Scaling Generative Pre-training for User Ad Activity Sequences

Sharad Chitlangia, Krishna Reddy Kesari and Rajat Agarwal

Amazon Ads

Abstract

User activity sequence modeling has significantly improved performance across a range tasks in advertising spanning across supervised learning tasks like ad response prediction to unsupervised tasks like robot and ad fraud detection. Self-supervised learning using autoregressive generative models has garnered interest due to performance improvements on time series and natural language data. In this paper, we present a scalable autoregressive generative pre-training framework to model user ad activity sequences and inspect its scaling properties with respect to model size, dataset size and compute. We show that test loss on pre-training task follows power law scaling with respect to model size, with larger models being more data and compute efficient than smaller models. We also demonstrate that improvement in pre-training test loss translates into better downstream task performance by benchmarking the models on conversion prediction and robot detection tasks in advertising.

Keywords

generative pre-training, self-supervised learning, scaling, invalid traffic, robot detection, digital advertising

1. Introduction

Advances in deep learning have driven a rapid adoption of sequence models applied to user behavioral data for advertising use cases spanning across personalization, ad response prediction, bidding and robot and fraud detection. Deep sequence models reduce reliance on manual feature engineering while utilizing fine grained event level information about the users' activity, leading to improved performance across a wide range of tasks. For tasks like ad response prediction, where labeled data is available at scale, typical approaches use supervised learning to train deep sequence models [4]. However, in domains like ad fraud detection, obtaining accurate labels at scale is implausible and error prone due to unavailability of high coverage ground truth, and attempts to create pseudo labels are fraught with risks of introducing bias. In such scenarios, learning self-supervised user representations is a natural choice. Recent advances have shown that self-supervised pre-training of sequence models not only improves performance on tasks with low-labeled data volumes but also enhances performance over traditional supervised learning on large labeled datasets.

Generative models, which aim to model the input data distribution $P(x)$, have been at the forefront in demonstrating the effectiveness of self-supervised learning. The key idea in self-supervised learning is to construct a proxy task on unlabeled data available at scale, training on which enables the model to learn robust task-agnostic embeddings capturing important characteristics and features about the dataset. Autoregressive models, a class of generative models that perform maximum likelihood estimation by defining an ordering over the input, are a natural fit for language and time series data and have yielded state-of-art results by training highly parallelizable deep sequence model architectures like Transformers [1] on the next-token prediction objective. This has motivated exploration of learning user embeddings using next event prediction on their ad activity sequences as the self-supervised pre-training objective [7, 12].

An interesting property of generative pre-training of Transformers is their enhanced performance with growing model size, data size and compute. Analysis of these scaling properties has garnered interest in the research community, with primary focus so far being on natural language and computer vision data [16, 17]. In this work, we investigate the scaling properties for autoregressive pre-training of user activity sequence models in advertising. Rather than generalizing scaling laws in natural language processing to advertising, we believe user activity sequence models merit an independent scaling analysis, since they are different from text based models in three significant ways. First, instead of a homogeneous time-series of text tokens, user activity sequence is a multi-dimensional time series where each event in the sequence can be described using a variety of features types typically seen in advertising, spanning across discrete, high cardinality, real valued and natural language types. Second, data size in advertising is upper bounded by the number of users interacting with the ad program. This is in contrast with scaling of text-based models where while increasing model size, dataset size is considered to

KDD 2023 Workshop on Artificial Intelligence for Computational Advertising (AdKDD), August 7, 2023, Long Beach, CA

chitshar@amazon.com (S. Chitlangia); kkesari@amazon.com (K. R. Kesari); agrajat@amazon.com (R. Agarwal)

be unbounded as one could crawl more webpages to get additional data. Finally, since traffic patterns and user behavior in advertising are continuously evolving, advertising models need to be retrained continuously, making training cost and time a critical factor in deciding the scaling strategy for deployed settings.

The paper is structured as follows: Section 2 describes the related work, we outline a scalable autoregressive generative pretraining framework for user activity sequences in Section 3. In Section 4, we analyze different scaling properties of the model with respect to model size, dataset size and compute. We present how improvement in test loss during pre-training translates to downstream performance on a supervised task of conversion prediction and an unsupervised task of bot detection in advertising in Section 5. We discuss the key learnings and contrast them with scaling properties in other data domains in Section 6 and conclude in Section 7.

2. Related Work

Generative models aim to learn the input data distribution $P(x)$, and help either estimate the probability of a given data point or sample a data point from the input distribution [31, 32, 33]. In this paper, we are primarily concerned with autoregressive deep generative models. The autoregressive formulation factorizes learning the distribution $P(x)$ as the product of conditional probabilities of current value given all previous values in a pre-defined ordering. This framework has been applied successfully across various domain such as image synthesis (pixel-RNN [22], CPCv2[26]), audio synthesis (Wavenet[25]) and text (GPT[23, 24]). Previous work has also applied a similar autoregressive frameworks to self-supervised modeling of user activity sequences [7, 2, 3, 12], with benefits across various downstream tasks.

Power law based scaling properties for generative pre-training on text datasets using Transformer models was studied in [16]. These properties have been successfully used to create large language models such as GPT-3 [27], GPT-4 [28], PaLM [29], LLaMA [30], etc. Works to establish scaling laws for other data domains such as vision followed in quick succession [17, 18, 19]. More recently, there has been a line of work suggesting that these laws might be less universal than earlier suggested [18]. In addition, methods have been proposed to make the scaling exponential for certain tasks, by either pruning the data effectively [21] or pre-training with a different objective [20].

Self-supervised learning has emerged as an important technique in domains of recommender systems [10], advertising [11] and fraud detection [12]. Particularly, in detection of fraud [12, 13] where we typically observe a lack of precise labels, pre-training representations help avoid label bias. There has been very little work towards exploring scaling behavior in these domains. [14] studied scaling behavior of DLRM style recommender system models across parameters, compute and data to show that unlike text data, model scaling does not contribute as much to performance improvements in recommender systems. Previous works on scaling laws in advertising use CLIP based models and assume data access across multiple domains [15]. To the best of our knowledge, our work is the first to study scaling behavior of Transformers built on user activity sequences alone that uses vanilla autoregressive pre-training and also includes an evaluation of large models on downstream tasks relevant in advertising and fraud detection.

3. Modeling Framework

3.1. Constructing Input Sequences

We order ad events (clicks) from the user based on timestamps to construct the activity sequence. Each event in the sequence is described using multiple features, creating a multi-dimensional time series of the user's ad activity. To handle multiple feature types describing the ad event, we encode each feature using an embedding function which is learnt in an end-to-end manner with the model training objective. Real-valued are converted to categorical using bucketing to tackle the large range. Formally, let S be the n-length sequence of events for a user entity ordered in time, where X_i indicates the event in position i in S. Let $[F_1, F_2,..,F_k]$ be the feature set used for the event description and let $[E_1, E_2,..,E_k]$ be the embedding functions for the corresponding features.

$$S = [X_1, X_2, ..., X_i, ..., X_n] \quad (1)$$

$$X_i = [F_1(i); F_2(i); ...; F_k(i)] \quad (2)$$

The descriptor of each ad event is a concatenation of associated feature embeddings. C_i represents concatenation of these embeddings for the event at position i.

$$C_i = Concat(E_1(F_1(i)), E_2(F_2(i)), ..., E_k(F_k(i))) \quad (3)$$

3.2. Training Objective

The time series of events S represented by their concatenated feature representations C is provided as input to an off-shelf autoregressive deep sequence model. The output representation at the last time step is taken as the output representation of the sequence.

$$R = SequenceModel(C) \quad (4)$$

where R is the output representation (embedding) obtained at the final time step of the model.

The model parameters along with the embedding matrices are trained using next event prediction as the self-supervised objective. At each time step, the model predicts the probability of the next event given only the history, making autoregressive property a necessary condition for the choice of the deep sequence model. We use the Transformer decoder block as the autoregressive model. The goal is to maximize the following likelihood,

$$L(S) = \sum_u \sum_i \log p(X_{i+1} | X_1, .., X_i; \theta) \quad (5)$$

where for each user entity u, $p(X_{i+1}|X_1, .., X_i)$ is the output probability of the next event at each time step and θ corresponds to the model and embedding matrix parameters. Assuming each feature of predicted event to be independent given the history,

$$p(X_{i+1}|X_1, .., X_i) = \prod_{j=1}^{k} p(F_j(i+1)|X_1, .., X_i) \quad (6)$$

For the probability terms corresponding to low cardinality and bucketed real valued feature inputs, full softmax can be computed without any computational bottleneck and cross-entropy of the predicted distribution with the next event feature is used in the loss function.

$$J_{cross-entropy F_j(i)} = H(F_j(i+1), \hat{F}_j(i)) \quad (7)$$

where $H(P, Q)$ is the cross entropy between probability distributions P and Q, $F_j(i+1)$ is the ground truth probability distribution for the j^{th} feature of the next event and $\hat{F}_j(i)$ is its predicted output probability distribution from the softmax function at time step i. To avoid the computational bottleneck in case of high cardinality and natural language features, contrastive predictive coding [5] is used, which classifies the ground truth feature value of the next time step against a set of randomly chosen negative examples directly in the embedding space. The dot product between the predicted embedding and the target embedding (ground truth or negative samples) represents the logits, using which the cross entropy is computed.

$$J_{CPC F_j(i)} = -\log p(E_j(F_j(i+1))|\hat{P}_{i,j}, \{l\})$$
$$= -\log \frac{e^{(E_j(F_j(i+1)))^T \hat{P}_{i,j}}}{e^{(E_j(F_j(i+1)))^T \hat{P}_{i,j}} + \sum_l e^{(E_j(F_j(l)))^T \hat{P}_{i,j}}} \quad (8)$$

where $\hat{P}_{i,j}$ is the prediction for the next time step embedding for the high cardinality / natural language feature F_j and $\{l\}$ are the set of events that form the negative samples.

The final loss function now consists of two parts - exact cross entropy loss for low cardinality features and contrastive loss for high cardinality and natural language features. Let c_j be an indicator variable that takes the value 1 if feature F_j is a high cardinality / natural language feature and 0 otherwise. The self-supervised loss function hence becomes:

$$J_{self-supervised} = \frac{1}{n-1} \sum_{i}^{n-1} \sum_{j=1}^{k}$$
$$((1 - c_j) J_{cross-entropy F_j(i)} + (c_j) J_{CPC F_j(i)}) \quad (9)$$

3.3. Data and Hyperparameters

The dataset consists of user ad click sequences aggregated over a pre-defined time window for a large-scale advertising program. Only sequences above a minimum length are considered and maximum sequence length is bounded to recent n events. We split users into train, validation and test sets in an 80:10:10 ratio. The models train on TensorFlow in a distributed multi-machine setup with NVIDIA V100 GPUs using synchronous weight updates. The loss is computed and optimized using AdamW [35] optimizer with β_1 as 0.9 and β_2 as 0.95. We clip the global norm of the gradients at 1.0. Decoupled weight decay with a rate of 0.1 is applied. Unless otherwise mentioned, we use a fixed learning rate of $1e-4$ after an initial warmup schedule that steadily increases learning rate from 0 to $1e-4$ over the first epoch.

4. Scaling Analysis

4.1. Model Size

We scale the model size in terms of the number of non-embedding trainable parameters in the Transformer by increasing the number of layers, the latent state dimension and number of heads. We vary the number of non-embedding parameters over 4 orders of magnitude and train each model till convergence on the entire training dataset, which is the upper bound of the available data. Table 1 shows the different model configurations and their test loss at convergence. We note that the performance varies only weakly with the individual layer hyperparameters but strongly with the overall model non-embedding parameter count as shown in [16].

Plotting the test loss at convergence follows a power-law relationship with number of non-embedding parameters at constant dataset size, as shown in Figure 1. We extrapolate the power-law trend observed between 50k parameters and 25M parameters to 85M parameters and highlight that the estimated test loss of 7.429 closely matches the experimental value of 7.425. This implies

Table 1
Parameter Scaling Configurations (global batch size and number of epochs set constant at 16384 and 20 respectively)

Parameters (non embedding)	Layers	Latent Dimension	Heads	Test Loss
54,240	4	32	4	7.631
410,240	2	128	4	7.595
3,186,432	4	256	8	7.535
6,359,552	2	512	4	7.478
12,664,320	4	512	8	7.493
25,273,856	8	512	8	7.466
85,136,640	12	768	8	7.425

that even at 85M parameters, we are not bottlenecked by the dataset size, indicating that a billion-scale parameter model is unlikely to overfit due to a data bottleneck. However, we acknowledge that this trend must eventually saturate, beyond which it would not be useful to further increase model size under the current training framework, as we are upper bounded in terms of organically available data.

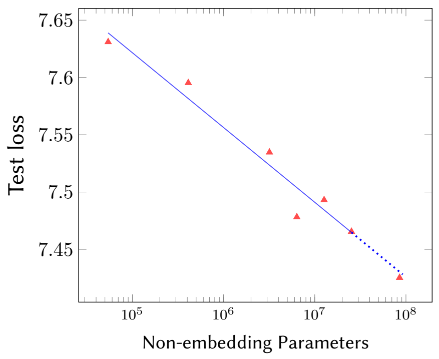

Figure 1: Parameter Scaling

4.2. Data Size

We analyze the impact of data scaling by considering different train dataset sizes, created by considering 0.1%, 1%, 25% and 100% of available user sequence data. The learning rate is kept constant and we scale number of GPUs with increased model size to keep the global batch size constant. We train three model sizes, with 410K, 3.1M and 12.6M trainable parameters, until convergence on varying dataset sizes and plot their test loss in Figure 2.

We obtain three key insights from Figure 2 - first, we observe that larger models are more data efficient. That is, larger models require a smaller dataset to achieve a fixed test loss. Second, smaller models benefit more from increasing dataset size when compared to larger models.

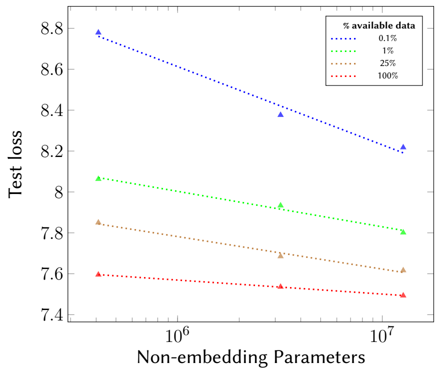

Figure 2: Data Scaling

Finally, for a fixed model size, increasing dataset size shows diminishing returns in terms of test loss improvement, suggesting that to maximize performance, model size and dataset size must be scaled in tandem. However, in the practical setting of activity sequence models, where the dataset size is upper bounded, it would still be useful to train the largest possible model to maximize performance within the bounds suggested in Section 4.1. As larger model training requires significantly more compute, we explore the compute allocation strategy in the next section.

4.3. Compute

In industry settings, training of models is bounded by monetary constraints. We use wall-clock GPU-hours on a homogenous GPU setup (NVIDIA V100s) as the measure of compute as against the standard PetaFLOP-days, as the monetary cost incurred to train a model in a standard cloud setup is a function of GPU wall-clock time usage and not GPU utilization. In this section, we explore for a fixed budget (monetary value or equivalent GPU-hours), the efficient scaling strategy for model size and data parallelism (global batch size) to achieve the lowest possible test loss. Scaling up model size at a fixed global batch size would require more GPUs to run in parallel, reducing the number of serial gradient update steps that can performed in a fixed GPU-hour budget. Alternatively, one could reduce global batch size and number of parallel GPUs for a model and increase the number of serial steps. We analyze this trade-off by fixing the number of GPU-hours and varying the configuration across different model sizes and global batch sizes in a way that GPU utilization stays maximized. . We plot the test loss for different model sizes at maximum GPU utilization in Figure 3 for configurations detailed in Table 2. We note that the learning rate is scaled proportionately with the global batch size [36].

Table 2
Compute configurations at 16 GPU-hours

Configuration	GPUs	Time (minutes)	Learning rate
1	64	15	0.0008
2	32	30	0.0004
3	16	60	0.0002
4	8	120	0.0001

Table 3
Benchmarking large models at batch size < B_{min}

Parameters	GPUs	Time (minutes)	Batch size (per GPU)	Test loss
201,649,152	8	120	48	7.732
85,136,640	8	120	128	7.601
25,273,856	8	120	256	7.554
410,240	8	120	960	7.677

We draw two key insights from Figure 3 - first, for fixed number of parallel GPUs and wall-clock time, larger models reach a lower test loss - indicating that sample efficiency (1/(number of serial gradient updates $\times$ global batch size)) increases with model size for a target test loss. Second, for all model sizes, increasing number of serial gradient updates is more effective than increasing batch size. Hence, the efficient scaling strategy would suggest scaling up model size while lowering the global batch size for a given compute budget. However, reducing batch size to extremely low numbers would make the gradient updates noisier. We empirically demonstrate in Table 3 that lowering batch size below a minimum threshold B_{min} for a larger model leads to worse performance than a smaller model at fixed compute.

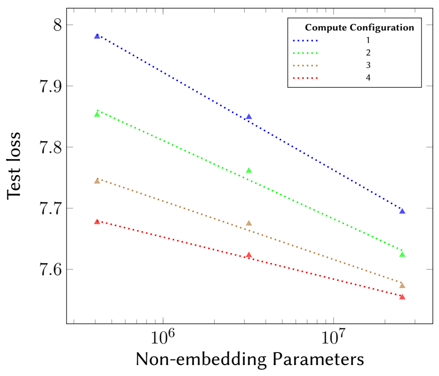

Figure 3: Compute Scaling

While scaling up model size at B_{min} ensures efficient allocation of compute between data parallelism and serial steps, a larger model at B_{min} requires a certain wall clock time before its loss outperforms smaller models due to more serial gradient steps in smaller models early on in the training. We define W_{min} as the minimum wall clock time required for a model with batch size B_{min} to outperform all smaller models trained at their individual B_{min} configuration for the same wall clock time.

We empirically demonstrate the existence of W_{min} in Figure 4, where the larger 25M parameter model at batch size 16k eventually achieves a lower test loss than smaller 410k parameter model with a more compute efficient configuration of batch size 5k, where both batch sizes are greater than B_{min}. Further extending to a fixed compute budget, Table 4 demonstrates that a larger 25M parameter model with batch size 8K achieves a lower test loss at the end of 30 minutes compared to a smaller 410K parameter model with batch size 7K at the end of 120 minutes on the wall-clock, indicating that the W_{min} for the 25M parameter model lies within the regime of the allocated compute budget even when data parallelism for the larger model was set at a more inefficient configuration than the smaller model.

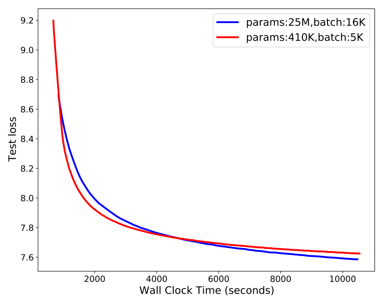

Figure 4: Larger model outperforms smaller model after W_{min} wall-clock time

Table 4
Compute efficiency when B_{min} and W_{min} are satisfied at fixed GPU-hours

Parameters	GPUs	Time (minutes)	Batch size (global)	Test loss
25,273,856	32	30	8192	7.623
410,240	8	120	7680	7.677

Hence, the efficient use of compute requires training the largest possible model for which both W_{min} and B_{min} are supported within the given compute budget.

5. Downstream Task Evaluation

We evaluate the performance of the learnt user representations on two downstream tasks - first, where accurate labels are available for training a classifier and another where no task specific fine-tuning is possible due to lack of labels.

5.1. Linear Separability in Classification

In this experiment, we benchmark the user embeddings on the user conversion prediction task based on linear separability. We train a linear binary classifier on the learnt user embeddings (output of the last timestep in the sequence) to predict if the user converts, and evaluate the efficacy based on AUC-ROC. Higher AUC-ROC implies that the embeddings have better linear separation with respect to the downstream conversion label.

5.2. Click bot detection

Due to absence of accurate ground truth labels, supervised techniques fall short in bot detection scenarios. While labeling individual samples accurately may not be possible, multiple domain-knowledge based heuristics can be applied to reliably evaluate if a given group of users are robotic. Hence, we cluster self-supervised user embeddings using k-means and clusters of users based on these heuristics are marked as robotic.

We calibrate the heuristics to achieve a fixed False Positive Rate (FPR), which refers to the fraction of genuine human traffic flagged as robotic by the algorithm. Since we do not have ground truth labels, FPR is approximated by using converting users as a proxy for the distribution of human labels. The fraction of converting clicks that were marked as robotic is computed as FPR. We also define Invalidation Rate (IVR) as the fraction of total ad clicks flagged as robotic by the algorithm at the program level. For a fixed operating point FPR, the model with higher IVR indicates better robotic recall.

5.3. Results

We consider embeddings from models described in Section 4.1, where we scale the non-embedding parameter count over 4 orders of magnitude on the entire training data and train till convergence. Table 5 shows the downstream performance of the models on the conversion prediction and the robot detection tasks.

Unsurprisingly, lower test loss of the larger models translates to better downstream performance for both supervised task of conversion prediction and unsupervised task of robot detection. We note that scaling patterns on downstream tasks do not necessarily follow the power law, making it challenging to predict the potential performance gains from a larger size model apriori.

Table 5
Lift over downstream task performance relative to 54K model

Params	IVR @ fixed FPR	pConversion AUC
3,186,432	+1.63 %	+0.02 %
6,359,552	+3.44 %	+2.51 %
85,136,640	+4.09 %	+3.57 %

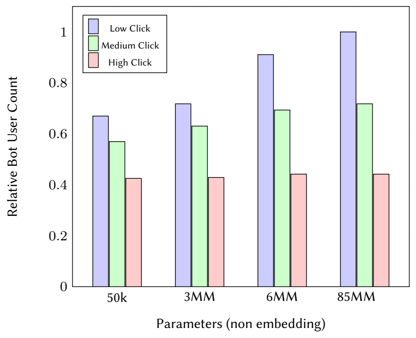

Figure 5: Relative count of bot accounts flagged across click sequence lengths

Figure 5 shows the relative count of bot accounts flagged by individual models, split into different click sequence length buckets. It is evident that the larger models are highly effective in identifying bot activity with low click bucket bot detection improving by 42% and medium click bot detection improving by 20% across the model sizes considered. This indicates that larger models are able to learn better representations for smaller sequence lengths and help disambiguate more sophisticated bot patterns with limited data.

6. Discussion

We show that the test loss of activity sequence models trained using generative pre-pretraining follows a power-law relationship with model size at constant dataset size, similar to observations made in text, images and audio domains [16, 17, 34]. Unlike text and images domains where increasing dataset size is relatively easier by gathering data from the web, user activity sequence datasets have a hard upper bound on dataset size, governed by number of users interacting with the ad program. Thus, increasing model sizes would eventually lead to overfitting, saturating the power law curve. However, our data scaling experiments show that present model sizes do not show saturating behavior even on 1% dataset size, indicating that there is significant room for model scaling at our current dataset size. We also show that larger models are more data efficient, achieving a lower test loss at fixed dataset size, consistent with the trends observed in text and image domain [16, 17] with a key distinction that smaller models benefit more from increased data in the activity sequence domain.

As monetary constraints are a key consideration in compute scaling in most industrial settings, we presented a strategy to allot fixed GPU-hours across model size and global batch size. In contrast to observations in natural language models [16], we observe that scaling serial gradient update steps are more effective than batch size, as long as the batch size is above B_{min}. Compute efficient training of activity sequence models involves limiting the number of GPUs such that a global batch size of B_{min} is achieved, and picking a model size such that training is performed for at least W_{min} wall clock time. Thus, compute efficient training stops far short of convergence, as highlighted to also be the case in natural language and computer vision models. While larger models have been shown to be sample efficient [16, 17, 34], we show that the same translates to activity sequence models, even under an additional constraint of fixed GPU-hours.

Finally, we show performance on downstream tasks of bot detection and conversion prediction improves with generative pre-training of larger model sizes. While we obtain performance gains, they do not follow a power law relationship, making it difficult to predict performance gains on business tasks with model size scaling. This observation is also consistent with findings in the text domain where just scaling model size has shown significant improvements in downstream task performance [28] that may not always follow the power law.

7. Conclusion and Future Work

We presented model, data and compute based scaling properties for generative pre-training of user activity sequence Transformer models and demonstrated how scaling translates to better next event prediction efficacy which in turn leads to better downstream performance on advertising tasks.

In future work we plan to to study scaling properties with respect to activity sequence lengths, by using longer time windows as a mechanism to scale the current bounded dataset size. We will also experiment with more efficient training strategies that help improve over the current power law, while reducing training costs. Finally, with recent work on joint representation learning of time-varying sequence data and fixed tabular data using masked language modeling [12], we will attempt to study if scaling properties from this work also generalize to other pre-training objectives.

References

[1] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pp. 5998-6008. 2017.

[2] Agarwal, Rajat, Shailendra Agarwal, Agniva Som, and Hemant Kowshik. Using Customer Ad Click Sequences to Identify Invalid Traffic in Sponsored Products. In Amazon Machine Learning Conference, 2020.

[3] Agarwal, Rajat, Agniva Som, Arvind Srinivasan, Jerin Francis, Anand Muralidhar, and Hemant Kowshik. Self-supervised Representation Learning for User Ad Activity Sequences. In Amazon Machine Learning Conference, 2021.

[4] Gligorijevic, Djordje, Jelena Gligorijevic, and Aaron Flores. Time-Aware Prospective Modeling of Users for Online Display Advertising. arXiv preprint arXiv:1911.05100 (2019).

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

[6] He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021).

[7] Liao, Yiping. On the Effectiveness of Self-supervised Pretraining for Modeling User Behavior Sequences. In AdKDD, 2020.

[8] Naumov, Maxim, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang et al. Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091 (2019).

[9] Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. Tensorflow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265-283. 2016.

[10] Yao, Tiansheng, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong et al. "Self-supervised learning for large-scale item recommendations." In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4321-4330. 2021

[11] Guo, Wei, Can Zhang, Zhicheng He, Jiarui Qin, Huifeng Guo, Bo Chen, Ruiming Tang, Xiuqiang He and Rui Zhang. “MISS: Multi-Interest Self-Supervised Learning Framework for Click-Through Rate Prediction.” 2022 IEEE 38th International Conference on Data Engineering (ICDE) (2021): 727-740.

[12] Agarwal, Rajat, Anand Muralidhar, Agniva Som and Hemant Kowshik. “Self-supervised Representation Learning Across Sequential and Tabular Features Using Transformers.” (2022).

[13] Chitlangia, Sharad, Anand Muralidhar and Rajat Agarwal. “Self Supervised Pre-training for Large Scale Tabular Data.” (2022).

[14] Ardalani, Newsha, Carole-Jean Wu, Zeliang Chen,

Bhargav Bhushanam and Adnan Aziz. “Understanding Scaling Laws for Recommendation Models.” ArXiv abs/2208.08489 (2022): n. pag.

[15] Shin, Kyuyong, Hanock Kwak, KyungHyun Kim, Su Young Kim and Max Nihl’en Ramstrom. “Scaling Law for Recommendation Models: Towards General-purpose User Representations.” ArXiv abs/2111.11294 (2021): n. pag.

[16] Kaplan, Jared, Sam McCandlish, T. J. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu and Dario Amodei. “Scaling Laws for Neural Language Models.” ArXiv abs/2001.08361 (2020): n. pag.

[17] Zhai, Xiaohua, Alexander Kolesnikov, Neil Houlsby and Lucas Beyer. “Scaling Vision Transformers.” 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021): 1204-1213.

[18] Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and L. Sifre. “Training Compute-Optimal Large Language Models.” ArXiv abs/2203.15556 (2022): n. pag.

[19] Henighan, Tom, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun et al. "Scaling laws for autoregressive generative modeling." arXiv preprint arXiv:2010.14701 (2020).

[20] Tay, Yi, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung et al. "Ul2: Unifying language learning paradigms." In The Eleventh International Conference on Learning Representations. 2022.

[21] Sorscher, Ben, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari Morcos. "Beyond neural scaling laws: beating power law scaling via data pruning." Advances in Neural Information Processing Systems 35 (2022): 19523-19536.

[22] Van Den Oord, Aäron, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." In International conference on machine learning, pp. 1747-1756. PMLR, 2016.

[23] Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." (2018).

[24] Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." OpenAI blog 1, no. 8 (2019): 9.

[25] Oord, Aaron van den, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio." arXiv preprint arXiv:1609.03499 (2016).

[26] Henaff, Olivier. "Data-efficient image recognition with contrastive predictive coding." In International conference on machine learning, pp. 4182-4192. PMLR, 2020.

[27] Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.

[28] OpenAI. “GPT-4 Technical Report.” ArXiv abs/2303.08774 (2023): n. pag.

[29] Chowdhery, Aakanksha, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov and Noah Fiedel. “PaLM: Scaling Language Modeling with Pathways.” ArXiv abs/2204.02311 (2022): n. pag.

[30] Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur’elien Rodriguez, Armand Joulin, Edouard Grave and Guillaume Lample. “LLaMA: Open and Efficient Foundation Language Models.” ArXiv abs/2302.13971 (2023): n. pag.

[31] Creswell, Antonia, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. "Generative adversarial networks: An overview." IEEE signal processing magazine 35, no. 1 (2018): 53-65.

[32] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[33] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." Advances in Neural Information Processing Systems 33 (2020): 6840-6851.

[34] Pu, J., Yang, Y., Li, R., Elibol, O., Droppo, J. (2021) Scaling Effect of Self-Supervised Speech Models. Proc. Interspeech 2021, 1084-1088, doi: 10.21437/Interspeech.2021-1935

[35] Loshchilov, Ilya, and Frank Hutter. "Decoupled weight decay regularization." arXiv preprint arXiv:1711.05101 (2017).

[36] Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. "Accurate, large minibatch sgd: Training imagenet in 1 hour." arXiv preprint arXiv:1706.02677 (2017).