Robust User Engagement Modeling With Transformers and Self Supervision

Yichao Lu
Layer 6 AI
Toronto, Canada
yichao@layer6.ai

Maksims Volkovs
Layer 6 AI
Toronto, Canada
maks@layer6.ai

ABSTRACT

Online advertising has seen exponential growth transforming into a vast and dynamic market that encompasses many diverse platforms such web search, e-commerce, social media and mobile apps. The rapid growth of products and services presents a formidable challenge for advertising platforms, and accurately modeling user intent is increasingly critical for targeted ad placement. The 2023 ACM RecSys Challenge, organized by ShareChat, provides a standardized benchmark for developing and evaluating user intent models using a large dataset of impression from the ShareChat and Moj apps. In this paper we present our approach to this challenge. We use Transformers to automatically capture interactions between different types of input features, and propose a self-supervised optimization framework based on the contrastive objective. Empirically, we demonstrate that self-supervised learning effectively reduces overfitting improving model generalization and leading to significant gains in performance. Our team, Layer 6 AI, achieved 1st place on the final leaderboard out of over 100 teams.

ACM Reference Format:
Yichao Lu and Maksims Volkovs. 2023. Robust User Engagement Modeling With Transformers and Self Supervision. In *ACM RecSys Challenge 2023 (RecSysChallenge '23), September 19, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626221.3627285

1 INTRODUCTION

Machine learning has become a key component of online advertising platforms providing vital capabilities for advertisers, publishers and consumers. With the rapid growth of products and services, automated ad placement that targets user intent through personalization is now the default approach on most advertising platforms. Improving the underlying models that power these services while ensuring fairness, privacy and transparency has been an ongoing area of research that has received much attention. The 2023 ACM RecSys Challenge, organized by ShareChat, provides a standardized benchmark for developing and evaluating user intent models. The challenge is based on a large scale dataset of user impressions from the ShareChat and Moj apps, and the goal is to predict whether users installed the app following the ad impression.

RecSysChallenge '23, September 19, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-1613-3/23/09…$15.00
https://doi.org/10.1145/3626221.3627285

In this paper we present our approach to this challenge. We introduce a model agnostic self-supervised learning framework that is based on contrastive optimization. To facilitate contrastive learning we develop data augmentation operators that produce positive and negative views tailored specifically to the types of input features that are commonly encountered in the online personalization tasks. Using a Transformer-based [26] model and jointly optimizing supervised and self-supervised objectives we demonstrate significant improvements in model accuracy and generalization. Over 100 teams participated in this challenge and our team, Layer 6 AI, achieved first place with over 2.5% relative improvement over the 2nd place team.

2 APPROACH

Each instance in the ShareChat dataset corresponds to a user-ad impression sampled from a 22-day time period. Typical for data of this type, instances contain real-valued $\mathbf{x}^r$, categorical $\mathbf{x}^c$ and binary $\mathbf{x}^b$ features that summarize user and ad information as well as historical user-ad interactions. Each instance is also associated with a binary target y where $y = 1$ if user installed the app following the ad impression and $y = 0$ otherwise. The dataset of instance-target pairs $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i = [\mathbf{x}_i^r, \mathbf{x}_i^c, \mathbf{x}_i^b]$ is partitioned into a training set that covers the first 21 days of the time period, and a test set that spans the 22nd day. The aim is to leverage the training data to predict which impressions led to the app install in the test set. In the following sections we describe our approach to this problem where we leverage Transformers and self-supervised learning to model user intent. Figure 1 shows the full architecture diagram of our model.

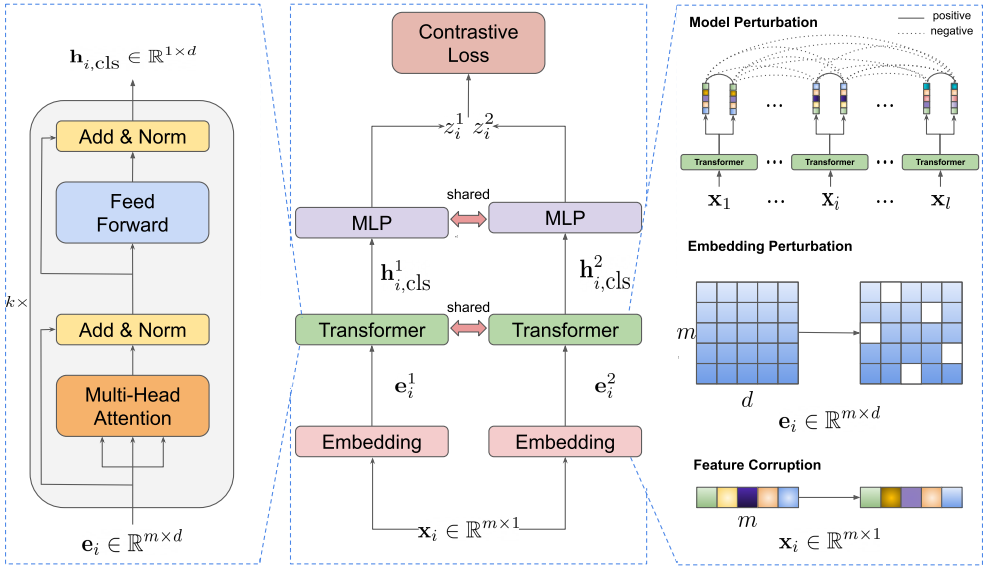
2.1 Input Representation

To facilitate information flow and reduce gradient instability that can result from features of different types, we represent each instance by a set of embeddings. Categorical and binary features are passed through a standard embedding layer where separate embedding is learned for each category (binary features are treated as having two categories):

$$\begin{aligned} \mathbf{e}_i^c &= \text{Embed}(\mathbf{x}_i^c) \\ \mathbf{e}_i^b &= \text{Embed}(\mathbf{x}_i^b) \end{aligned} \tag{1}$$

Numerical features are converted into embeddings through discretization. Discretizing numerical features stabilizes optimization when features have different ranges and makes the model more robust to noises and outliers. A number of feature discretization techniques have been proposed in the literature, such as fixed (equal-width) binning [13], quantile-based (equal-frequency) binning [9],

Figure 1: Model architecture diagram for our contrastive learning approach. For each input instance $\mathbf{x}_i$ we sample two augmentation operators and apply them to $\mathbf{x}_i$ as it is passed through an embedding layer and k Transformer blocks, resulting in two augmented view representations $\mathbf{h}_i^1$ and $\mathbf{h}_i^2$. The views are then passed through an MLP to get the embeddings $\mathbf{z}_i^1$ and $\mathbf{z}_i^2$ which are treated as a positive pair in the contrastive loss and pushed together; all other views in the batch are treated as negatives and pushed away from $(\mathbf{z}_i^1, \mathbf{z}_i^2)$.



and k-means clustering [11]. We adopt the quantile-based binning here due to its robustness to highly skewed data distributions.

Quantile binning discretizes a continuous variable into discrete bins based on specified quantiles (or percentiles). Formally, for a given numerical feature x^r, we divide its value range into the disjoint set of t bin intervals $b_1, ..., b_t$. Bin intervals are constructed on the training set so all bins have the same number of data points. Note that when the number of unique values for a feature is less than t, we directly assign each value to a bin. Once bins are constructed, each instance x_i^r is mapped to a bin interval based on its feature value effectively transforming $\mathbf{x}^r$ into a categorical feature. We then treat $\mathbf{x}^r$ the same way as categorical features by learning an embedding for each category:

$$\mathbf{e}_i^r = \text{Embed}(\text{Bin}(\mathbf{x}_i^r)) \tag{2}$$

where Bin() is the binning operation that assigns a bin index to each feature in $\mathbf{x}_i^r$. The resulting embedding tensor $\mathbf{e}_i = [\mathbf{e}_i^r, \mathbf{e}_i^c, \mathbf{e}_i^b] \in \mathbb{R}^{m\times d}$, where m is the number of features and d is the embedding dimension, forms the input to our model.

2.2 Model Architecture

We aim to learn extensive feature interactions that can accurately identify patterns of user behavior that lead to the target action of app install. Transformers [26] have shown excellent generalization in many domains [4, 10, 18, 20, 31] with an intrinsic ability to learn rich representation from diverse input data. Self-attention can be highly beneficial to personalization allowing to dynamically focus on subsets of features relevant to a given user [14, 19, 23]. Inspired by these results we form our model architecture with k Transformer blocks:

$$\mathbf{h}_i = \text{Transformer}_k(....\text{Transformer}_1([[\textbf{CLS}], \mathbf{e}_i])) \tag{3}$$

where $\mathbf{h}_i \in \mathbb{R}^{(m+1)\times d}$ is the output representation. Note that we prepend the [**CLS**] token to $\mathbf{e}_i$ that summarizes relevant information from the input so $\mathbf{h}_i$ has an extra dimension.

2.3 Training and Inference

Self-supervised learning has established itself as a default approach to learn robust representations that improve model generalization [17, 19]. Virtually all recent advances in generative models are largely enabled by self-supervision [3, 5, 8, 22, 25, 29]. Given that our labeled training set has relatively few instances, we apply self-supervised learning to provide additional regularization to the model and improve representation quality. We combine self-supervised and supervised objectives, and first train with a joint loss followed by finetuning with a supervised loss only to adapt the model to the target task. In the following sections we describe both objectives.

2.3.1 Supervised Objective. The supervised objective follows the standard binary classification set-up. Given the output representation $\mathbf{h}_i$ from the last Transformer block, we pass it through a multilayer perceptron (MLP) followed by a sigmoid activation to get the app install probability:

$$\hat{y}_i = \sigma\left(\text{MLP}\left(\mathbf{h}_{i,\text{cls}}\right)\right) \tag{4}$$

where $\mathbf{h}_{i,\text{cls}}$ is the representation for the [**CLS**] token. We then apply the binary cross entropy loss to maximize the probability of

installs:

$$\mathcal{L}_{\text{sup}} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log \hat{y}_i + (1-y_i)\log\left(1-\hat{y}_i\right)\right) \tag{5}$$

During inference, given a test instance $\mathbf{x}_j$, we apply the full pipeline and take $\hat{y}_j$ as model prediction.

2.3.2 Self-supervised Objective. For the self-supervised objective we focus on contrastive learning [7, 24] where representations are learned via a contrastive loss that pushes apart dissimilar data pairs while pulling together similar ones. Similar (positive) pairs are typically generated by constructing different views of the same data instance via augmentation, while dissimilar (negative) pairs are generated from views of other data instances. We utilize three different data augmentation approaches: feature corruption [1, 28], embedding perturbation [27, 30] and model perturbation [12], which perturb the input instance at different levels of granularity.

Feature Corruption. This augmentation is applied directly to input features. Given an input instance $\mathbf{x}_i$, we randomly sample a subset of features and replace the value for each sampled feature with a random sample from a uniform distribution over all possible values that this feature takes in the training set. The augmented instance $\widetilde{\mathbf{x}}_i$ is then passed to the embedding layer described in Section 2.1, and the resulting embedding $\widetilde{\mathbf{e}}_i$ is given to the model. This augmentation is applicable to both numeric and categorical features and encourages the model to be robust to noise. Moreover, there is an interesting analogy to masked training in NLP where models such as BERT [8] corrupt input sentences by randomly replacing words with other words sampled from the corpus.

Embedding Perturbation. Inspired by previous works on self-supervised learning in recommender systems [27, 30], we also construct augmented views by perturbing the input embeddings $\mathbf{e}_i$. The embeddings are perturbed with random binary masks sampled from the Bernoulli distribution, which is analogous to applying the embedding dropout. Formally, given the input embedding tensor $\mathbf{e}_i \in \mathbb{R}^{m\times d}$, the perturbed embedding tensor $\widetilde{\mathbf{e}_i} \in \mathbb{R}^{m\times d}$ is obtained by:

$$\widetilde{\mathbf{e}}_i = \mathbf{e}_i \cdot \boldsymbol{I}, \quad \boldsymbol{I}_j \sim \text{Bernoulli}(p_j) \in \mathbb{Z}_2^{1\times d} \tag{6}$$

where $\boldsymbol{I} \in \mathbb{Z}_2^{m\times d}$ is the binary tensor, and p_j is the masking rate for the j-th feature. We use different perturbation rates for different features to account for the fact that each feature has its own distribution. Similarly to feature corruption, this augmentation is designed to make the model robust to feature corruption but at the embedding level.

Model Perturbation. Both feature corruption and embedding perturbation are applied explicitly to the input and encourage the model to be robust to noise and missing features. We additionally apply implicit data augmentation, which uses the internal stochasticity of the model to generate different views of a data instance. In particular, for the same data instance we make two passes through the model with different dropout seeds at each Transformer block. This produces two representations of the same input that have been perturbed by the internal model structure and encourages full model robustness.

The three augmentation approaches cover all stages of our pipeline from raw features to input embeddings and the model itself. By stochastically combining them we can encourage all components of the model to be robust to noise and missing data. We leverage this approach to define the contrastive objective. Given a batch of data with l instances $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_l\}$, we generate two augmented views for each instance by sampling separate augmentation operators. The views generated from the same instance are considered a positive pair, while the other $2(l-1)$ augmented views from the batch are taken as negative examples. Passing these views through the model produces $2l$ representations $\{(\mathbf{h}_1^1, \mathbf{h}_1^2), (\mathbf{h}_2^1, \mathbf{h}_2^2), ..., (\mathbf{h}_l^1, \mathbf{h}_l^2)\}$ where $(\mathbf{h}_i^1, \mathbf{h}_i^2)$ is the positive pair for the i'th input instance $\mathbf{x}_i$.

Table 1: Final leaderboard results for the top 5 teams. Our team is Layer 6 AI.

Position	Team Name	Score
1	Layer 6 AI	**5.744062**
2	LearningFE	5.892977
3	hahaha	5.904369
4	Ainvest	5.949816
5	Shield	5.958641

Inspired by SimCLR [7], we use an MLP with one hidden layer to obtain compact hidden representations:

$$\begin{aligned}\mathbf{z}_i^1 &= \max\left(0, \mathbf{h}_{i,\text{cls}}^1 \cdot \mathbf{W}_1 + \mathbf{b}_1\right)\cdot \mathbf{W}_2 + \mathbf{b}_2 \\ \mathbf{z}_i^2 &= \max\left(0, \mathbf{h}_{i,\text{cls}}^2 \cdot \mathbf{W}_1 + \mathbf{b}_1\right)\cdot \mathbf{W}_2 + \mathbf{b}_2\end{aligned} \tag{7}$$

Here, we also use the [**CLS**] representation $\mathbf{h}_{i,\text{cls}}$ to ensure that information flow is consistent with the supervised objective. These representations are then passed to the contrastive loss [21]:

$$\mathcal{L}_{con} = -\frac{1}{l}\sum_{i=1}^{l}\log\left(\text{softmax}\left(\text{sim}(\mathbf{z}_i^1, \mathbf{z}_i^2)/\tau\right)\right) \tag{8}$$

where $\text{sim}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^\top \boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}$ is the cosine similarity, τ is the temperature, and softmax is normalized over all views in the batch. The contrastive objective aims to bring views in each positive pair together and separate them from all other views in the batch.

The joint objective combines supervised and contrastive losses:

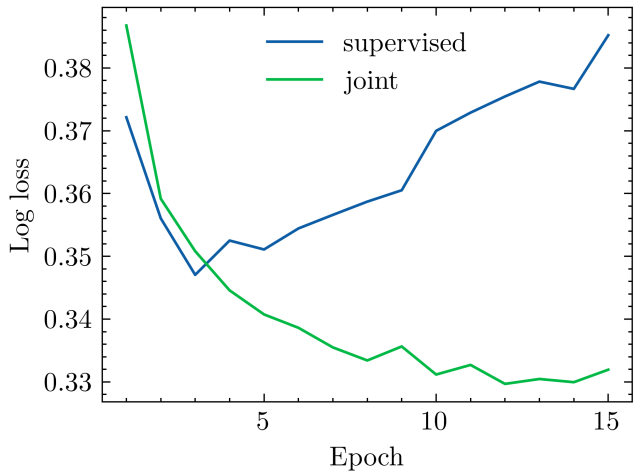
$$\mathcal{L}_{joint} = \alpha \cdot \mathcal{L}_{sup} + (1-\alpha)\cdot \mathcal{L}_{con} \tag{9}$$

where $\alpha \in [0, 1]$ is a hyperparameter controlling the contribution of each loss. For the pre-training phase we set $\alpha < 1$ to include the contrastive optimization, then during fine-tuning we set $\alpha = 1$ to focus on the target supervised task.

3 EXPERIMENTS

We partition the data by taking the last day of the 21-day training period as validation set and the other 20 days as training set. This results in 3,387,880, 97,972 and 160,973 instances for training, validation and test respectively. Our Transformer model architecture has $k = 6$ self-attention blocks with input embedding dimension 128, feed forward dimension 128 and 8 attention heads. The dropout rate is set to 0.1 for pre-training, and to 0.05 for fine-tuning. The model is trained using the AdamW optimizer [16] with a batch size of 4096. Learning rate is warmed up for 2000 iterations until $3e-4$ and decayed with cosine decay afterwards. For pre-training α is set to 0.6 in the joint objective, and this phase is followed by fine-tuning

Figure 2: Validation log loss curves for models trained with supervised and joint supervised-contrastive objectives.



with a supervised only objective. In both phases we monitor the validation log loss to find the best epoch for early stopping.

To compute views for the contrastive loss we randomly sample two augmentation operators for each instance in the batch. For Feature Corruption we randomly select a subset of 5% - 25% features and replace their values. Similarly, for Embedding Perturbation we sample masks from the Bernoulli distribution with masking rates $p_j \in [0.05, 0.25]$. Finally, for the Embedding Perturbation we use the default dropout rate of 0.1 and make two passes through the model for each instance letting the stochasticity of the dropout do the perturbation.

We perform distributed hyperparameter search across GPUs, and use the Tree-structured Parzen Estimator algorithm [2] to perform efficient search. To mitigate the effect of random seeds on final performance, for each set of hyperparameters, we retrain the model with 5 different random seeds and use the validation log loss of the ensembled predictions to evaluate the hyperparameters. After obtaining the best hyperparameters, we retrain the model on the full 21 day training period to get the final model for submission. All experiments are done on the p4d.24xlarge instance from the Amazon Web Services with 8 NVIDIA A100 GPUs.

3.1 Results

The final leaderboard test set results are shown in Table 1. The teams are evaluated using the normalized cross entropy metric which is proportional to log loss. Over 100 teams participated in this challenge and our team, Layer 6 AI, achieved first place with over 2.5% relative improvement over the 2nd place team. Our final submission is based on a linear blend of multiple models trained with different seeds using the joint supervised-contrastive objective described in Section 2.3.

To demonstrate the effectiveness of incorporating self-supervised learning into model optimization, Figure 2 shows validation log loss curves for models trained with supervised and joint supervised-contrastive objectives. The supervised only model exhibits significant overfitting after the 3rd epoch despite having dropout in every Transformer block and weight regularization. This is expected as with 6 Transformer blocks the model has over 2M learnable parameters, and has the capacity to memorize a large portion of the relatively small training set. Adding the contrastive objective provides effective regularization and improves model generalization leading to significantly lower validation log loss. Across all experiments we were unable to achieve competitive results with the supervised only setting so self-supervised learning is essential for high capacity models.

Table 2: Top: results for tree-based gradient boosting baselines. Middle: ablation results for different objectives. Bottom: results for different self-supervised tasks.

Method	Log loss
LightGBM	0.3615
XGBoost	0.3582
supervised	0.3471
supervised + augmentation	0.3448
joint	**0.3297**
value estimation [28]	0.3385
value + mask estimation [28]	0.3353
contrastive	**0.3297**

3.1.1 Ablation Study. To further investigate the contribution of individual components in our model pipeline we conduct an extensive ablation study. Table 2 expands on results in Figure 2 where in addition to supervised only and joint objectives, we also experiment with adding augmentation to the supervised training. This is done by passing each instance in a batch through a randomly sampled augmentation operator, but still optimizing the model with a supervised only loss. From the middle section of Table 2 we see that supervised+augmentation improves over the supervised only training indicating that augmentation can improve generalization. However, performance is still considerably lower than for joint training so explicitly incorporating contrastive loss is important. Moreover, comparing to popular tree-based gradient boosting models LightGBM [15] and XGBoost [6], that are typically the default choice for tabular datasets, we see that our Transformer approach is significantly better even with the supervised only objective. Note that both LightGBM and XGBoost were extensively tuned during the competition and represent the best scores that we were able to obtain.

Bottom section of Table 2 shows performance for different self-supervised tasks. We baseline contrastive learning against the popular approach of mask prediction used in BERT [8] and other models. We follow the adaptation of this approach to tabular data by [28] that introduced the tasks of value and mask estimation. From the table we see that value+mask estimation is also effective and improves performance considerably over the supervised only model. However, contrastive task is still the best performing method so

Table 3: Ablation results for augmentation operators.

Augmentation	Log loss
Feature Corruption (FC)	0.3417
Embedding Perturbation (EP)	0.3438
Model Perturbation (MP)	0.3455
FC + EP	0.3337
FC + EP + MP	**0.3297**

placing explicit constraints on distances between representations is beneficial for tabular data.

Table 3 shows ablation results for augmentation operators in the contrastive loss. To evaluate the contribution of each operator we apply them individually where all augmented views are done by the same operator. We also evaluate Feature Corruption and Embedding Perturbation without Model Perturbation. The results show that contrastive learning is beneficial even with a single augmentation operator as results for all three individual operators improve over the supervised only model. Interestingly, Feature Corruption shows the strongest performance indicating that conditioning the model to noise in raw input features is highly beneficial for performance. Combining operators further improves performance by a large margin and using all three operators gives the best results. Augmenting at different stages of the pipeline promotes robustness of the entire model and stochasticity between operators acts as additional regularization.

4 CONCLUSION

In this paper, we present our approach to the 2023 ACM RecSys Challenge organized by ShareChat. Our best model is based on a Transformer architecture tailored to online advertising. The model is trained with a combination of self-supervised and supervised tasks, and we demonstrate the effectiveness of incorporating self-supervised learning for online user intent modeling. We achieved highly competitive performance, placing 1st on the final leaderboard out of over 100 teams.

REFERENCES

[1] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. 2021. Scarf: Self-Supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*.

[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems* 24 (2011).

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Neural Information Processing Systems*.

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*.

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision*.

[6] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

[9] Elena S Dimitrova, M Paola Vera Licona, John McGee, and Reinhard Laubenbacher. 2010. Discretization of time series data. *Journal of Computational Biology* 17, 6 (2010), 853–868.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[11] James Dougherty, Ron Kohavi, and Mehran Sahami. 1995. Supervised and unsupervised discretization of continuous features. In *Machine Learning*. Elsevier, 194–202.

[12] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Empirical Methods in Natural Language Processing*.

[13] Syed Tanveer Jishan, Raisul Islam Rashu, Naheena Haque, and Rashedur M Rahman. 2015. Improving accuracy of students' final grade prediction model using optimal equal width binning and synthetic minority over-sampling technique. *Decision Analytics* 2 (2015), 1–25.

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining*.

[15] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[16] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA.

[17] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. In *Transactions on Knowledge and Data Engineering*.

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision*.

[19] Yichao Lu, Zhaolin Gao, Zhaoyue Cheng, Jianing Sun, Bradley Brown, Guangwei Yu, Anson Wong, Felipe Pérez, and Maksims Volkovs. 2022. Session-based Recommendation with Transformers. In *Recommender Systems Challenge*.

[20] Yichao Lu, Himanshu Rai, Jason Chang, Boris Knyazev, Guangwei Yu, Shashank Shekhar, Graham W Taylor, and Maksims Volkovs. 2021. Context-aware scene graph generation with seq2seq transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 15931–15941.

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. *OpenAI Blog* (2018).

[23] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *International Conference on Information and Knowledge Management*.

[24] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020. What makes for good views for contrastive learning?. In *Neural Information Processing Systems*.

[25] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

[27] Fangye Wang, Yingxu Wang, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2023. CL4CTR: A Contrastive Learning Framework for CTR Prediction. In *ACM International Conference on Web Search and Data Mining*.

[28] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. 2020. Vime: Extending the success of self-and semi-supervised learning to tabular domain. In *Neural Information Processing Systems*.

[29] Wei Yu, Yichao Lu, Steve Easterbrook, and Sanja Fidler. 2020. Efficient and information-preserving future frame prediction and beyond. (2020).

[30] Xin Zhou, Aixin Sun, Yong Liu, Jie Zhang, and Chunyan Miao. 2023. Selfcf: A simple framework for self-supervised collaborative filtering. *ACM Transactions on Recommender Systems* 1, 2 (2023), 1–25.

[31] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. 2020. Deformable DETR: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*.