E-HyperGCN+ : E-commerce Return Prediction on HyperGraph based Graph Convolutional Networks with Clustering

Sungwoo Cho
Graduate School of AI, KAIST
Seoul, South Korea
peter8526@kaist.ac.kr

Hyogon Ryu
Graduate School of AI, KAIST
Seoul, South Korea
hyogon.ryu@kaist.ac.kr

Seojeong Park
Graduate School of AI, KAIST
Seoul, South Korea
seojeong.park@kaist.ac.kr

ABSTRACT

In the e-commerce market, predicting product returns is crucial for companies selling products, as it can significantly impact their profitability. Especially when returns occur, customers may opt to return individual products from their order or return the entire order altogether. To address this scenario, we design the problem into two stages: return prediction at the order level and the product level within the order. For each level of prediction, we propose an efficient hypergraph-based algorithm called *E-HyperGCN+*, which allows customers to organize products and orders effectively. Furthermore, we introduce a method incorporating Graph Convolutional Networks (GCN), the most prominent methodology for understanding graph representations, into hypergraph. Additionally, we utilize multi-hot encoding-based K-mean clustering to design feature vectors for individual nodes in the hypergraph, aiming to create a hypergraph with high-quality embedding features.

1 INTRODUCTION

In recent years, the rapid growth of the e-commerce market has made online transactions one of the most common ways for consumers to purchase products. Large-scale marketplaces such as Amazon and AliExpress have emerged, and companies that traditionally operated offline have also established their online presence. However, a significant issue with online shopping is that consumers cannot physically inspect the products before purchasing. As a result, they often return items if they are dissatisfied after receiving them. The return rate in e-commerce typically ranges from 20% to 30%[1].

Processing returned items incurs additional costs for reprocessing and resale. Moreover, various indirect costs, such as shipping, restocking, and reprocessing, are also involved in the return process. To address this issue, many companies strive to reduce return rates through various strategies. For instance, they offer discount coupons for products with high return rates and provide accurate size and color guides to customers. However, to maximize the effectiveness of such strategies, it is crucial to accurately predict product returns even before the order is placed. This project employs data mining and machine learning techniques to conduct E-commerce Product Return Prediction.

The project encompasses two main tasks, each focusing on different aspects of return predictions. Both tasks will utilize data about the products contained in shopping carts. The first task centers on predicting order returns, while the second task focuses on predicting the return of individual products within orders.

[1]https://www.shopify.com/za/enterprise/blog/ecommerce-returns

Task 1: Order return prediction. The objective of Task 1 is to predict the return status of entire orders. This classification task aims to distinguish whether a given order will result in all items being returned, some items being returned, or no items being returned.

Task 2: Product return prediction. The objective of Task 2 is to predict the return status of each individual product within an order. This classification task aims to determine whether each product in a given order will be returned or not. Further details regarding the dataset and experimental settings will be discussed in Section 4.

To address these task, we designed an algorithm based on hypergraph [1, 10] called *E-HyperGCN+*. This algorithm represents each product and order as nodes and edges in a hypergraph to solve the classification problem. In a hypergraph, an edge can connect multiple nodes, making it suitable for representing orders and the multiple products within those orders. We represented not only the products and orders but also customer information and detailed product information (such as color, size, and group) for each order as edges in the hypergraph. This comprehensive representation allows us to capture the complex relationships and interactions between various elements in the dataset, providing a rich structure for feature extraction and subsequent classification tasks.

To extract features from the hypergraph, we utilized clustering techniques and PCA. For embedding the hypergraph, a multi-hot method is typically required. However, due to the extremely large number of hyperedges in the given dataset, representing each edge with a multi-hot vector would result in prohibitively high computational costs. Therefore, we reduced the number of hyperedges through clustering and generated dense embeddings using PCA, enabling hyperGCN to process them effectively.

We then employed Graph Convolutional Networks (GCNs) to classify each node in the hypergraph. The GCN can effectively learn and propagate information through the hypergraph, allowing it to capture both local and global patterns in the data. This enables our model to make accurate predictions about the return status in product and basket level.

Our contributions can be summarized as follows:

- We propose a novel algorithm called *E-HyperGCN+* specifically designed for the e-commerce domain to predict product returns. This algorithm leverages hypergraphs to represent complex relationships between products, orders, and customers, providing a more comprehensive and intricate structure for analysis.
- We utilize clustering techniques and PCA to effectively reduce the number of hyperedges and generate dense embeddings, addressing the computational challenges associated with representing large-scale hypergraphs.

- We validate our proposed approach through extensive experiments using e-commerce data. Our results demonstrate the effectiveness of *E-HyperGCN+* in two tasks, outperforming other methods and highlighting the benefits of our hypergraph-based approach.

2 RELATED WORKS

2.1 E-tail Return Prediction

The rapid growth of e-commerce has led to a significant increase in product returns, posing substantial financial and logistical challenges for e-tailers. Several studies [4, 8, 11] have focused on predicting e-tail product returns to mitigate these challenges. For instance, HyperGo [8] proposes a framework for predicting customer return intentions by leveraging a novel hypergraph representation of historical purchase and return records. By identifying similar historical baskets using a local graph cut algorithm and truncated random walk, HyperGo can predict return rates on both basket-level and product-level aiding e-tailers in taking proactive measures to reduce returns and prepare for reverse logistics.

HyGraph [11] introduces a weighted hybrid graph model that integrates rich information from purchase and return histories to predict product returns. This model comprises customer and product nodes, along with undirected and directed edges reflecting return history and purchase behaviors. A random-walk-based local algorithm is employed, whose computational efficiency is enhanced by its dependency on the size of the output cluster rather than the entire graph, making it suitable for large-scale data sets.

[4] explores the use of customer reviews to understand and predict return reasons, which can help identify issues such as manufacturing defects or misleading product information. A BERT-based multi-class classifier is utilized to encode review text as features, significantly improving the precision of return reason prediction. This approach also demonstrates the potential of aggregated review information to predict product returns without direct customer feedback. These methods aim to achieve more accurate e-tail return predictions by utilizing additional information. However, they have not utilized easily accessible information in e-tail data, such as the product's category, size, and color. Our approach improves the accuracy of return prediction by leveraging the most common information about the product.

2.2 Hypergraph Neural Network

Hypergraphs provide a flexible data structure to model complex relationships and interactions within data, especially suitable for scenarios involving more than two entities. HyperGNN [6] presents a hypergraph neural networks (HGNN) framework for data representation learning, which encodes high-order data correlation in a hypergraph structure.

HyperGCN [9] introduces a novel graph convolutional network for semi-supervised learning (SSL) on attributed hypergraphs, which effectively assigns labels to initially unlabeled vertices in a hypergraph. HyperGCN demonstrates its effectiveness through detailed experimentation on real-world hypergraphs, showcasing its utility in modeling complex relationships in datasets such as co-authorship, co-citation, and email communication networks.

WhatsNet [5] addresses the problem of edge-dependent node labels in hypergraphs, which vary depending on the hyperedges they participate in. By modeling the relationships between nodes within each hyperedge and using relative centrality as positional encodings, WhatsNet performs well in various applications, including ranking aggregation, node clustering, and product return prediction. These advancements in hypergraph-based modeling and feature embedding highlight the potential of leveraging hypergraphs to improve the accuracy and efficiency of e-tail return prediction, addressing the scale and complexity of the data involved.

3 PROPOSED METHOD

In this section, we introduce the E-HyperGCN+ framework, a novel approach designed to predict e-commerce product returns using a GCN with the hypergraph-based representation of historical purchase and return data. This representation allows for a detailed and interconnected view of customer shopping behaviors, which is crucial for predicting return intentions more accurately at both the order and product levels. We modified and applied the graph Laplacian, traditionally used in conventional graphs, to hypergraphs in the existing GCN. The overall framework of E-HyperGCN+ is outlined in Algorithm 1, represented in pseudocode.

3.1 Notation

Let us denote V_L and the hypergraph as $\mathcal{H} = (V, E, w)$, where:

- V is the total nodes representing individual shopping orders.
- V_L is the nodes representing individual shopping orders for which labels are already provided.
- E is the set of hyperedges, each representing a unique product that connects nodes (orders) containing this product.
- X is the feature vector of each node in the hypergraph.

The hypergraph is further described by its $n \times m$ incidence matrix H_m, which establishes the connectivity between nodes and hyperedges. The matrix entry $H_m(v, e)$ is positive if node v is connected by hyperedge e, and zero otherwise. This setup allows us to define various metrics, such as the degree of a node, the degree of a hyperedge, and the volume of nodes, which are essential for understanding the complex relationships within the data.

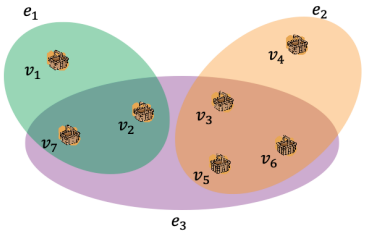
3.2 Hypergraph-based Representation

The hypergraph G effectively leverages the rich information embedded in historical purchase and return records. By representing orders as nodes and products as hyperedges that connect these nodes, the hypergraph captures not just pairwise relationships but complex interactions among multiple products within orders. This is particularly useful for identifying scenarios where orders contain similar or identical products, which may increase the likelihood of returns due to customer preferences or errors in order composition. Moreover, hypergraphs enable more accurate predictions by understanding the larger picture of an order, unlike typical graphs that connect only at the product level. In E-HyperGCN+, we constructed a hypergraph as shown in Figure 1. Each node represents an order, and the same hyperedge distinguishes orders containing one item in common.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
e_1	1	1	0	0	0	0	1
e_2	0	0	1	1	1	1	0
e_3	0	1	1	0	1	1	1

$$hypergraph = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{vmatrix}$$

(a)



(b)

Figure 1: An example of the E-HyperGCN+'s hypergraph built with 3 products $E = e_1, e_2, e_3$ and 7 shopping orders $V = v_1, v_2, v_3, v_4, v_5, v_6, v_7$. (a) The numbers in the table reflect the quantity of each product in a order. (b) The hypergraph is able to capture the complex relationship among products and shopping orders.

Algorithm 1 Algorithm for E-HyperGCN+

Input: An attributed hypergraph $\mathcal{H} = (V, E, X)$, with attributes X, a set of labelled vertices V_L
Output: All hypernodes in $V - V_L$ labelled
for each epoch τ of training **do**
  for layer $l = 1, 2$ of the network **do**
    set $A_{vv}^{(l)} = 1$ for all hypernodes $v \in V$
    let $\Theta = \Theta^t$ be the parameters for the current epoch
    for $e \in E$ **do**
      $H \leftarrow$ hidden representation matrix of layer $l - 1$
      $i_e, j_e \leftarrow \arg\max_{i,j \in e} \|H_i(\Theta^{(l)}) - H_j(\Theta^{(l)})\|_2$
      $A_{i_e j_e}^{(l)} = A_{j_e i_e}^{(l)} = \frac{1}{2|e|-3}$
      $K_e := \{k \in e : k \neq i_e, k \neq j_e\}$
      for $k \in K_e$ **do**
        $A_{i_e k}^{(l)} = A_{k i_e}^{(l)} = \frac{1}{2|e|-3}$
        $A_{j_e k}^{(l)} = A_{k j_e}^{(l)} = \frac{1}{2|e|-3}$
      end for
    end for
  end for
  $Z = \text{softmax}\left(A^{(2)} \text{ ReLU}\left(A^{(1)} X \Theta^{(1)}\right) \Theta^{(2)}\right)$
  update parameters Θ^t to minimize cross entropy loss on the set of labelled hypernodes V_L
end for
label the hypernodes in $V - V_L$ using Z

3.3 Two-stage Clustering for Large-scale Hypernode embedding

In this task, the given data contains an extremely large number of hyperedges. Therefore, simply forming node embeddings, as shown in Figure 1, results in highly dimensional embeddings, making it difficult for GCN to process. To address this, we clustered numerous hyperedges to group similar ones. After clustering, we represented hypernodes using multi-hot encoding and generated dense embeddings through PCA. The detailed method is as follows.

The clustering process is carried out in two stages. In stage 1, we first separate the components that need clustering and perform clustering based on data that do not require clustering. In stage 2, we conduct clustering once more using the entire dataset. The reason for performing clustering twice is to utilize the related information among edges with an extremely large number of entities, which is not considered in stage 1. After reducing each edge, we represent each hypernode in the form of multi-hot encoding. Subsequently, we used PCA to generate dense embeddings, which were then provided as inputs together with multi-hot encoding.

3.4 Graph Convolutional Networks on HyperGraph

We consider semi-supervised hypernode classification on an undirected hypergraph $\mathcal{H} = (V, E)$ with $|V| = n$, $|E| = m$, and a small set V_L of labeled hypernodes. Each hypernode $v \in V = \{1, \ldots, n\}$ is also associated with a feature vector $x_v \in \mathbb{R}^p$ of dimension p, given by $X \in \mathbb{R}^{n \times p}$. The task is to predict the labels of all the unlabelled hypernodes, i.e., all the hypernodes in the set $V \setminus V_L$.

The critical working principle is that hypernodes in the same hyperedge are similar and thus are likely to share the same label [7]. Suppose we use $\{h_v : v \in V\}$ to denote some representation of the hypernodes in V. Then, for any $e \in E$, the function $\max_{i,j \in e} \|h_i - h_j\|_2$ will be *small* only if the vectors corresponding to the hypernodes in e are *close* to each other. Therefore, $\sum_{e \in E} \max_{i,j \in e} \|h_i - h_j\|_2$ as a regularizer is likely to ensure that hypernodes in the same hyperedge have similar representations. However, instead of using it as an explicit regularizer, we can achieve the same goal by using a GCN over an appropriately defined Laplacian of the hypergraph. In other words, we use the concept of hypergraph Laplacian as an implicit regularizer to achieve this objective.

A hypergraph Laplacian motivated by the same principle stated above was proposed in prior works [2, 3]. We present this Laplacian first, then run a GCN over the simple graph associated with this hypergraph Laplacian. Check out [3] for more mathematical details.

3.5 Dual-Level Return Prediction

The dual-level return prediction mechanism operates at two levels: order-level and product-level.

Task 1: Order return prediction. We assign a feature embedding vector to each node and then use the GCN technique on a hypergraph to train a model for order-level return prediction. We

conduct the training process by leveraging the order label information provided in task 1. During this process, the feature embedding vectors prepared for each order capture the hidden information of the orders and help understand the relationships between surrounding orders. Consequently, graph representation is learned not only with respect to orders and products but also by integrating various information such as color, size, and group to determine the return of an order.

Task 2: Product return prediction. At the product level, the framework predicts which specific products within an order are likely to be returned. This prediction is based on the incidence and patterns of returns observed for similar products in similar orders, again using a method derived from Task 1. This method allows for more sophisticated predictions through a double process of first predicting at the order level and then predicting probability again at the product level. Using mathematical proof, we use the return probability of the order from Task 1 to predict product-level returns without any additional modifications. According to our knowledge, the return probability of product g can be expressed as follows:

$$\begin{aligned}\hat{P}_r\{R_p(p)=1\} = \hat{P}_r\{R_p(p)=1 \mid R(o)=1 \cup R(o)=2\} \\ \cdot \hat{P}_r\{R(o)=1 \cup R(o)=2\}\end{aligned} \tag{1}$$

where $R_p(p) = 1$ is product return probability and p represents product. Also, the second term on Equation 1, $\hat{P}_r\{R(o) = 1 \cup R(o) = 2\}$ is obtained using our algorithm from Task 1, which utilizes E-HyperGCN+. Here, we must remember that the labels for Task 1 and Task 2 are different. In Task 2, the label is 2 if all products within an order are returned and 1 if only some of the products are returned. Using a simple approach, we treated orders with a label of 1 in Task 1 (indicating partial product returns) as baskets where all products are returned in Task 2. Here, we must remember that the labels for Task 1 and Task 2 are different. In Task 2, the label is 2 if all products within an order are returned and 1 if only some of the products are returned. Using a simple approach, we treated orders with a label of 1 in Task 1 (indicating partial product returns) as baskets where all products are returned in Task 2. Also, By using the product-specific return labels from Task 2, we can recreate data identical to the scenario in Task 1. By using the return labels for each product, we can assign labels to the order accordingly.

Additionally, the first term in Equation 1, $\hat{P}_r\{R_p(p) = 1 \mid R(o) = 1 \cup R(o) = 2\}$ can be expressed as the ratio of orders that include the hyperedge of the product p among all orders where a return has occurred. This equation can be represented as follows:

$$\begin{aligned}&\hat{P}_r\{R_p(p)=1 \mid R(o)=1 \cup R(o)=2\} \\ &= \frac{|V(R(o)=1 \cup R(o)=2)| \text{ in } E_p}{|V| \text{ in } E_p}\end{aligned} \tag{2}$$

Through a simple mathematical approach, we can easily and accurately determine the return probability at the product level by using order-level predictions without changing the design of the hypergraph. This approach does not require additional algorithms and is derived while maintaining the hypernode as the order and the hyperedge as the product in the hypergraph.

Details of the Task2 dataset transformation. Since the IDs in the Task 2 dataset have been randomly shuffled compared to the Task 1 dataset, the Task 2 product-level dataset needs to be reorganized into an order-level dataset to apply Task 1's methodology. In Task 2, we labeled each order based on the labels of the products within that order: if all products were returned, the label was set to 2; if some products were returned, the label was set to 1; and if no products were returned, the label was set to 0. This labeling method is identical to the one used in Task 1. However, since the products in the `task2_test_query.txt` file do not have labels, it was not possible to assign labels to those orders. Therefore, the orders in the `task2_test_query.txt` file were used as test orders for label prediction. The orders with confirmed labels were split into training and validation datasets in a 75:15 ratio, ensuring the original dataset's train/valid/test split of 70:15:15.

Consequently, the E-HyperGCN+ framework's integration of hypergraph modeling with GCN techniques represents a significant advancement in predicting product returns. Addressing the problem at both the order and product levels provides a comprehensive tool for e-tailers to proactively manage returns, enhancing customer satisfaction and reducing operational costs.

4 EXPERIMENTS

We explored various methods to create feature embeddings for each hypernode. First, we created a bag of words using attributes such as customer, color, group, and size for each unique order and then applied SVD for dimensionality reduction. After that, we grouped the attributes by order and used a hash function to create binary feature embeddings for training. Additionally, we applied word2vec to each attribute to obtain feature embeddings, grouped them into unique orders, and then concatenated the resulting feature embedding vectors. Ultimately, the clustering-based methodology showed the best performance on our dataset. As mentioned in the Methods section, the best performance was achieved when we used multi-hot encoding for each attribute, followed by K-means clustering, and then PCA for dimensionality reduction of the cluster-based feature embeddings.

Our dataset posed challenges for model convergence due to its large size, unlike DBLP and other graph datasets. To address this issue, we used a normalization term before the final log softmax layer, accelerating the convergence rate. Based on these various approaches, we evaluated E-HyperGCN+ using the datasets provided for Task 1 and Task 2 with embeddings created using word2vec, node2vec with random walk, clustering, and clustering with PCA on the validation dataset.

4.1 Task1 Results

As seen in Table 1, clustering with PCA showed the best performance. For the Word2Vec method, we treated each node as a word and each edge as a sentence to embed the hypergraph. While this approach effectively captured the connections between nodes, it had limitations in representing the relationships among nodes connected by multiple edges and handling the large number of nodes.

To address these limitations, we applied clustering followed by a random walk to apply Word2Vec. However, this approach performed worse than using multi-hot embeddings.

Through these experiments, we found that providing multi-hot encoding along with additional information yielded better performance for hyperGCN compared to using dense embeddings without multi-hot encoding. However, using only multi-hot encoding requires the neural network to expend significant learning capacity to extract important information from the features. Therefore, we aimed to pre-extract crucial information and provide it as input. To achieve this, we concatenated the PCA-transformed features with the multi-hot encoded features, which resulted in the highest performance.

	Accuracy
Random guessing	0.3329
SVM	0.3789
Random Forest	0.3874
Ours (word2vec)	0.5176
Ours (node2vec)*	0.5176
Ours (clustering)	0.5719
Ours (clustering w/ PCA)	**0.5956**

Table 1: The performances of Task1. node2vec* were generated based on the clustered edges; however, multi-hot encoding was not used.

4.2 Task2 Results

Figure 2 and table 2 show the results when various approaches of feature embedding methods are applied to E-HyperGCN+.

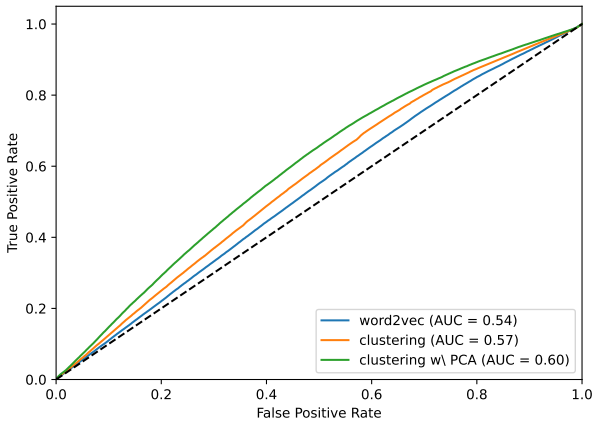


Figure 2: Receiver Operating Characteristic for Various Embeddings on Ours(E-HyperGCN+)

	AUROC	Accuracy
Random guessing	0.4994	0.4994
SVM	0.4942	0.5028
Random Forest	0.5354	0.5231
Ours (word2vec)	0.5350	0.5247
Ours (clustering)	0.5652	0.5453
Ours (clustering w/ PCA)	**0.5991**	**0.5719**

Table 2: The performances of Task2

5 CONCLUSIONS

In conclusion, we recognize the importance of predicting product returns in the e-commerce market and propose an effective method called E-HyperGCN+, which predicts at both the product and order levels. Our method, where nodes are set as orders and edges as products, applies GCN to hypergraphs, providing a deep understanding of e-commerce order data. Additionally, our approach enables end-to-end prediction at the product level based on order-level predictions through mathematical proof without requiring additional model modifications. Furthermore, to utilize the diverse attributes of each order, we propose a feature embedding method using K-means clustering, and we achieve better performance by reducing dimensions with PCA. This step-by-step approach offers a more efficient understanding of the given e-commerce shopping dataset and effectively comprehends the complexity of real-world data.

REFERENCES

[1] Sameer Agarwal, Kristin Branson, and Serge Belongie. Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pages 17–24, 2006.

[2] T-H Hubert Chan and Zhibin Liang. Generalizing the hypergraph laplacian via a diffusion process with mediators. *Theoretical Computer Science*, 806:416–428, 2020.

[3] T-H Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. Spectral properties of hypergraph laplacian and approximation algorithms. *Journal of the ACM (JACM)*, 65(3):1–48, 2018.

[4] Hao-Fei Cheng, Eyal Krikon, and Vanessa Murdock. Why do customers return products? using customer reviews to predict product return behaviors. In *Proceedings of the 2024 Conference on Human Information Interaction and Retrieval*, pages 12–22, 2024.

[5] Minyoung Choe, Sunwoo Kim, Jaemin Yoo, and Kijung Shin. Classification of edge-dependent labels of nodes in hypergraphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 298–309, 2023.

[6] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[7] Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, and Kijung Shin. A survey on hypergraph neural networks: An in-depth and step-by-step guide. *arXiv preprint arXiv:2404.01039*, 2024.

[8] Jianbo Li, Jingrui He, and Yada Zhu. E-tail product return prediction via hypergraph-based local graph cut. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 519–527, 2018.

[9] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32, 2019.

[10] Chenzi Zhang, Shuguang Hu, Zhihao Gavin Tang, and TH Hubert Chan. Re-revisiting learning on hypergraphs: confidence interval and subgradient method. In *International Conference on Machine Learning*, pages 4026–4034. PMLR, 2017.

[11] Yada Zhu, Jianbo Li, Jingrui He, Brian Leo Quanz, and Ajay A Deshpande. A local algorithm for product return prediction in e-commerce. In *IJCAI*, pages 3718–3724, 2018.

A APPENDIX

A.1 Labor Division

The team performed the following tasks

- Writing introduction and propose improvements for baseline [Hyogon]
- Code reproduction and fixed [Sungwoo, Seojeong]
- Related work investigation [all]
- write the method, experiment sections [all]

A.2 Full disclosure wrt dissertations/projects

Sungwoo: He is not doing any project or dissertation related to this project: his thesis is on audio-visual multimodal speech recognition.

Hyogon: He is not doing any project or dissertation related to this project: his thesis is on model compression, specifically quantization of Large Vision Model.

Seojeong: She is not doing any project or dissertation related to this project: her thesis is on video moment retrieval and highlight detection.