Course Project (IE506)
Prof. P Balamurugan

Removing Hidden Confounding in Recommendation: A Unified Multi-Task Learning Approach

TEAM Y

Anurag Singh Yadav(22b0617)
Vishal Kumar(22b0687)

Outline

Problem Setup

Selection bias and **hidden confounding** in Recommendation Systems

- We train a machine learning model that can **predict pCVR**—the probability that a user **converts (buys, subscribes, etc.) after clicking on an item**.

- The model's predicted conversion probability is:

$$\hat{r}_{u,i} = f(x_{u,i}, \theta_{\text{CVR}}) \in [0, 1]$$

where f is the prediction model with parameters θ_{CVR}

- $r_{u,i}=1$ means user u converted after clicking item i.
- $r_{u,i}=0$ means user u did **not** convert after clicking item i.

Selection Bias:

- Users **only click on items they are interested in**, so we only observe **conversion outcomes for clicked items**.
- Since we **never see conversion data for unclicked items**, we cannot directly compute the ideal loss function across all user-item pairs.

Hidden Confounding

- Some factors affecting clicks and conversions are **measurable** (e.g., user age, item price) and included in $x_{u,i}$
- However, some **hidden (unmeasured) factors** $h_{u.i}$ also influence the results. These could be:
    - **Unobservable user preferences** (e.g., private information like salary, browsing history).
    - **External influences** (e.g., friend recommendations).
- Since these hidden confounders **affect both clicks and conversions**, they introduce bias in the model.

Problem: If we train the pCVR model using only clicked items, the model might **overfit to the observed data** and not generalize well to all user-item pairs.Hidden confounders make it even harder to learn the **true** conversion probabilities, as we lack complete information.

Previous Methods Lead to Biased Learning under Hidden Confounding

Multi-Task Learning (MTL)

- The **post-click conversion rate (pCVR)** prediction task is closely related to other tasks:
  a. **Click-Through Rate (CTR):** Probability that a user **clicks** on an item.
  b. **Post-View Click-Through & Conversion Rate (CTCVR):** Probability that a user **clicks** and **then converts**.

CTCVR = CTR × pCVR

- The ESMM method learns pCVR by joint-training CTR and CTCVR losses

$$\mathcal{L}_{\text{CTR}}(\theta_{\text{CTR}}) = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \delta\left(o_{u,i}, \hat{p}_{u,i}\right), \quad \mathcal{L}_{\text{CTCVR}}(\theta_{\text{CTR}}, \theta_{\text{CVR}}) = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \delta\left(o_{u,i}r_{u,i}, \hat{p}_{u,i}\hat{r}_{u,i}\right).$$

- The ESMM method optimizes the sum of these two losses
- To **reduce bias**, researchers have proposed two improved methods: **IPS (Inverse Propensity Scoring)** and **DR (Doubly Robust)**.

$$\mathcal{L}_{\text{IPS}}(\theta_{\text{CTR}}, \theta_{\text{CVR}}) = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \frac{o_{u,i}\delta_{u,i}}{\hat{p}_{u,i}}, \; \mathcal{L}_{\text{DR}}(\theta_{\text{CTR}}, \theta_{\text{CVR}}, \theta_{\text{IMP}}) = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \left[\hat{\delta}_{u,i} + \frac{o_{u,i}(\delta_{u,i} - \hat{\delta}_{u,i})}{\hat{p}_{u,i}}\right]$$

- L_{IPS} is an unbiased estimator of the ideal loss when the learned propensities are accurate, i.e., $\hat{p}_{u,i} = p_{u,i}$ and L_{DR} is unbiased if either $\hat{p}_{u,i} = p_{u,i}$ or $\hat{\delta}_{u,i} = g_{u,i}$

  However, both IPS and DR are biased under hidden confounders.

Debiasing with a Few Unbiased Ratings

- Earlier methods like **IPS, DR, and ESCM²** try to correct bias in pCVR prediction but still **fail under hidden confounders**.
- One way to fix this is to **use some unbiased data** (data not affected by selection bias) **along with the biased dataset**.

Several methods follow this idea, including:

1. **Learning to Debias (LTD)** – Uses **bi-level optimization** to adjust the model.
2. **AutoDebias** – A more advanced version of LTD that aims for **universal debiasing**.
3. **Causal Embedding Method (CausE)** – Adjusts item embeddings to reduce bias.
4. **Knowledge Distillation Framework (KDCRec)** – Uses **uniform data** (random samples) to counter bias.
5. **Causal Balancing Methods** – Modify training weights to neutralize bias.

Bi-Level Optimization?

Bilevel optimization is used to **reduce bias** in recommendation systems by combining **a small unbiased dataset (U)** with **a large biased dataset (B)**. This is important because the biased dataset (B) only contains data from clicked items, leading to **selection bias**.

Bi-level optimization is a technique where **two models are trained at the same time**:

- The **upper-level model** (CTR model) is trained using **unbiased data U**.
- The **lower-level model** (pCVR model) is trained using **biased data B**.

Debiasing with a Few Unbiased Ratings

Lower-level optimization (Training on Biased Data - B)

- We train a model using the **biased dataset (B)** to predict how likely a user is to convert (**pCVR prediction**).
- This helps us learn from the large dataset, but since B is biased, the results won't be perfect.
- The goal here is to minimize the **bias-adjusted loss**:

$$\theta^*_{\mathrm{CVR}}(\theta_{\mathrm{CTR}}) = \arg\min_{\theta_{\mathrm{CVR}}} \mathcal{L}^{\mathcal{B}}_{\mathrm{CVR}}(\theta_{\mathrm{CTR}}, \theta_{\mathrm{CVR}})$$

- This equation means we find the best model parameters θ_{CVR} by minimizing the loss on the **biased dataset (B)**.

Upper-level optimization (Fine-tuning with Unbiased Data - U)

- Since the lower-level model was trained on biased data, it may still contain errors.
- Now, we **fine-tune** the model using the **small unbiased dataset (U)** to correct these errors.
- The goal here is to minimize the loss on the **unbiased dataset (U)**:

$$\theta^*_{\mathrm{CTR}} = \arg\min_{\theta_{\mathrm{CTR}}} \mathcal{L}^{\mathcal{U}}_{\mathrm{CVR}}(\theta^*_{\mathrm{CVR}}(\theta_{\mathrm{CTR}}))$$

- This equation means we adjust θ_{CTR} so that the model performs well on the **unbiased dataset (U)**.
- Essentially, we are choosing **the best CTR model** based on how well it helps in **pCVR prediction on unbiased data**.

Where….

$$\mathcal{L}^{\mathcal{U}}_{\mathrm{CVR}}(\theta^*_{\mathrm{CVR}}(\theta_{\mathrm{CTR}})) = \frac{1}{|\mathcal{U}|} \sum_{(u,i)\in\mathcal{U}} \delta\left(r_{u,i}, \hat{r}_{u,i}(\theta^*_{\mathrm{CVR}}(\theta_{\mathrm{CTR}}))\right)$$

Debiasing with a Few Unbiased Ratings

Even though **LTD and AutoDebias** try to correct bias using **unbiased data**, **they are still biased under hidden confounders**

Because LTD and AutoDebias still have limitations, researchers explored **new methods** that use unbiased data in a different way.

- CausE (Causal Embedding Method)
- KDCRec (Knowledge Distillation for Counterfactual Recommendation)

CausE (Causal Embedding Method)

- Builds a connection between **a biased model and an unbiased model**.
- Adds an **alignment term** (a mathematical way to compare the two models).
- The goal is to make the two models **as similar as possible** while still using both biased and unbiased data.

KDCRec (Knowledge Distillation for Counterfactual Recommendation)

- Uses **knowledge distillation** (a technique where one model teaches another).
- Transfers knowledge from a **biased model** to an **unbiased model** using:
  - **Label-based distillation** – Learning from final predictions.
  - **Feature-based distillation** – Using specific data characteristics.
  - **Sample-based distillation** – Training on selected data points.
  - **Model structure-based distillation** – Adjusting how the model learns.

But, There is no theoretical guarantee that these methods fully correct hidden confounding.

Hidden Confounding with Sensitivity Analysis

The **Robust Deconfounder (RD) method** tries to fix this problem using **Sensitivity Analysis** from **causal inference**. The idea is simple:

- **We don't know the true probability of clicking** because of hidden confounders.
- Instead, we assume that the true probability is **somewhere close** to the estimated probability (within a range).
- We **minimize the worst-case prediction error**, ensuring our model is robust even if we made mistakes due to hidden confounders.

The real probability of a user clicking on an item (**true propensity**) is:

Or The probability of clicking, given user-item features $x_{u,i}$ and the **hidden confounders** $h_{u,i}$.

$$\bar{p}_{u,i} \triangleq \mathbb{P}(o_{u,i} = 1 | x_{u,i}, h_{u,i})$$

But since we **don't know** the hidden confounders, we only estimate the probability using **known features**:

Nominal propensity, meaning "the probability of clicking, based only on observable data.

$$p_{u,i} = \mathbb{P}(o_{u,i} = 1 | x_{u,i})$$

Since our estimate $p_{u,i}$ is not 100% accurate, RD assumes that the **true probability is close to our estimated probability** but within a **certain range** controlled by a parameter Γ(Gamma).

This creates an **upper bound** and a **lower bound** for the inverse of the true propensity: $a_{u,i} \leq w_{u,i} \leq b_{u,i}$

- $w_{u,i} = 1/\bar{p}_{u,i}$ (inverse of the true probability)
- **Upper bound** $b_{u,i}$ and **lower bound** $a_{u,i}$ are estimated using a parameter Γ, which controls how much hidden confounding we think is present.
- If Γ is **large**, we assume there is **more hidden bias**.
- If Γ is **small**, we assume there is **less hidden bias**.

Hidden Confounding with Sensitivity Analysis

So, Mathematically $$\frac{1}{\Gamma} \leq \frac{(1-p_{u,i})\,\bar{p}_{u,i}}{p_{u,i}\,(1-\bar{p}_{u,i})} \leq \Gamma$$

This means:

- We **don't assume** we know the exact true propensity, but we assume it lies **within a reasonable range**.
- The values **$a_{u,i}$ and $b_{u,i}$ define this range**.

Since we have uncertainty about the **true propensity**, we **minimize the worst-case error** using a modified **IPS method** called **RD-IPS**.

- **IPS normally re-weights** data based on the observed propensity score $p_{u,i}$ but it fails if hidden confounders exist.
- **RD-IPS instead uses the worst-case bound** ($a_{u,i}$ to $b_{u,i}$) to make sure the model performs well even in the worst possible scenario.

$$\mathcal{L}_{\text{RD-IPS}}(\theta_{\text{CTR}}, \theta_{\text{CVR}}) = \max_{W \in \mathcal{W}} \frac{1}{|\mathcal{D}|} \sum_{(u,i) \in \mathcal{D}} o_{u,i} \delta_{u,i} w_{u,i},$$

This ensures that **our model is robust**, meaning it performs well even if there is some hidden bias in the data.

- Just like **RD-IPS corrects the IPS method**, we can also correct the **DR (Doubly Robust) method** in a similar way.
- This would involve minimizing the **worst-case DR loss** rather than the worst-case IPS loss.

Challenges & Limitations of This Approach:

- The strength of hidden confounding is **unknown**, so picking the correct Γ is difficult.If we choose **too high** a value, we may make the model overly conservative and If we choose **too low** value, we may still have hidden bias.
- The assumptions might be wrong, RD assumes that the **true propensity** is **close to the observed one** within the range we estimate.If this assumption is incorrect, then RD methods **might not actually remove the bias**.

Hidden Confounding in Recommendation systems

The paper addresses the issue of **hidden confounding** in recommendation systems, which arises when unobserved factors influence both user preferences and item exposure, leading to biased recommendations.

Traditional recommendation models often assume that observed user-item interactions fully capture user preferences, but hidden confounders (e.g., social influence, marketing bias) distort this relationship.

The paper proposes a **unified multi-task learning approach with residual network as in figure 1** to mitigate hidden confounding effects and improve recommendation fairness and accuracy.

Mathematical Formulation

Let R(ui) denote user u's rating of item i. The observed rating depends on the user's true preference P(ui) and hidden confounders C(ui):

$$R(ui)=P(ui)+C(ui)+\epsilon$$

where ϵ is noise.

The proposed multi-task learning framework estimates both **user preference** and **hidden confounders .**

Example-Music Streaming App

A user listens to a lot of pop songs on a music app. The system thinks pop is their favorite genre, but in reality, they were only listening to it because **their friends created a shared playlist** (a hidden confounder).

Without fixing this bias, the app keeps recommending only pop songs, ignoring the fact that the user might actually enjoy jazz or rock as well. The proposed method helps adjust recommendations by identifying such hidden influences.

Residual Networks (ResNets) - A Solution to Deep Learning Challenges

Key Problem:

- As neural networks grow deeper, they suffer from the **vanishing gradient problem** and **degradation issue** (increased depth does not always improve performance).
- Deeper networks become difficult to train because learning a direct mapping becomes complex.

This diagram indicates it is not an overfitting problem as

adding more layers to a suitably deep model leads to higher

training error, as reported in figure 1.

Solution: Residual Learning

Instead of learning a direct mapping **H(x)**, ResNets learn the **residual**

function: **F(x)=H(x)−x**

which simplifies to: **H(x)=F(x)+x**

This allows the model to **focus on learning the difference (residual)** between

the input and output, making it easier to optimize.

How It Works:

- **Shortcut Connections** (also called "skip connections") allow information to

  bypass layers, ensuring that gradients flow smoothly during backpropagation.

- Identity mappings help retain information without modifying it unnecessarily.
- Stacking multiple residual blocks allows the network to scale effectively to hundreds of layers.

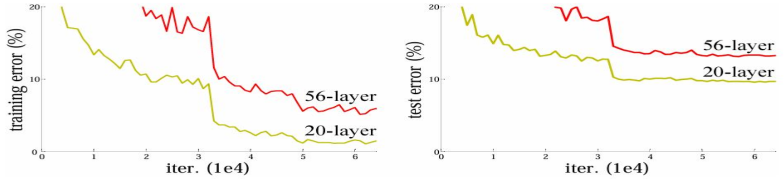


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

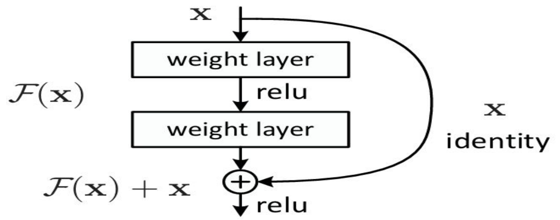


Figure 2. Residual learning: a building block.

How Does the Paper Address the Problem?

We define the **true propensity-** $\bar{p}_{u,i} \triangleq \mathbb{P}(o_{u,i} = 1 | x_{u,i}, h_{u,i})$ and **true imputation-** $\bar{g}_{u,i} \triangleq \mathbb{E}(\delta_{u,i} | x_{u,i}, h_{u,i})$, both of them are functions of $(x_{u,i}, h_{u,i})$, with $\tilde{p}_{u,i}$ and $\tilde{\delta}_{u,i}$ as their estimates.

To distinguish, we call $p_{u,i} = \mathbb{P}(o_{u,i} = 1 | x_{u,i})$ and $g_{u,i} = \mathbb{E}[\delta_{u,i} | x_{u,i}]$ the **nominal propensity** and **nominal imputation**, with $\hat{p}_{u,i}$ and $\hat{\delta}_{u,i}$ as their estimates.

Next, we show the necessity of calibrations on both $\hat{p}_{u,i}$ and $\hat{\delta}_{u,i}$ estimated from the baised data B.

Theorem 2 (Necessity of Calibration). *Suppose the partial derivative of $\bar{p}_{u,i}$ and $\bar{g}_{u,i}$ with respect to hidden confounders $h_{u,i}$ are not always equal to 0, and $\hat{p}_{u,i}$ and $\hat{\delta}_{u,i}$ are consistent estimators of $p_{u,i}$ and $g_{u,i}$, then there exists $\eta > 0$, such that*

$$\lim_{|\mathcal{D}| \to \infty} \mathbb{P}(|\hat{p}_{u,i} - \bar{p}_{u,i}| > \eta) > 0, \qquad \lim_{|\mathcal{D}| \to \infty} \mathbb{P}(|\hat{\delta}_{u,i} - \bar{g}_{u,i}| > \eta) > 0.$$

Theorem 2 shows that in the presence of hidden confounding, the estimated nominal propensities and nominal imputed errors deviate from the true one, even with the infinite sample size. To address this problem, as shown in Figure 1, we propose a novel consistency loss that utilizes unbiased data to calibrate the learned nominal propensities and imputed errors from the biased data.

How Does the Paper Address the Problem?

This figure illustrates two proposed debiasing residual networks designed to remove hidden confounding in machine learning models for click-through rate (CTR) and conversion rate (CVR) prediction. The two subfigures represent different methods:

(a) **Res-IPS (Inverse Propensity Scoring)**

The model has a **shared MLP network** that takes **input covariates** and feeds them to two sub-networks:

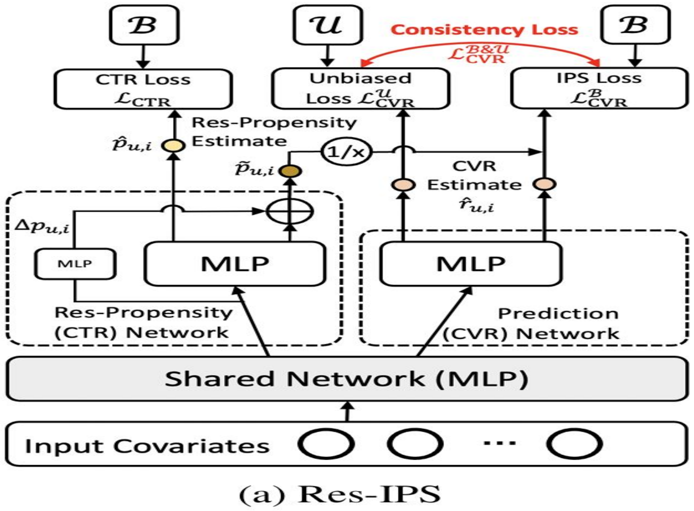
1.**Res-Propensity (CTR) Network**: Estimates the residual-propensity $\tilde{p}_{u,i}$ for user uuu and item i.

2.**Prediction (CVR) Network**: Predicts the CVR estimate $\hat{r}_{u,i}$ which is adjusted using inverse propensity weighting $\frac{1}{\tilde{p}_{u,i}}$

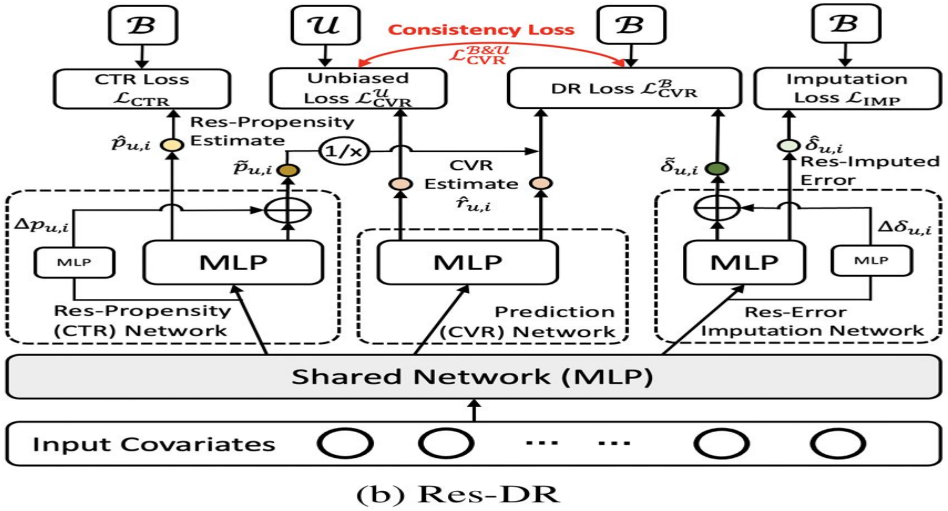
The model optimizes different loss functions:

1. $\mathcal{L}_{\text{CTR}}$ (CTR loss) from observed biased data B. 2. $\mathcal{L}_{\text{CVR}}^{\mathcal{U}}$ unbiased loss) from unbiased data U.

3. $\mathcal{L}_{\text{CVR}}^{\mathcal{B}}$ (IPS loss) for debiasing 4.**Consistency Loss** $\mathcal{L}_{\text{CVR}}^{\mathcal{B}\&\mathcal{U}}$ ensures consistency between unbiased and biased estimates.



$\mathcal{B}$
$\mathcal{U}$
Consistency Loss
$\mathcal{L}_{\text{CVR}}^{\mathcal{B}\&\mathcal{U}}$
$\mathcal{B}$
CTR Loss $\mathcal{L}_{\text{CTR}}$
Unbiased Loss $\mathcal{L}_{\text{CVR}}^{\mathcal{U}}$
IPS Loss $\mathcal{L}_{\text{CVR}}^{\mathcal{B}}$
$\hat{p}_{u,i}$
Res-Propensity Estimate
1/x
$\tilde{p}_{u,i}$
CVR Estimate $\hat{r}_{u,i}$
$\Delta p_{u,i}$
MLP
MLP
MLP
Res-Propensity (CTR) Network
Prediction (CVR) Network
Shared Network (MLP)
Input Covariates
(a) Res-IPS



$\mathcal{B}$
$\mathcal{U}$
Consistency Loss
$\mathcal{L}_{\text{CVR}}^{\mathcal{B}\&\mathcal{U}}$
$\mathcal{B}$
$\mathcal{B}$
CTR Loss $\mathcal{L}_{\text{CTR}}$
Unbiased Loss $\mathcal{L}_{\text{CVR}}^{\mathcal{U}}$
DR Loss $\mathcal{L}_{\text{CVR}}^{\mathcal{B}}$
Imputation Loss $\mathcal{L}_{\text{IMP}}$
$\hat{p}_{u,i}$
Res-Propensity Estimate
1/x
$\tilde{p}_{u,i}$
CVR Estimate $\hat{r}_{u,i}$
$\tilde{\delta}_{u,i}$
$\hat{\delta}_{u,i}$
Res-Imputed Error
$\Delta p_{u,i}$
$\Delta \delta_{u,i}$
MLP
MLP
MLP
MLP
MLP
Res-Propensity (CTR) Network
Prediction (CVR) Network
Res-Error Imputation Network
Shared Network (MLP)
Input Covariates
(b) Res-DR

Figure 1: Proposed debiasing residual networks for removing hidden confounding.

How Does the Paper Address the Problem?

(b) **Res-DR (Doubly Robust)**

1.This model builds upon Res-IPS by incorporating **error imputation**.

2.Additional **Res-Error Imputation Network** estimates the residual-imputed error $\ddot{\delta}_{u,i}$

3.The model uses an additional **Imputation Loss** $\mathcal{L}_{\mathbf{IMP}}$ to correct biases.

4.The DR loss $\mathcal{L}_{\text{CVR}}^{\mathcal{B}}$ combines the IPS-based CVR estimate with the imputed residual error to further remove hidden confounding.

Key Differences

1.Res-IPS only uses inverse propensity weighting for debiasing.

2.Res-DR improves on Res-IPS by adding residual error imputation for a more robust debiasing approach.

Key Components of the Solution:

1. **Calibrated Propensity & Imputation Models**
   - Instead of using only **biased estimations** of propensity scores $\hat{p}_{u,i}$ and imputation errors $\hat{\delta}_{u,i}$ the model **learns correction factors** to improve accuracy.
2. **Residual Networks (Res-IPS & Res-DR)**
   - Two separate residual networks are used:
     - One for **propensity correction** $\Delta p_{u,i}$
     - One for **imputation correction** $\Delta\delta_{u,i}$
3. **Consistency Loss for Bias Correction**
   - The model learns residuals by comparing the biased loss with the unbiased loss.
4. **Multi-Task Learning Framework**
   - Uses a **small unbiased dataset** to fine-tune residual corrections and improve **debiasing performance**.

Mathematical Formulation & Equations

True Propensity & True Imputation

$$\bar{p}_{u,i} \triangleq \mathbb{P}(o_{u,i} \underset{\sim}{=} 1 | x_{u,i}, h_{u,i}), \ \bar{g}_{u,i} \triangleq \mathbb{E}(\delta_{u,i} | x_{u,i}, h_{u,i})$$

Residual Calibration Model

$$\tilde{p}_{u,i} = \sigma\left(\sigma^{-1}(\hat{p}_{u,i}(\theta_{\text{CTR}})) + \sigma^{-1}(\Delta p_{u,i}(\phi_{\text{CTR}}))\right),$$
$$\tilde{\delta}_{u,i} = \sigma\left(\sigma^{-1}(\hat{\delta}_{u,i}(\theta_{\text{IMP}})) + \sigma^{-1}(\Delta \delta_{u,i}(\phi_{\text{IMP}}))\right),$$

Loss Function for Learning Residuals

$$\mathcal{L}_{\text{Res}} = \underbrace{\mathcal{L}_{\text{CTR}}(\hat{p}) + \alpha \cdot \mathcal{L}_{\text{IMP}}(\hat{\delta})}_{\text{Initialization using biased data}} + \beta \cdot \underbrace{\left(\mathcal{L}^{\mathcal{B}}_{\text{CVR}}(\tilde{p}, \tilde{\delta}) + \mathcal{L}^{\mathcal{B}}_{\text{CTCVR}}(\tilde{p}) + \mathcal{L}^{\mathcal{U}}_{\text{CVR}}\right)}_{\text{Prediction model training with calibrated losses}} + \gamma \cdot \underbrace{\mathcal{L}^{\mathcal{B}\&\mathcal{U}}_{\text{CVR}}(\hat{p}, \tilde{p}, \hat{\delta}, \tilde{\delta})}_{\text{Calibration on } \hat{p} \text{ and } \hat{\delta}},$$

where σ is the sigmoid function, and the transformations are designed for numerical stability, e.g., to control range from 0 to 1. Compared with $\hat{p}_{u,i}$ and $\hat{\delta}_{u,i}$ adopted in the previous multi-task learning approaches , the residual terms $\Delta p_{u,i}$ and $\Delta \delta_{u,i}$ are further added to $\tilde{p}_{u,i}$ and $\tilde{\delta}_{u,i}$ to capture the effect of hidden confounding.

where α, β and γ are hyper-parameters for trade-off.

Algorithm (Residual Debiasing) /Debiased Prediction Model Training

Step 1: Train the **nominal propensity model** $\hat{p}_{u,i}$ and **imputation model** $\hat{\delta}_{u,i}$ using biased data.

Step 2: Train **residual networks** to learn the corrections $\Delta p_{u,i}$ and $\Delta\delta_{u,i}$

Step 3: Compute the **calibrated values** $\tilde{p}_{u,i}$ and $\tilde{\delta}_{u,i}$

Step 4: Train the final **recommendation model** using:

1.Residual-IPS Loss (Res-IPS):

$$\mathcal{L}_{\text{IPS}}^{\mathcal{B}}(\theta_{\text{CVR}} \mid \hat{p}_{u,i}(\theta_{\text{CTR}}), \Delta p_{u,i}(\phi_{\text{CTR}})) = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \frac{o_{u,i} \cdot \delta_{u,i}}{\tilde{p}_{u,i}}.$$

2.Residual-DR Loss (Res-DR):

$$\mathcal{L}_{\text{DR}}^{\mathcal{B}}(\theta_{\text{CVR}} \mid \hat{p}_{u,i}(\theta_{\text{CTR}}), \Delta p_{u,i}(\phi_{\text{CTR}}), \hat{\delta}_{u,i}(\theta_{\text{IMP}}), \Delta\delta_{u,i}(\phi_{\text{IMP}})) = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \left[\tilde{\delta}_{u,i} + \frac{o_{u,i}(\delta_{u,i} - \tilde{\delta}_{u,i})}{\tilde{p}_{u,i}}\right].$$

3. Calibrated CTCVR Loss:

$$\mathcal{L}_{\text{CTCVR}}(\theta_{\text{CVR}} \mid \hat{p}_{u,i}(\theta_{\text{CTR}}), \Delta p_{u,i}(\phi_{\text{CTR}})) = \frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \delta\left(o_{u,i} r_{u,i}, \tilde{p}_{u,i} \hat{r}_{u,i}\right).$$

Step 5: Minimize **Consistency Loss:**

$$\mathcal{L}_{\text{CVR}}^{\mathcal{B}\&\mathcal{U}}(\theta_{\text{CTR}}, \phi_{\text{CTR}}, \theta_{\text{CVR}}, \theta_{\text{IMP}}, \phi_{\text{IMP}}) = \delta(\mathcal{L}_{\text{CVR}}^{\mathcal{B}}, \mathcal{L}_{\text{CVR}}^{\mathcal{U}}),$$

-**Unbiased data helps correct biased predictions.**

-**Residual Networks learn corrections dynamically**, making recommendations fairer

-**Consistency Loss** ensures that biased models align with real user preferences.

Debiased Prediction Model Training

Why Is Training on Biased Data a Problem?

1. **Nominal propensity scores** $\hat{p}_{u,i}$ and **nominal imputed errors** $\hat{\delta}_{u,i}$ are trained on **biased data**, meaning they do not fully capture hidden confounders.

2. If we **directly use these biased values**, the prediction model will continue making **biased recommendations**.

Instead of using raw biased estimates, the approach **calibrates them** by introducing **residual corrections** $\Delta p_{u,i}$ and $\Delta \delta_{u,i}$

The model can be trained using **three different loss functions**, all incorporating calibrated values to reduce bias as discussed in **step 4 .**

While **biased data needs correction**, **unbiased data** provides the **golden standard** for evaluation.

The **unbiased loss function** is:

$$\mathcal{L}_{\mathrm{CVR}}^{\mathcal{U}}(\theta_{\mathrm{CVR}}) = \frac{1}{|\mathcal{U}|} \sum_{(u,i)\in\mathcal{U}} \delta\left(r_{u,i}, \hat{r}_{u,i}\left(\theta_{\mathrm{CVR}}\right)\right).$$

Where: U = Set of **unbiased user-item pairs** , $r_{u,i}$ = **Actual user rating or conversion label** , , $\hat{r}_{u,i}\left(\theta_{\mathrm{CVR}}\right)$ = **Predicted conversion rate.**

Experimental Setup

Three datasets were used in this paper

1.**COAT:** 6,960 biased ratings, 4,640 unbiased ratings.

2.**Yahoo! R3:** 311,704 biased ratings, 54,000 unbiased ratings.

3.**KuaiRec:** A large-scale industrial dataset with 4,676,570 video records.

Dataset Details:

- **COAT:** Users rate 24 items based on preference (biased) + 16 randomly assigned items (unbiased).
- **Yahoo! R3:** First 5,400 users rate 10 randomly selected items (unbiased).
- **KuaiRec:** Fully exposed dataset with users rating videos.

Datasets & Preprocessing

Ratings are binarized:

- Scores < 4 → 0
- Scores ≥ 4 → 1

Filtering applied to remove inactive users/items.

Evaluation Metrics:

- AUC (Area Under the Curve)
- NDCG@K (Normalized Discounted Cumulative Gain)
- Recall@K

Computational Setup & Programming Details

Hardware Setup:

- **GPU Used:** Tesla T4
- **Software:**
  - **Programming Language:** Python
  - **Frameworks & Libraries:** PyTorch, NumPy, Scikit-learn

Hyperparameter Tuning:

- **Learning rate:** {0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05}
- **Weight decay**: {0, 1e−6, ..., 1e−1}
- Trade-off parameters **(α, β, γ)** tuned for optimal performance.
- **α** in {0.1, 0.5, 1}, **β** in {0.1, 0.5, 1, 5, 10}, and **γ** in {0.001, 0.005, 0.01, 0.05, 0.1}.

In paper K = 5 for COAT and YAHOO! R3, and K = 50 for KUAIREC.

All the experiments are implemented on **Pytorch** with **Adam** as the optimizer.

In addition, we randomly split **5%** of unbiased data from the test set to train models for all methods that require unbiased data.

Comparison with Competing Methods

Baseline Methods Compare:

1.Matrix Factorization (MF)

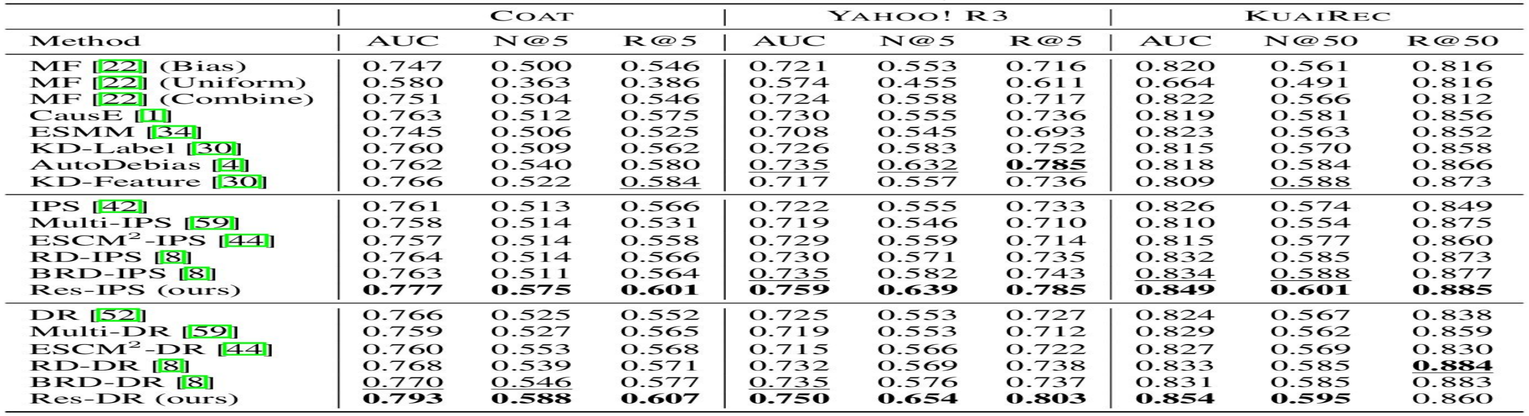
2.Inverse Propensity Scoring (IPS), Doubly Robust (DR)

3.Sensitivity Analysis-based RD-IPS and RD-DR

4.Multi-task learning-based methods (ESMM, Multi-IPS, Multi-DR, ESCM2-IPS, ESCM2-DR)

5.Knowledge Distillation-based Methods (CausE, KD-Label, KD-Feature, AutoDebias

Table 1: Performance in terms of AUC, NDCG@K, and Recall@K on the unbiased dataset of COAT, YAHOO! R3 and KUAIREC. The best two results are bolded, and the best baseline is underlined.



	COAT			YAHOO! R3			KUAIREC		
Method	AUC	N@5	R@5	AUC	N@5	R@5	AUC	N@50	R@50
MF [22] (Bias)	0.747	0.500	0.546	0.721	0.553	0.716	0.820	0.561	0.816
MF [22] (Uniform)	0.580	0.363	0.386	0.574	0.455	0.611	0.664	0.491	0.816
MF [22] (Combine)	0.751	0.504	0.546	0.724	0.558	0.717	0.822	0.566	0.812
CausE [1]	0.763	0.512	0.575	0.730	0.555	0.736	0.819	0.581	0.856
ESMM [34]	0.745	0.506	0.525	0.708	0.545	0.693	0.823	0.563	0.852
KD-Label [30]	0.760	0.509	0.562	0.726	0.583	0.752	0.815	0.570	0.858
AutoDebias [4]	0.762	0.540	0.580	0.735	0.632	**0.785**	0.818	0.584	0.866
KD-Feature [30]	0.766	0.522	0.584	0.717	0.557	0.736	0.809	0.588	0.873
IPS [42]	0.761	0.513	0.566	0.722	0.555	0.733	0.826	0.574	0.849
Multi-IPS [59]	0.758	0.514	0.531	0.719	0.546	0.710	0.810	0.554	0.875
ESCM^2-IPS [44]	0.757	0.514	0.558	0.729	0.559	0.714	0.815	0.577	0.860
RD-IPS [8]	0.764	0.514	0.566	0.730	0.571	0.735	0.832	0.585	0.873
BRD-IPS [8]	0.763	0.511	0.564	0.735	0.582	0.743	0.834	0.588	0.877
Res-IPS (ours)	**0.777**	**0.575**	**0.601**	**0.759**	**0.639**	**0.785**	**0.849**	**0.601**	**0.885**
DR [52]	0.766	0.525	0.552	0.725	0.553	0.727	0.824	0.567	0.838
Multi-DR [59]	0.759	0.527	0.565	0.719	0.553	0.712	0.829	0.562	0.859
ESCM^2-DR [44]	0.760	0.553	0.568	0.715	0.566	0.722	0.827	0.569	0.830
RD-DR [8]	0.768	0.539	0.571	0.732	0.569	0.738	0.833	0.585	**0.884**
BRD-DR [8]	0.770	0.546	0.577	0.735	0.576	0.737	0.831	0.585	0.883
Res-DR (ours)	**0.793**	**0.588**	**0.607**	**0.750**	**0.654**	**0.803**	**0.854**	**0.595**	0.860

Why Compare?

To evaluate if **Res-IPS and Res-DR** can effectively remove hidden confounding.

Comparison with Competing Methods

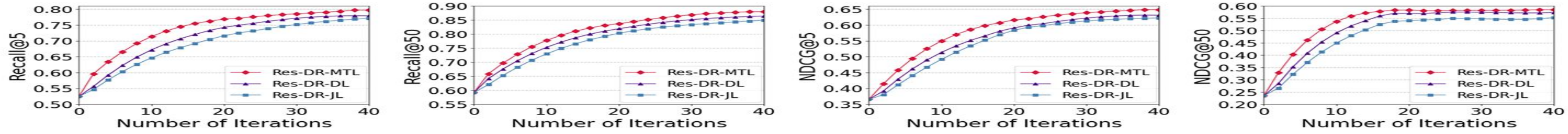
1. What is Recall@5?

Measures the proportion of relevant items correctly identified within the top 5 recommendations out of the total number of relevant items.

2.What is AUC (Area Under the Curve)?

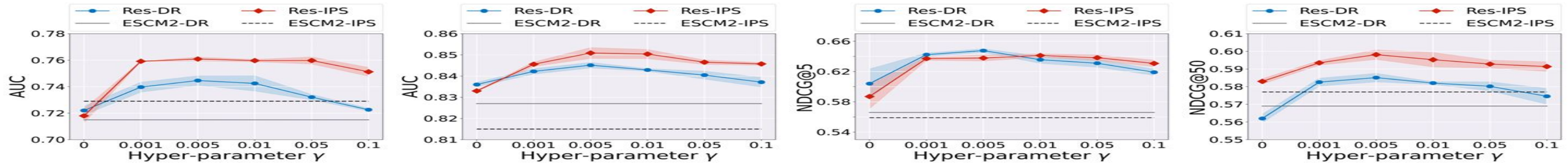
AUC=P(relevant item ranked higher than irrelevant item)

It computes the probability that a randomly chosen relevant item is ranked higher than a randomly chosen irrelevant item.



(a) R@5 on Yahoo! R3 (b) R@50 on KuaiRec (c) N@5 on Yahoo! R3 (d) N@50 on KuaiRec

Figure 2: Joint learning (JL), double learning (DL), and multi-task learning (MTL) on Res-DR.



(a) AUC on Yahoo! R3 (b) AUC on KuaiRec (c) N@5 on Yahoo! R3 (d) N@50 on KuaiRec

Figure 3: Effects of varying weights γ of the consistency loss $\mathcal{L}_{\mathrm{CVR}}^{\mathcal{B}\&\mathcal{U}}$ on Res-IPS and Res-DR.

List of references

1.**Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun**. *Deep Residual Learning for Image Recognition*, **Conference on Computer Vision and Pattern Recognition (CVPR)**, 2015. https://arxiv.org/abs/1512.03385

2.**Yu Tian, Yuhao Yang, Xudong Ren, Pengfei Wang, Fangzhao Wu, Qian Wang, and Chenliang Li**. *Joint Knowledge Pruning and Recurrent Graph Convolution for News Recommendation*, **SIGIR Conference**, 2021. https://doi.org/10.1145/3404835.3462912

3.**Yiling Jia and Hongning Wang**. *Scalable Exploration for Neural Online Learning to Rank with Perturbed Feedback*, **SIGIR Conference**, 2022. https://doi.org/10.1145/3477495.3532057

Youtube videos

1.Residual Network and Skip Connections (DL15) by Professor Bryce
https://www.youtube.com/watch?v=Q1JCrG1bJ-A&t=623s&ab_channel=ProfessorBryce

2.Session on ResNets | Batch 6 | 27 Nov 2020 by CampusX
https://www.youtube.com/watch?v=B1O1Ilh1F5E&list=PLKnIA16_Rmvb6SAVpWjm0ZYPHo_KRXIjU&index=5&ab_channel=CampusX

Usage of AI Tools

AI Tools Used: ChatGPT, DeepSeek, Perplexity

Tasks Performed:

1.Summarizing research papers

2.Clarifying technical terminologies

3.Generating simplified explanations for complex concepts

4.Assisting in structuring references and report writing