20**
- Evaluation metrics: **Hit Rate @ 50**
- Stastistical significance of results: **Student's paired t-test, Wilcoxon test**
- Epochs: **10**
- Batch-size: **256**
- Bias Regularization: **0.0001**
- Weight Regularization: **0.0001**
- Adversarial Regularization: **1**
- For the remaining hyperparameters:
  - **Hyperparamenter optimization strategy: Bayesian optimization**

ADVERSARIAL MODEL CONFIGURATION

```yaml
experiment:
  dataset: movielens_1m
  data_config:
    strategy: dataset
    dataset_path: ../data/movielens_1m/dataset.tsv
  splitting:
    test_splitting:
      strategy: random_subsampling
      test_ratio: 0.2
  top_k: 50
  evaluation:
    cutoffs: 50
    simple_metrics: [HR, nDCG, EFD, ACLT, APLT]
    paired_ttest: True
    wilcoxon_test: True
  gpu: 1
  models:
    AMF:
      meta:
        save_recs: False
        hyper_max_evals: 5
        hyper_opt_alg: tpe
      epochs: 10
      batch_size: 256
      factors: [quniform, 50, 100, 1]
      lr: [loguniform, -2, -1]
      l_w: 0.0001
      l_b: 0.0001
      eps: [uniform, 0.1, 0.5] # Magnitude of the Perturbation
      l_adv: 1 # Adversarial Regularization Parameters
      adversarial_epochs: [quniform, 4, 8, 1]
          ...
```

```python
import zipfile
import io
import requests
import os

from elliot.Run import run_experiment

url = "http://files.grouplens.org/datasets/movielens/ml-1m.zip"
print(f"getting movielens 1million from : {url} ..")
response = requests.get(url)

ml_1m_ratings = []

print("extracting ratings.dat ..")
with zipfile.zipfile(io.bytesio(response.content)) as zip_ref:
  for line in zip_ref.open("ml-1m/ratings.dat"):
    ml_1m_ratings.append(str(line, "utf-8").replace("::", "\t"))

print("printing ratings.tsv to data/movielens_1m/ ..")
os.makedirs("data/movielens_1m", exist_ok=true)
with open("data/movielens_1m/dataset.tsv", "w") as f:
  f.writelines(ml_1m_ratings)

print("done! We are now starting the elliot's experiment")
run_experiment("config_files/sample_adversarial.yml")
```

BEST HYPERPARAMETERS


```
[
    {
        "default_validation_metric": "HR",
        "default_validation_cutoff": 50,
        "rel_threshold": 0
    },
    {
        "meta": {
            "save_recs": false,
            "hyper_max_evals": 5,
            "hyper_opt_alg": "tpe"
        },
        "recommender": "AMF",
        "configuration": {
            "epochs": 10,
            "batch_size": 256,
            "factors": 86.0,
            "lr": 0.20394681270896606,
            "l_w": 0.0001,
            "l_b": 0.0001,
            "eps": 0.47314719953913353,
            "l_adv": 1,
            "adversarial_epochs": 5.0,
            "name": "..."
        }
    },
...
]
```

EVALUATION RESULTS

model	HR	nDCG	EFD	ACLT	APLT
AMF	0.7597	0.0655	0.4250	1.8717	0.0374
NeuMF	0.9763	0.2912	2.0393	3.3798	0.0676

model	e	bs	factors	lr	lw	lb	eps	ladv	Adv epochs	HR	nDCG	EFD	ACLT	APLT
AMF	10	256	92	0.2132	0.0001	0.0001	0.1276	1	5	0.7563	0.0676	0.4358	2.2939	0.0459
AMF	10	256	86	0.2039	0.0001	0.0001	0.4731	1	5	0.7597	0.0655	0.4250	1.8717	0.0374
AMF	10	256	94	0.3127	0.0001	0.0001	0.2214	1	8	0.7008	0.0528	0.3704	3.6809	0.0736
AMF	10	256	96	0.2112	0.0001	0.0001	0.1419	1	6	0.7483	0.0632	0.4123	1.9432	0.0389
AMF	10	256	59	0.3050	0.0001	0.0001	0.3392	1	7	0.7211	0.0560	0.3823	2.9942	0.0599

STUDENT'S PAIRED TTEST

Model 0	Model 1	Metric	P-value
AMF	NeuMF	HR	2.7459735368482E-299
NeuMF	AMF	HR	2.7459735368482E-299
AMF	NeuMF	nDCG	0
NeuMF	AMF	nDCG	0
AMF	NeuMF	EFD	0
NeuMF	AMF	EFD	0
AMF	NeuMF	ACLT	1.67288727400142E-52
NeuMF	AMF	ACLT	1.67288727400142E-52
AMF	NeuMF	APLT	1.67288727400147E-52
NeuMF	AMF	APLT	1.67288727400147E-52

WILCOXON TEST

Model 0	Model 1	Metric	P-value
AMF	NeuMF	HR	4.1137985478166E-268
NeuMF	AMF	HR	4.1137985478166E-268
AMF	NeuMF	nDCG	0
NeuMF	AMF	nDCG	0
AMF	NeuMF	EFD	0
NeuMF	AMF	EFD	0
AMF	NeuMF	ACLT	9.09682112936256E-138
NeuMF	AMF	ACLT	9.09682112936256E-138
AMF	NeuMF	APLT	2.01678107092771E-137
NeuMF	AMF	APLT	2.01678107092771E-137

ELLIOT

THE REPRODUCIBILITY FRAMEWORK

TO TEST YOUR ADVERSARIAL MODEL

https://github.com/sisinflab/elliot

SECOND EXAMPLE: GANS

- Models: **IRGAN, CFGAN**
- Splitting: **random hold-out 80-20**
- Evaluation metrics: **Hit Rate, RMSE, ItemCoverage, PopREO, PopRSP**
- Cutoff: **50**

- **IRGAN:**
  - Epochs: **10**
  - Batch-size: **128**
  - Bias Regularization: **0**
  - Weight Regularization: **0.000001**
  - GAN Regularization: **0.0001**
  - For the remaining hyperparameters:
    - **Hyperparamenter optimization strategy: Grid search**

- **CFGAN:**
  - Epochs: **2**
  - Batch-size: **32**
  - Bias Regularization: **0.001**
  - Adversarial Regularization: **0.001**
  - Discriminative epochs: **1**
  - For the remaining hyperparameters:
    - **Hyperparamenter optimization strategy: Bayesian optimization**

GAN MODELS CONFIGURATION

```
experiment:
  dataset: movielens_100k
  data_config:
    strategy: dataset
    dataset_path: ../data/movielens_100k/dataset.tsv
  splitting:
    test_splitting:
      strategy: random_subsampling
      test_ratio: 0.2
  top_k: 10
  evaluation:
    simple_metrics: [HR, RMSE, ItemCoverage, PopREO, PopRSP]
  gpu: 0
  models:
    IRGAN:
      meta:
        validation_rate: 5
      epochs: 10
      batch_size: 128
      factors: [50, 100]
      lr: [0.00001, 0.0001]
      l_w: 0.000001
      l_b: 0
      l_gan: 0.0001
      predict_model: generator # You can chose also 'discriminator'
      g_epochs: 2 # generator epochs for each model epoch
      d_epochs: 1 # discriminator epochs for each model epoch
      g_pretrain_epochs: [0, 2] # pre-train the generator
      d_pretrain_epochs: [0, 2] # pre-train the discriminator
      sample_lambda: [0.2, 0.5]
          ...
```

```
          ...
  CFGAN:
    meta:
      hyper_max_evals: 5
      hyper_opt_alg: tpe
      validation_metric: HR
    epochs: 2
    batch_size: 32
    factors: [50, 100]
    lr: [0.001, 0.1]
    l_w: [0.00001, 0.01]
    l_b: 0.001
    l_gan: 0.001
    g_epochs: [1, 5]
    d_epochs: 1
    s_zr: [0.001, 0.1]
    s_pm: [0.001, 0.1]
```

BEST HYPERPARAMETERS


```
[
    {
        "default_validation_metric": "HR",
        "default_validation_cutoff": 10,
        "rel_threshold": 0
    },
    {
        "meta": {
            "hyper_opt_alg": "grid",
            "validation_rate": 5,
            "verbose": true,
            "save_weights": false,
            "save_recs": false,
            "validation_metric": "HR"
        },
        "recommender": "IRGAN",
        "configuration": {
            "epochs": 10,
            "batch_size": 128,
            "factors": 100,
            "lr": 0.0001,
            "l_w": 1e-06,
            "l_b": 0,
            "l_gan": 0.0001,
            "predict_model": "generator",
            "g_epochs": 2,
            "d_epochs": 1,
            "g_pretrain_epochs": 0,
            "d_pretrain_epochs": 2,
            "sample_lambda": 0.5,
            "name": "..."
        }
    },
...
```

```
...
    {
        "meta": {
            "save_weights": false,
            "save_recs": false,
            "hyper_max_evals": 5,
            "hyper_opt_alg": "tpe",
            "validation_metric": "HR"
        },
        "recommender": "CFGAN",
        "configuration": {
            "epochs": 2,
            "batch_size": 32,
            "factors": 100,
            "lr": 0.1,
            "l_w": 1e-05,
            "l_b": 0.001,
            "l_gan": 0.001,
            "g_epochs": 1,
            "d_epochs": 1,
            "s_zr": 0.1,
            "s_pm": 0.001,
            "name": "..."
        }
    }
]
```

EVALUATION RESULTS

model	HR	RMSE	I-Cov	PopREO	PopRSP
IRGAN	0.6401	inf	42	1	1
CFGAN	0.3015	inf	32	0.8982	0.8933

model	e	bs	factors	lr	lw	lb	lgan	gepochs	depochs	szr	spm	HR	RMSE	I-Cov	PopREO	PopRSP
CFGAN	2	32	100	0.1	0.01	0.001	0.001	5	1	0.1	0.1	0.1263	inf	1209	0.0076	0.0640
CFGAN	2	32	50	0.001	0.01	0.001	0.001	1	1	0.001	0.1	0.1221	inf	479	0.2797	0.0805
CFGAN	2	32	50	0.001	0.01	0.001	0.001	5	1	0.1	0.001	0.1008	inf	510	0.1888	0.0944
CFGAN	2	32	100	0.1	1.00E-05	0.001	0.001	1	1	0.1	0.001	0.3015	inf	32	0.8982	0.8933
CFGAN	2	32	100	0.001	1.00E-05	0.001	0.001	5	1	0.001	0.001	0.3015	inf	32	0.8982	0.8933

IRGAN HYPERPARAMETERS' EXPLORATION

model	e	bs	predmodel	factors	lr	lw	lb	lgan	gepochs	depochs	gptep	dptep	samplelambda	HR	RMSE	I-Cov	PopREO	PopRSP
IRGAN	10	128	gen	50	0.0001	1.00E-06	0	0.0001	2	1	0	0	0.5	0.6136	inf	41	1.0000	1.0000
IRGAN	10	128	gen	100	0.0001	1.00E-06	0	0.0001	2	1	2	2	0.2	0.6359	inf	44	1.0000	1.0000
IRGAN	10	128	gen	100	1.00E-05	1.00E-06	0	0.0001	2	1	0	2	0.2	0.4130	inf	915	0.9698	0.9590
IRGAN	10	128	gen	100	0.0001	1.00E-06	0	0.0001	2	1	2	0	0.2	0.6327	inf	44	1.0000	1.0000
IRGAN	10	128	gen	100	1.00E-05	1.00E-06	0	0.0001	2	1	0	0	0.5	0.4268	inf	922	0.9710	0.9566
IRGAN	10	128	gen	50	1.00E-05	1.00E-06	0	0.0001	2	1	2	2	0.5	0.3747	inf	960	0.9326	0.9398
IRGAN	10	128	gen	100	1.00E-05	1.00E-06	0	0.0001	2	1	2	0	0.2	0.3248	inf	1109	0.8664	0.8845
IRGAN	10	128	gen	50	0.0001	1.00E-06	0	0.0001	2	1	0	0	0.2	0.6136	inf	41	1.0000	1.0000
IRGAN	10	128	gen	50	1.00E-05	1.00E-06	0	0.0001	2	1	0	2	0.2	0.4448	inf	514	0.9863	0.9930
IRGAN	10	128	gen	50	0.0001	1.00E-06	0	0.0001	2	1	2	0	0.2	0.6348	inf	42	1.0000	1.0000
IRGAN	10	128	gen	50	1.00E-05	1.00E-06	0	0.0001	2	1	2	0	0.5	0.3662	inf	975	0.9316	0.9352
IRGAN	10	128	gen	50	0.0001	1.00E-06	0	0.0001	2	1	0	2	0.2	0.6115	inf	45	1.0000	1.0000
IRGAN	10	128	gen	100	0.0001	1.00E-06	0	0.0001	2	1	0	0	0.5	0.6274	inf	42	1.0000	1.0000
IRGAN	10	128	gen	50	1.00E-05	1.00E-06	0	0.0001	2	1	0	2	0.5	0.4193	inf	589	0.9853	0.9844
IRGAN	10	128	gen	50	0.0001	1.00E-06	0	0.0001	2	1	2	2	0.2	0.6274	inf	46	1.0000	1.0000
IRGAN	10	128	gen	50	1.00E-05	1.00E-06	0	0.0001	2	1	0	0	0.2	0.4926	inf	536	1.0000	0.9918
IRGAN	10	128	gen	100	0.0001	1.00E-06	0	0.0001	2	1	2	0	0.5	0.6263	inf	47	1.0000	1.0000
IRGAN	10	128	gen	100	1.00E-05	1.00E-06	0	0.0001	2	1	0	0	0.2	0.4363	inf	914	0.9721	0.9591
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

ELLIOT

THE REPRODUCIBILITY FRAMEWORK

TO TEST YOUR ADVERSARIAL MODEL

https://github.com/sisinflab/elliot