Deep & Wide Recommender System for Amazon

Peilin Liu, Guang Yang, Zonghao Yang
SID: 56514896, 56551056, 56572875

April 19, 2022

1 Motivation

The rise of deep learning methods has brought breakthroughs in research in many fields [9, 7, 16, 3]. Deep learning is capable of processing unstructured data, such as images in computer vision [7, 3] and text data in natural language processing [16, 2, 4]. It has also revolutionized many traditional statistical and data processing methods [10, 13]. Recommender systems [11, 6] stand out as the most profitable deep learning application in the business regime.

As everyone faces a massive flow of information daily, how to extract useful information is the concern of the virtual community and e-commerce platform operators. A powerful recommender system can screen effective information for users and enhance users' stickiness to social platforms, business platforms, and other Internet communities. In turn, it also brings more traffic and loyal users to the platform operators. Moreover, the improving user experience will significantly engage the users' sense of belonging in the platform.

We propose a deep recommender system for the famous e-commerce platform Amazon. The designed neural network calculates the degree of matching between any pair of user and product given multimodal data, which composes the core computing part of a large-scale recommender system.

2 Background

2.1 Recommendation Based on Collaborative Filtering

Recommendation based on collaborative filtering works through the power of collective intelligence, filtering out items that are not of interest to users. Collaborative filtering (CF) [8] is based on the assumption that similar users will have similar preferences over contents. Thus, the recommendation is first to find other users who have similar interests to the target user and then recommend the content similar users like. In practice, the algorithm calculates the distance for each pair of users based on the user's historical preference. Given the pairwise distances, the algorithm uses the nearest neighbor algorithm to predict the target user's choice for a specific product. The User-Item (UI) matrix (Figure 1) is usually the starting point of the recommendation.

Collaborative filtering recommendation can be further divided into memory-based collaborative filtering recommendation (Memory-based CF) and model-based collaborative filtering recommendation (Model-based CF) according to the use of machine learning.

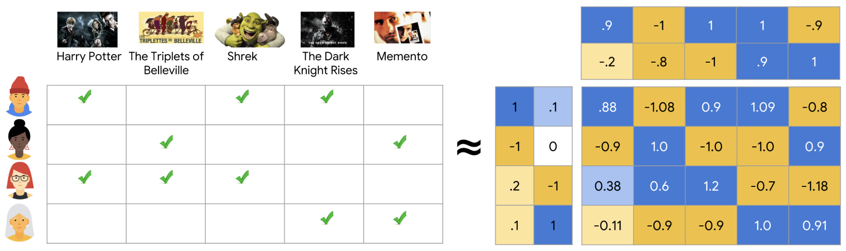


Figure 1: Collaborative Filtering

2.2 Recommendation Based on Contents

Recommendation based on contents uses the inherent quality or inherent attributes of the project to recommend, such as the genre, type of music, style and category of movies, etc., without the need to build a UI matrix. It is based on the content information of the item to make recommendations, and does not need to be based on the user's evaluation of the item, and more needs to use the machine learning method to obtain the user's interest data from the case of the feature description of the content. A general idea of content-based recommendation is to utilize the idea of machine learning to fit the user's feature attributes through training. To achieve this goal, we need a utility function to evaluate the rating of a specific user c for a specific item s. In the era of deep learning, the most common way to calculate the rating between c and s is to construct a neural network to approximate the target function. In the following, we show some classical deep learning methods for calculating the rating.

$$u(c, s) = \text{score}(\text{User}_{\text{feature}}(c), \text{Item}_{\text{feature}}(s))$$

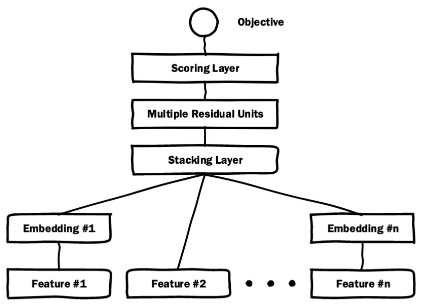


Figure 2: Deep Crossing

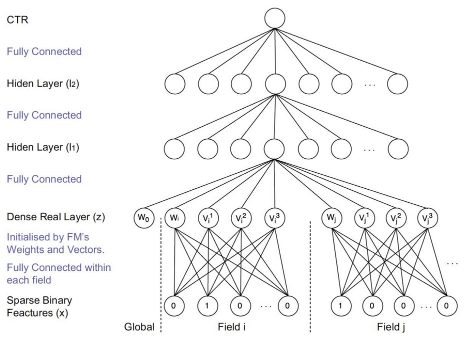


Figure 3: FNN

2.2.1 Deep Crossing: Base Model in the Era of Deep Learning

Deep Crossing [15] is the most typical and basic model of the deep learning in content-based recommendations. As shown in the model structure diagram in Figure 2, it covers the most typical elements of the deep click-through ratio (CTR) model. That is, by adding an embedding layer to convert sparse features into low-dimensional dense features, and using stacking layer, or concat layer to convert segmented feature vectors Connected. Then, complete the combination and transformation of features through a multi-layer neural network. Lastly, use the scoring layer to complete the calculation of CTR. Different from the classic DNN, the multilayer perceptron used by deep crossing is composed of residual networks, which undoubtedly benefits from the famous 152-layer ResNet.

2.2.2 Factorization-Machine Supported Neural Network

The innovation of factorization-machine supported neural network (FNN) [5] compared to deep crossing is to use the hidden layer vector of FM as the Embedding of user and item, thus avoiding training the Embedding from random states completely. Since a large number of id features use the one-hot encoding, their dimensions are extremely high and the vectors are extremely sparse. Therefore, there are many connections between the Embedding layer and the input layer, and the efficiency of gradient descent is very low, which greatly increases the training time and cost of the model. The instability of Embedding, using the pre-train method to complete the training of the Embedding layer is an effective engineering trick to reduce the complexity and training instability of the deep learning model.

2.2.3 Product-Based Neural Network

The key to a product-based neural network (PNN) [14] is to add a Product layer between the embedding layer and the fully connected layer. The traditional DNN directly completes the intersection and combination of features through multiple fully connected layers, but this method lacks particular "targeting". First, the fully connected layer is not designed for crossover between different feature domains. Secondly, the operation of the fully connected layer is not directly designed for feature crossover. However, in practical problems, the importance of feature intersection is self-evident. For example, the intersection of age and gender is an essential grouping feature, which contains much high-value information. We urgently need a deep learning network with a targeted structure that can represent these messages. Therefore, PNN completes the targeted feature cross by adding the Product layer, and its product operation performs feature combinations between different feature domains. And define the operation of inner product, outer product, and other products to capture additional cross information and enhance the ability of the model to represent different data patterns.

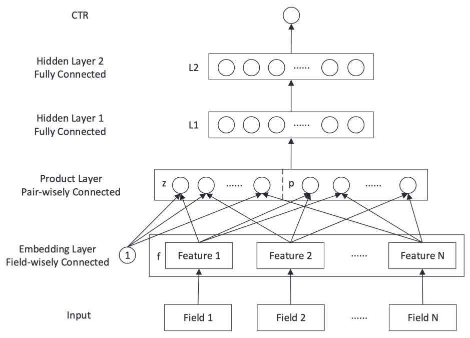

CTR
Hidden Layer 2 Fully Connected
L2
Hidden Layer 1 Fully Connected
L1
Product Layer Pair-wisely Connected
z
p
Embedding Layer Field-wisely Connected
1
f
Feature 1
Feature 2
Feature N
Input
Field 1
Field 2
Field N

Figure 4: PNN

2.2.4 Wide & Deep Model

We advocate the Google Wide & Deep model [1]. The main idea is to connect the Wide part of the single input layer and the Deep part of the multi-layer perceptron and input the final output layer together. The primary function of the Wide part is to make the model have memory (Memorization). The single-layer Wide part is good at processing a large number of sparse id-type features so that the model can directly "remember" a large amount of historical information of the user. On the other hand, the main role of the Deep part is to let the model have "generalization". The Deep part uses the characteristics of DNN's strong expressive ability to mine the data patterns hidden behind the features. Finally, the regression output layer combines the Wide part and the Deep part to form a unified model. There is a large influence of Wide & Deep on subsequent models. Many of them adopt the form of two-part or even multi-part combinations and use different network structures to mine additional information. They make full use of and combine the characteristics of different network structures.

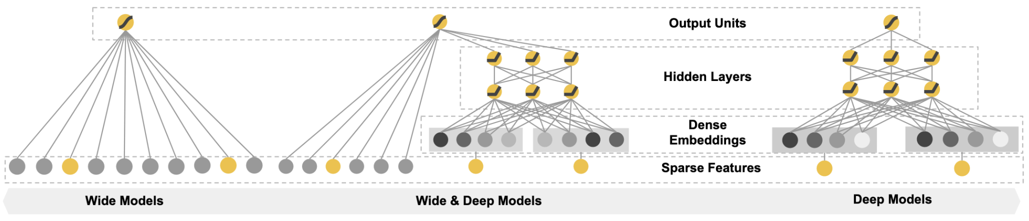


Figure 5: Wide & Deep Model

3 Description

Using the Amazon reviewer dataset, we investigate the data structure of the dataset and generate a rating score for each pair of user and product. We propose a new recommendation pipeline that combines sentiment analysis and a deep neural network to approximate the target score function.

3.1 Sentiment Analysis on Product Reviews

We embed the sentiment analysis of product reviews in our recommender system. Reviews are helpful information that significantly influences customers' shopping decisions. Each product review has an overall score of 1 to 5 and a piece of review text. Compared to text, the overall score is more subjective. In other words, the standards of a five-star or one-star vary from individual to individual. On the other hand, the review text gives a more objective description of the product. Consistency in the evaluation of products is beneficial for a recommender system. Thus, we conduct sentiment analysis on the product reviews, and the resulting sentiment score is an essential input to our recommender network.

The critical challenge for analyzing text is the high dimensionality. Take a short product review of 15 words for example. Assume the corpus only consists of 2000 common English vocabularies, the dimension of such review is 2000^{15}, which is intractable even for a modern computer with advanced CPU and GPU units. With the development of natural language processing (NLP) and deep learning, algorithms such as Word2Vec [12] and BERT [2] can learn the word embeddings so that we can represent the vocabularies in a K-dimension vector space, where K is much smaller than the one-hot representation. Moreover, words with similar meanings are optimized to colocate at similar places so that the wording embedding also carries the semantic meaning. With "low"-dimension word embedding, computers now are able to represent text as data and then use deep learning and statistical machine learning methods to do the downstream tasks [4].

We conduct sentiment analysis on product reviews with BERT. BERT stands for Bidirectional Encoder Representations from Transformers. The BERT model has gained popularity in language preprocessing, as it can generate a low-dimensional word embedding that captures the semantic meaning. And it has demonstrated state-of-the-art performance in many downstream tasks. We use the pre-trained BERT language model provided by Hugging Face[1] and conduct a binary sentiment classification. The outcome of interest is the sentiment score (0-1), which indicates how much the customer likes the product. We can generally interpret the sentiment score as positive, neutral, and negative.

3.2 Wide & Deep Model

As we mentioned before, the Wide & Deep Model is designed to balance the memorization and generalization problems of the model. Its wide sub-component and deep sub-component enable us to deal with extensive sparse data and dense data simultaneously.

In our project, we will encounter many binary variables to indicate whether the item or user has specific properties. Even sometimes, the nonlinear changes of these variables (such as the product of two variables) are very important. The wide sub-component handles this part of the task. For multi-modal dense data (such as product text descriptions and product images), we combine the most widely used natural language processing and computer vision deep neural networks to perform relevant feature extraction.

The most important thing when applying the model is the choice of sparse features and their combinations. In our model, we rank users according to the number of user reviews and combine it with the feature of user verification, creating a new cross-feature to measure the reliability of user reviews. In the deep model part, we use the product image and description as the primary information. Then, we use multi-modal data processing methods to extract the hidden features that can best represent the product.

4 Data

The target per-category data Cell Phones and Accessories includes reviews data and metadata of items. Among available datasets we focus on two of them, *5-core* and *metadata*. *5-core* is a subset of the complete reviews data in which all users and items have at least 5 reviews. And *metadata* is the complete information dataset of items. Our main goal is to utilize information of reviews, items and users if any, to establish a recommender system, which is expected to make appropriate recommendation for users. This section mainly introduces the raw data, and talks about its preprocessing to accord to model input. We show the final processed data at the end.

4.1 Review Dataset: *5-Core*

The review dataset contains 1128437 reviews from 157212 users to 48186 items. Each review is recorded corresponding to 12 features, including overall rating, review text, review images and so on. We select six of them: *asin*, *reviewID*, *overall*, *verified*, *reviewText* and *summary* as they are meaningful to recommendation and easy to utilize. The summary of these variables follows.

- asin: item ID
- reviewer ID: user ID
- overall: rating (numerical)
- verified: if the user is verified or not (binary)
- reviewText: content of review text (text)
- summary: summary of review text (text)

[1]Hugging Face is an open-source AI community that shares benchmark models and pre-trained models.

Among the above, *overall* is our target, i.e. the labels which supervise the training of the model. Naturally, *reviewText* is indispensable and *summary* also matters, both of which will be used as text input. *verified* is a binary variable about users. It is considered that the ratings and reviews from verified users hold higher reference value.

4.2 Item Dataset: *Metadata*

The item dataset contains 589356 items. Each item is recorded corresponding to 19 features, including title, main category, images and so on. In similar way of the selection in review dataset, we select seven columns of them, *asin*, *main_cat*, *description*, *title*, *feature*, *price* and *imageURL*.

- asin: item ID
- main_cat: main category of the item (categorical)
- description: description of the item (text)
- title: title of the item (text)
- feature: feature of the item (text)
- price: price of the item (categorical)
- imageURL: url of image of the item (image)

Among the above, *main_cat* and *price* are categorical variables. We point out that we binarize the price for there are over half of the items not labeled price. That is, if there is a price, *price* is set to 1, otherwise 0. For *main_cat*, there are 29 main categories, covering office products, car electronics, home supplies, etc. *description*, *title* and *feature* are all text and remain to be processed to a standard format.

As for image part, each item should be given an image while there are cases where products either do not have an image or have multiple images. First, we remove the items without images directly, because those without image are not appealing at all. Then, for the items with more than one image, we simply use the first one, which is supposed to be the most representative. After ensuring 'one item one image', we download these images to our local server and resize all of them to (32, 32, 3). For the dataframe, we replace the *imageURL* column by a new column called *image* to match the item-image pairs, whose value is the image filename of uniform format 'asin.jpg'. In training, the model will read the images according to the *image* column to retrieve the image input.

4.3 Extracted Information

The extracted information consists of user and item information, both of which are significant for a recommender system. For the lack of users' information in the raw data, it is necessary to do extraction of this aspect. Note that there exist records of users in the review data, therefore we extract some user information, including average score, average sentiment, total number of reviews and weighted score, as aforementioned in sentiment analysis part. These extracted features are used as user characteristics. Symmetrically, we also employ the review data to extract these variables for items, which to some degree complement the item metadata. The rationale behind is that the user with higher average ratings tends to give high ratings for potential items.

4.4 Joint Dataset

By merging the review dataset, item dataset, and user dataset, we end up with a new dataset containing the joint information of reviews, items, and users. The joint dataset contains 1021580 reviews with 21 corresponding item and user features. We illustrate the first 5 columns of first 2 rows in Figure 6. We show the images of the first two items in Figure 7. Additionally, Figure 8 and Figure 9 demonstrate the distributions of some tabular variables with regard to sentiment. We can find that the distributions of these variables of items and users are similar. Specifically, the ratings and sentiment are negatively skewed distributed, and the total review amount is positively skewed distribution.

In summary, this joint dataset covers the data fields on image, text, and tabular, in according to the input of our model.

5 Implementation

5.1 Sentiment Analysis

We use a pre-trained sentiment model based on BERT to generate the sentiment score of product reviews. Hugging Face Pipeline provides a simple API, which is built on the pre-trained BERT models and abstracts the complex code. It leverages a fine-tuned model on sst2, which is a GLUE task[2]. The training process can breakdown to the following steps.

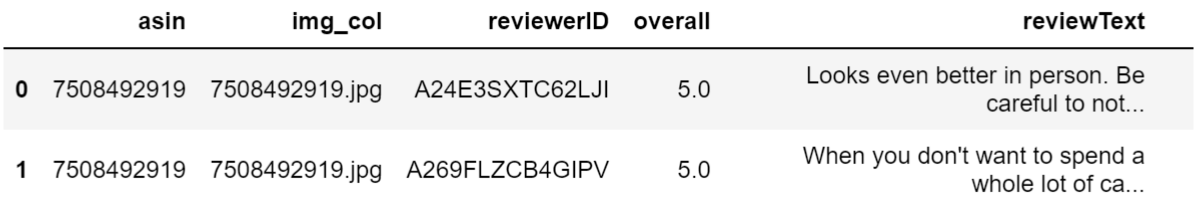
1. Instantiate a tokenizer and a pre-trained BERT model; Load the weights stored in the checkpoint
2. Build a sequence from the review text with the correct token
3. Pass this sequence through the model so that it is classified in one of the two available classes: 0 (negative) or 1 (positive)
4. Compute the softmax of the result to get probabilities over the classes

5.2 Wide & Deep Model

We use the pytorch-widedeep library for specific model implementation. In our code, Preprocesser is to preprocess sparse data, text data, image data, and dense data respectively:

1. Take a nonlinear combination of sparse data
2. Implement word embedding for describing product text data with the help of glove
3. Resize the image so that its dimensions can adapt to the input requirements of the model

[2]GLUE, the General Language Understanding Evaluation benchmark (https://gluebenchmark.com/) is a collection of resources for training, evaluating, and analyzing natural language understanding systems.



	asin	img_col	reviewerID	overall	reviewText
0	7508492919	7508492919.jpg	A24E3SXTC62LJI	5.0	Looks even better in person. Be careful to not...
1	7508492919	7508492919.jpg	A269FLZCB4GIPV	5.0	When you don't want to spend a whole lot of ca...

Figure 6: Joint dataset

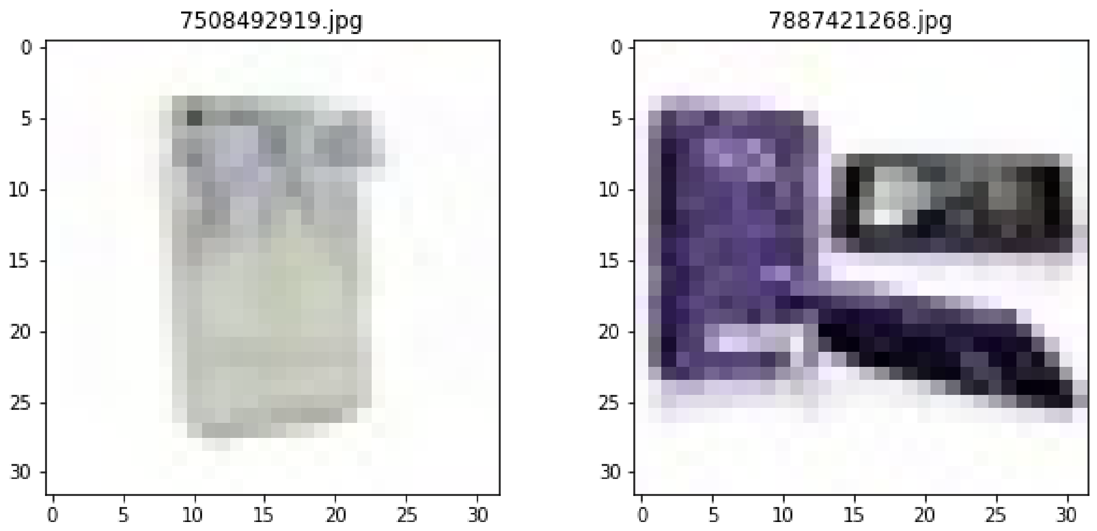


Figure 7: Item images

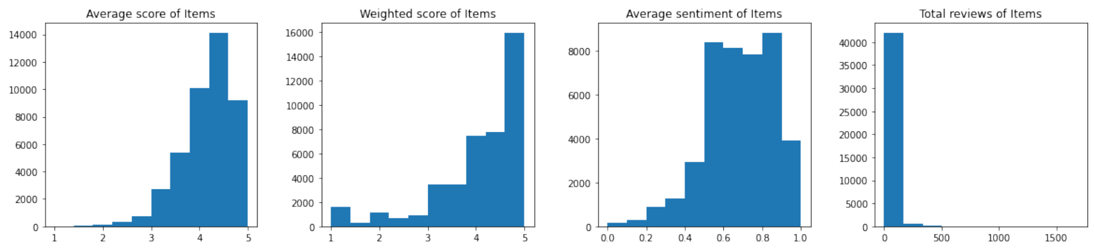


Figure 8: Distribution of Items

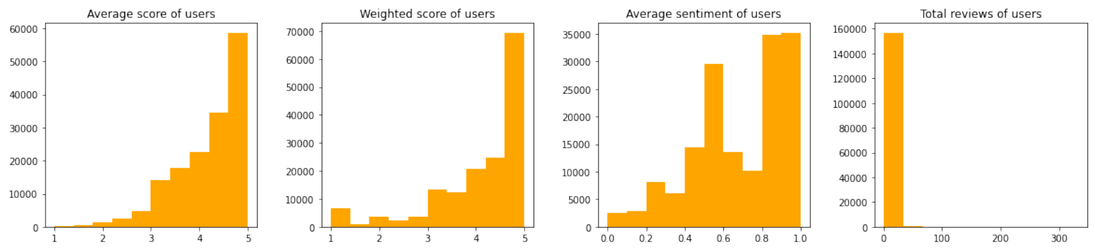


Figure 9: Distribution of Users

As far as the construction of the entire network model is concerned, it is also divided into corresponding parts: Wide, TabMlp, Vision, BasicRNN, WideDeep. Wide is used to process binary data and its nonlinear combination. Vision contains models for processing images (such as ResNet, etc.). BasicRNN mainly processes text data by LSTM. The WideDeep is a combination of these models.

6 Results and Observations

6.1 Sentiment Analysis

We compare the sentiment classification with the overall score given by the customers. We define scores over 3 as positive and others as negative, a binary variable that can be compared with the results from the sentiment analysis.

We use f1 score as the evaluation metric for this binary classification model, defined as a harmonic mean of the precision and the recall of the defaulted class:

$$f1\ score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

where precision is computed as the number of true positives divided by the total number of predictions and recall is the number of true positives divided by the total number of positive instances.

The f1 score of our sentiment classification is 0.867 on the entire sample. The corresponding area under the receiver operating characteristic curve (AUC) is 0.848. The result shows that our sentiment analysis model accurately estimates the customers' sentiment. More importantly, this sentiment score is a uniform and objective description of customers' impression of the products.

6.2 Wide & Deep Model

We used Root Mean Square Error between predicted ratings and true values as the loss function of the model, and divided the dataset into training set, validation set, and test set by 7:1:2, and set the training epoch to 1000 and batch size to 32. The optimizer is Adam, and different learning rates are set for different modules in the network: 0.03 for the Wide network module, 0.0001 for the feature processing sub-network (such as LSTM, pre-trained ResNet) in the deep module, and 0.001 for the embedding network. The following are our experimental results using different combinations of sub-networks:

Image	**Text**	**Training**	**Test**
ResNet18	LSTM (2 stacks)	1.42	1.57
ResNet18	LSTM (4 stacks)	1.30	1.35
ResNet34	LSTM (4 stacks)	1.13	**1.17**
ResNet50	LSTM (4 stacks)	**1.11**	1.20
EfficientNet	LSTM (4 stacks)	1.14	1.23

The combination of ResNet34 and four stacks of LSTM performs the best out of sample with 1.17 of the Root Mean Square Error. Comparing the training and test errors, our model does not term to overfit.

7 Discussions

The project let us reconsider the relationship between business and model design. On Amazon, new products are constantly launched, and there are also many unpopular and niche products. The most popular items are almost always the standard items such as books and snacks. Recall the initial goal of memorization and generalization, the Deep part cannot learn effective embedding for old combinations, but the Wide part steps up which takes advantage of the memory based on the correlation between the product purchased by the user and the recommended product. For the Wide part, its job is to make up for the defects of the Deep part, and most of the other work is handed over to the Deep part. Therefore, the Wide part in the Deep & Wide model is very lightweight compared to an independent Wide module. Finall, the memorization and generalization co-exists in the model which fulfill the intended business needs.

References

[1] Heng-Tze Cheng et al. “Wide & deep learning for recommender systems”. In: *Proceedings of the 1st workshop on deep learning for recommender systems*. 2016, pp. 7–10.

[2] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[3] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[4] Matthew Gentzkow, Bryan Kelly, and Matt Taddy. “Text as data”. In: *Journal of Economic Literature* 57.3 (2019), pp. 535–74.

[5] Huifeng Guo et al. “DeepFM: a factorization-machine based neural network for CTR prediction”. In: *arXiv preprint arXiv:1703.04247* (2017).

[6] Soyeon Caren Han et al. “GLocal-K: Global and Local Kernels for Recommender Systems”. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 3063–3067.

[7] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[8] Xiangnan He et al. “Neural collaborative filtering”. In: *Proceedings of the 26th international conference on world wide web*. 2017, pp. 173–182.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[10] Zichao Long et al. “Pde-net: Learning pdes from data”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3208–3216.

[11] Linyuan Lü et al. “Recommender systems”. In: *Physics reports* 519.1 (2012), pp. 1–49.

[12] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[13] Marcelo Prates et al. “Learning to solve np-complete problems: A graph neural network for decision tsp”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 4731–4738.

[14] Yanru Qu et al. “Product-based neural networks for user response prediction”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE. 2016, pp. 1149–1154.

[15] Ying Shan et al. “Deep crossing: Web-scale modeling without manually crafted combinatorial features”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 255–262.

[16] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).