Exploiting Variable Correlation with Masked Modeling for Anomaly Detection in Time Series

Anonymous Author(s)
Affiliation
Address
email

Abstract

Online anomaly detection in multi-variate time series is a challenging problem particularly when there is no supervision information. Autoregressive predictive models are often used for this task, but such detection methods often overlook correlations between variables observed in the most recent step and thus miss some anomalies that violate normal variable relations. In this work, we propose a masked modeling approach that captures variable relations and temporal relations in a single predictive model. Our method can be combined with a wide range of predictive models. Our experiment shows that our new masked modeling method improves detection performance over pure autoregressive models when the time series itself is not very predictable.

1 Introduction

Unsupervised anomaly detection is the problem of detecting data points that do not conform to a normal data distribution (Chandola, Banerjee, and Kumar, 2009; Chalapathy and Chawla, 2019). In many applications, there are abundant anomaly-free training data but no information about possible anomalies. In the context of sequential data, novelties usually manifest as events that appear "out of context" or that do not follow the typical time-dependent data generating process (Cook, Mısırlı, and Fan, 2019; Blázquez-García et al., 2021; Choi et al., 2021). The ability to detect novelty in time series data in an online manner is very useful in real-time monitoring systems. In our motivating application, an agent in a game needs to detect abnormal game transitions and adjust game strategy in real time. In this work, we focus on online anomaly detection in time series and assume no access to anomalies.

Autoregressive predictive models are often used for anomaly detection. Specifically, the predictive model takes previous observations $\mathbf{x}_{<t} = (\mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$ and makes a prediction $\hat{\mathbf{x}}_t$ of the current step. Then the received observation $\mathbf{x}_t$ at the current time step is compared against the prediction: if $\mathbf{x}_t$ deviates much from the prediction, then $\mathbf{x}_t$ is claimed to be anomalous. The detection performance largely depends on the accuracy of the predictive model. Research in this direction has proposed various methods to improve the predictive model of such detection system (Malhotra et al., 2015; Hundman et al., 2018; Munir et al., 2018; Deng and Hooi, 2021; Chen et al., 2021). However, these methods often overlook correlations between variables observed in the most recent step and thus miss some anomalies that violate normal variable relations. In our motivating applications, such are abundant as game rules enforce strong correlations between variables and stochasticity in the game makes the time-series difficult to predict.

In this work, we devise a masking strategy that allows any predictive model to capture not only temporal relations but also correlations between variables in the same time step. Inspired by masked language modeling (Devlin et al., 2018), we withhold data entries at the current time step $\mathbf{x}_t$, then we

Submitted to Workshop on Robustness in Sequence Modeling at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

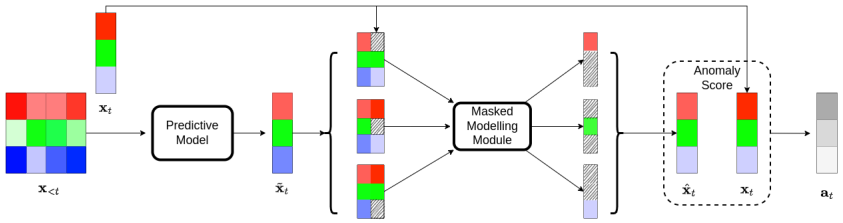

Figure 1: Illustration of masked modeling strategy. The current observed variables in $\mathbf{x}_t$ are masked K times, and each mask (dashed) produces a separate prediction. These predictions are assembled to form the final prediction, which is compared to $\mathbf{x}_t$ to get the anomaly score.

train the model to recover masked entries based on other entries as well as previous observations. In this new approach, the model is able to capture relations between data entries within $\mathbf{x}_t$.

We perform experiments on our game datasets and show that our method improves the detection performance of a predictive model. Along with detection, we also measure *localization* performance, that is the ability to identify anomalous variables in the input vector, rather than just the input vector as a whole.

2 Related Work

Autoregressive predictive models are commonly used for anomaly detection in the online setting Malhotra et al. (2015). These models are commonly implemented with either a fixed window size over temporal context or a recurrent neural network (RNN) (Filonov, Lavrentyev, and Vorontsov, 2016). Deng and Hooi (2021) improve the predictive model in the anomaly detection task: the proposed model learns a graph over features to better capture relationships between the sensors in time series. However, they solely rely on past observations to predict observations at the current step.

3 Masked Autoregressive Training

3.1 Problem Setup

Suppose we have a normal system that generates time series $\mathbf{X} = (\mathbf{x}_\tau : \tau = 1, \ldots, T)$ according to some unknown distribution $p^*(\mathbf{x}_\tau|\mathbf{x}_{<\tau})$. Here the observation $\mathbf{x}_\tau \in \mathbb{R}^d$ at each time step τ is a vector of d variables. In the problem of *online anomaly detection* in time series, we only observe the time series $\mathbf{x}_{\leq t} = (\mathbf{x}_\tau : \tau = 1, \ldots, t)$ up to t, and the task is to decide whether $\mathbf{x}_t$ follows the same underlying distribution $p^*(\mathbf{x}_t|\mathbf{x}_{<t})$. Note that there are no training examples of anomalies, so this is an unsupervised problem.

In many applications, we are interested whether a single entry $x_{t,i}$ is from the model $p^*(\mathbf{x}_{t,i}|\mathbf{x}_{t,\eta\setminus i}, \mathbf{x}_{<t})$. Here the subscript $\setminus i$ denotes all entries in a vector except i and η denotes the set of normal entries in $\mathbf{x}_t$. While η is also time dependent, we omit the subscript to simplify the notation.

In online anomaly detection, a typical method is to train a predictive model $p(\mathbf{x}_\tau|\mathbf{x}_{<\tau})$. Then, an observation $\mathbf{x}_t$ is compared to the prediction $\hat{\mathbf{x}}_t$ from $p(\mathbf{x}_t|\mathbf{x}_{<t})$. If $\mathbf{x}_t$ deviates significantly from the prediction, then it is decided that $\mathbf{x}_t$ contains anomalous entries. Similarly, a single entry $x_{t,i}$ is compared to the prediction $\hat{x}_{t,i}$ to decide whether $x_{t,i}$ is anomalous. Typically, this method implicitly assumes entries $x_{t,i}$ are conditionally independent given $\mathbf{x}_{<t}$. In our motivating application of modeling game data where $\mathbf{x}_t$ represents a game state at time t, two variables $x_{t,i}$ and $x_{t,j}$ might look normal according to their marginal distributions $p(x_{t,i}|\mathbf{x}_{<t})$ and $p(x_{t,j}|\mathbf{x}_{<t})$ but are anomalous according to their joint distribution $p(x_{t,i}, x_{t,j}|\mathbf{x}_{<t})$. This type of anomaly is only obvious when we model the relation between $x_{t,i}$ and $x_{t,j}$.

3.2 Masked modeling for detecting and locating anomalies

The goal of our method is to approximate the true data distribution $p^*(x_{t,i}|\mathbf{x}_{t,\eta\setminus i}, \mathbf{x}_{<t})$. The two main challenges we face are determining the normal set of variables η to condition on and approximating the conditional with a neural model. The first problem is addressed by an *error-based filter* that approximates the set η using an underlying autoregressive model $g_\psi(\cdot)$. This model can be either pretrained and fine-tuned with our module, or trained from scratch in an end to end manner. To address the second problem we notice that the form of p^* has structure analogous to masked modeling and employ a *masking procedure* that allows us to efficiently fit the distribution.

Error-based filter As highlighted earlier, $x_{t,i}$ depends on the temporal context $\mathbf{x}_{<t}$ and the values of *normal observations* $\mathbf{x}_{t,\eta\setminus i}$. Identifying the set η is not trivial, as it is tantamount to solving the full anomaly localization problem. However, we can achieve an initial estimate by predicting the current step $\tilde{\mathbf{x}}_t$ using *only* the temporal context $\mathbf{x}_{<t}$ and then comparing it against current observations $\mathbf{x}_t$: those entries with large predictive errors will not be in our approximation of η.

Let the predictive model be

$$\tilde{\mathbf{x}}_t = g_\psi(\cdot). \tag{1}$$

The predictive model can be implemented with RNNs and alike. Let $\mathbf{e}_t$ be the error vector containing the predictive error at each entry, then the binary representation $\boldsymbol{\eta}$ of the set η is computed as follows.

$$\mathbf{e}_t = \mathrm{err}(\mathbf{x}_t, \tilde{\mathbf{x}}_t) \tag{2}$$

$$\boldsymbol{\eta} = \mathbf{e}_t > r \cdot \max_j e_{t,j} \tag{3}$$

Again we omit the subscript t from $\boldsymbol{\eta}$ for simplicity. Here $r \in \mathbb{R}$ is a hyperparameter. Importantly, this is not how we detect anomalies, as the second step of the module will leverage current-step information to improve the prediction of $x_{t,i}$. Instead, the error filter allows the module to base its final prediction only on normal observed entries in η by removing suspicious entries. Using the error filter we can produce $\mathbf{x}_t^*$ which corresponds to $\mathbf{x}_{t,\eta}$ and treats variables not in η as missing values. There are various ways to impute those values but for simplicity we impute with the corresponding values of $\mathbf{x}_{t-1}$.

Masked modeling module We then approximate the conditional distribution $p(x_{t,i}|\mathbf{x}_{t,\eta\setminus i}, \mathbf{x}_{<t})$ using the masking mechanism. The main challenge here is the computational cost: while an autoregressive model $p(\mathbf{x}_t|\mathbf{x}_{<t})$ only needs to predict once, the masked model needs to predict each variable in $\mathbf{x}_t$ separately. To address this issue, we consider two strategies: using a shared model to predict all variables and compressing information $\mathbf{x}_{<t}$ once for all predictions.

Let $f_\phi(\cdot)$ denote the shared predictive module that has d outputs and predicts all variables in $\mathbf{x}_t$. Here ϕ denotes all trainable parameters of the module. To prepare the input, we first impute values not in η for the module. There are different ways to impute these values (e.g. using zeros). Here we choose to use values from the previous step.

$$\mathbf{x}'_t = \boldsymbol{\eta} \odot \mathbf{x}_t + (1 - \boldsymbol{\eta}) \odot \mathbf{x}_{t-1} \tag{4}$$

Then the masked predictive module $f_\phi(\cdot)$ works as follows. Let $\mathbf{m}_i = \mathrm{onehot}(i)$ denote the one-hot encoding of i.

$$\hat{\mathbf{x}}_t = \sum_{i=1}^{d} \mathbf{m}_i \odot f_\phi([(1 - \mathbf{m}_i) \odot \mathbf{x}'_t, \tilde{\mathbf{x}}_t, \boldsymbol{\eta}]). \tag{5}$$

Here $\odot$ represents element-wise multiplications. We also feed $\boldsymbol{\eta}$ to the module to indicate imputed values. The shared network f_ϕ takes a simple MLP architecture. It runs efficiently even though it predicts d times in total for all variables in $\mathbf{x}_t^*$. Note that the information about $\mathbf{x}_{<t}$ is encoded in the prediction $\tilde{\mathbf{x}}_t$ of the autoregressive model, and *is only computed once*. Additional running time considerations are discussed in the appendix section.

Training loss and Anomaly score The training loss of the model is still the same as for an autoregressive model:

$$L_\phi = \sum_{t=2}^{T} \sum_{i=1}^{d} \mathrm{loss}(\mathbf{x}_{t,i}, \hat{\mathbf{x}}_{t,i}) \tag{6}$$

Here the loss function can adapt to different types of variables. For example, it can be the squared difference if $x_{t,i}$ is a numerical variable or the binary cross-entropy loss if $x_{t,i}$ is a binary variable. A similar calculation is also used for compute anomaly scores at test time for each variable, which are aggregated to form the anomaly score for the time step. For more details please refer to the appendix.

4 Experiments

We conduct 2 experiments to evaluate the performance of models with the masking module on anomaly detection and localization. An ablation of the error filter is available in the appendix.

Model	Polycraftv2			Monopoly		
	F1	**Precision**	**Recall**	**Best F1**	**Precision**	**Recall**
MLP	0.902±0.006	0.961±0.008	0.851±0.009	0.534±0.005	0.423±0.009	0.73±0.021
Masked MLP	**0.976**±0.004	0.989±0.003	0.964±0.007	**0.824**±0.004	0.967±0.005	0.718±0.007
GDN	0.915±0.008	0.875±0.006	0.96±0.014	0.528±0.007	0.46±0.004	0.622±0.019
Masked GDN	**0.962**±0.002	0.949±0.007	0.976±0.008	**0.824**±0.005	0.97±0.014	0.718±0.006

Table 1: Anomaly detection performances over 10 runs on Polycraftv2 and Monopoly. F1 scores are in bold when masked training yields statistically significant difference in performance.

Model	Polycraftv2			Monopoly		
	F1	**Precision**	**Recall**	**Best F1**	**Precision**	**Recall**
MLP	0.43±0.004	0.803±0.06	0.306±0.017	0.458±0.004	0.369±0.005	0.608±0.016
Masked MLP	**0.615**±0.017	0.637±0.042	0.619±0.032	**0.588**±0.004	0.941±0.011	0.427±0.004
GDN	0.616±0.003	0.74±0.043	0.55±0.03	0.496±0.003	0.404±0.007	0.647±0.01
Masked GDN	**0.67**±0.006	0.874±0.021	0.545±0.006	**0.608**±0.012	0.946±0.017	0.449±0.012

Table 2: Anomaly localization performances over 10 runs on Polycraftv2 and Monopoly. F1 scores are in bold when masked training yields statistically significant difference in performance.

Datasets We collect data from two games Polycraftv2 (Smaldone et al., 2017) and Monopoly (Haliem et al., 2021). Polycraftv2 is an open environment where agents collect resources and craft items. Monopoly is a simulation derived from the well-known board game Monopoly. The games are played by planning agents based on the DIARC architecture(Schermerhorn et al., 2007). Anomalies in games include changes of rules, objects, or entity relations in the game. We perform evaluation on both novelty detection and localization. For novelty detection, we use the positive label for anomalous time steps. For novelty localization we use the positive label for anomalous features. Table 4 in the appendix summarizes the dataset statistics for each game.

Experiment settings We augment two autoregressive models with our module: multilayer-perceptron (MLP) and GDN (Xu et al., 2021). Further information on training settings is available in the appendix. We evaluate anomaly detection by comparing step-wise anomaly scores from different models against anomaly labels. Following previous work, we set the threshold to the maximum error over the validation set and compute the F1 score for each model. We also show the precision and recall at this threshold. For anomaly localization, we take the micro-average of F1 scores of all variables at different times: we pool anomaly scores for all variables and then compare them against true labels to compute the F1 score, precision, and recall.

Results Results on anomaly detection are shown in table 1 and anomaly localization in table 2. Both games have strong correlations between features as enforced by game rules, and sources of stochasticity that make the time-series difficult to predict. In Polycraftv2, resource amounts are strongly correlated as some are used to directly craft others. Similarly, in Monopoly the player's cash is strongly correlated with property ownership. The stochasticity in the games comes from random effects such as dice rolls and agent's decision-making. As a result, these conditions allow the masking module to offer statistically significant improvement in F1 score for both models in both environments.

5 Supplement

5.1 Anomaly Score calculation

For any predictive model including those with our module, we compare model predictions to observations to calculate the anomaly score.

We first consider locating anomalies to individual variables. Here we compare $x_{t,i}$ and $\hat{x}_{t,i}$ to get the anomaly score $\alpha_{t,i}$ for entry i at time t. Let $r_{t,i} = \mathrm{err}(x_{t,i}, \hat{x}_{t,i})$ be the predictive error with some type of error function. For example, it can be the absolute error if $x_{t,i}$ is continuous. We also compute a single anomaly score a_t for the entire time step t by aggregating anomaly scores of single entries.

$$a_t = \mathrm{aggregate}(\alpha_{t,1}, \ldots, \alpha_{t,d}). \tag{7}$$

There are various choices for the aggregation function. For example, we can take the average or maximum of the d anomaly scores. In our experiments, we use the maximum as the step-wise anomaly score. Figure 1 illustrates the architecture.

5.2 Running Time Considerations

If the number d of variables is very large, variables can be grouped together in masking to further reduce the required computation. Suppose we create k groups from d variables and denote each group with a binary vector.

$$M = \{\mathbf{m}_{k'} \in \{0,1\}^d : k' = 1, \ldots, k\}, \quad \sum_{k'=1}^{k} \mathbf{m}_{k'} = \mathbf{1} \tag{8}$$

Then the conditional distribution we need to fit is $p^*(\mathbf{x}_{t,\mathbf{m}_k} | \mathbf{x}_{t,(1-\mathbf{m}_k)\odot\eta}, \mathbf{x}_{<t})$, which is a generalization of the conditional distribution of a single variable. The prediction $\hat{\mathbf{x}}_t$ is computed in the same way as (5) except that the summation loops over k masks.

However, in practice we find that for non-time critical applications the lightweight architecture of the module means the extra computational cost is not very significant. Table 3 shows per training epoch running times for models with the masked module *without* variable grouping.

Models	**Polycraftv2** (s)	**Monopoly** (s)
MLP	4.65 ± 0.75	8.36 ± 0.44
Masked MLP	5.45 ± 0.36	9.64 ± 0.38
GDN	9.19 ± 1.32	17.87 ± 1.87
Masked GDN	11.43 ± 1.04	20.32 ± 2.47

Table 3: Per epoch running time comparison for all models in Polycraftv2, and Monopoly

5.3 Dataset Statistics

Table 4 shows the statistics of the game datasets.

5.4 Training information and Hyperparameters

The game datasets include binary, numerical and categorical features. Categories in categorical features are encoded as vectors. Numerical features are normalized by subtracting their training data mean and dividing by the variance. To scale up the training, all models use fixed window size for the temporal context and are trained for a maximum of 200 epochs with early stopping. We run each model 10 times and collect the mean.

Common Training parameters: Adam, learning rate 1e-3, beta1,beta2= (0.9,0.99). Early stopping with patience 15.

Dataset : Polycraftv2
GDN: Embedding dimension 64, topk 5, Hidden dimension 16, dropout=0.2

Dataset	# variables	# Train	# Test	% Anomalous Steps	% Anomalous Variables
Polycraftv2	26	3461	993	10	1.1
Monopoly	9	7393	2148	10	2.2

Table 4: Statistics of two game datasets.

Masked GDN: masks 10, Embedding dimension 64, topk 15, Hidden dimension 64

MLP: Hidden dimension 32, last layer dimension 16, number of hidden layers 4, dropout 0.2

Masked MLP: masks 10, Hidden dimension 16, last layer dimension 16, number of hidden layers 4, dropout 0.2

Dataset : Monopoly
GDN: Embedding dimension 64, topk 5, Hidden dimension 16, dropout=0.2

Masked GDN: masks 19, Embedding dimension 64, topk 15, Hidden dimension 32

MLP: Hidden dimension 256, last layer dimension 64, number of hidden layers 2, dropout 0.2

Masked MLP: masks 19, Hidden dimension 64, last layer dimension 64, number of hidden layers 2, dropout 0.2

Additional Experiments

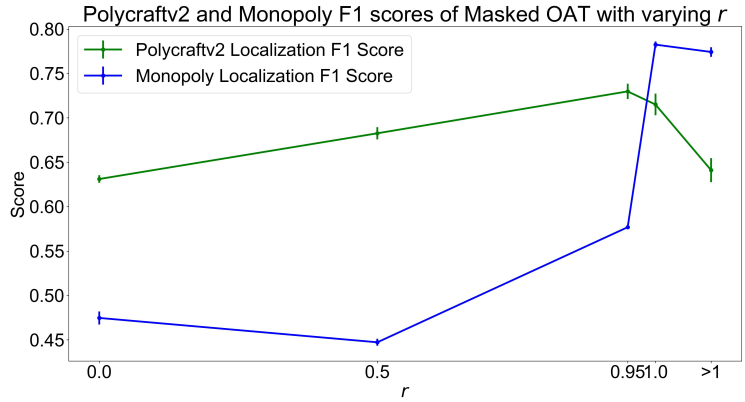

Figure 2: Polycraftv2 and Monopoly localization performance for varying error filter opacity. When $r = 0$ the filter is opaque, when $r > 1$ the filter is transparent.

5.4.1 Ablation of error filter

In this experiment we perform ablation of the error filter by varying the value of r. We would like to note that this experiment was conducted after the completion of all previous experiments and did not influence the selection of r.

Experiment Settings We perform the ablation study on the anomaly localization tasks for monopoly and Polycraftv2 as these tasks are particularly susceptible to the problem of conditioning on anomalies. We vary the hyperparameter r to values of $0, 0.5, 0.95, 1.0$ and also completely remove the filter by setting $r > 1$ thus using the full current step.

Results The results shown in Figure 2 demonstrate that the error filter is necessary to extract the full benefit of the masked modeling module. In both games, when $r > 1$ the model suffers in performance as it conditions predictions on anomalous entries. When r is small, the predictive power of the model is decreased so detection performance suffers. In Polycraftv2 $r = 0.95$ provides the best trade-off. In Monopoly $r = 1$ provides the best trade-off. While our chosen value of $r = 0.95$ seems reasonable across tasks, the optimal choice is application dependent and not easily tuned.

References

Blázquez-García, A.; Conde, A.; Mori, U.; and Lozano, J. A. 2021. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3): 1–33.

Chalapathy, R.; and Chawla, S. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3): 1–58.

Chen, Z.; Chen, D.; Yuan, Z.; Cheng, X.; and Zhang, X. 2021. Learning Graph Structures with Transformer for Multivariate Time Series Anomaly Detection in IoT. *CoRR*, abs/2104.03466.

Choi, K.; Yi, J.; Park, C.; and Yoon, S. 2021. Deep Learning for Anomaly Detection in Time-Series Data: Review, Analysis, and Guidelines. *IEEE Access*.

Cook, A. A.; Mısırlı, G.; and Fan, Z. 2019. Anomaly detection for IoT time-series data: A survey. *IEEE Internet of Things Journal*, 7(7): 6481–6494.

Deng, A.; and Hooi, B. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4027–4035.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Filonov, P.; Lavrentyev, A.; and Vorontsov, A. 2016. Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. *CoRR*, abs/1612.06676.

Haliem, M.; Bonjour, T.; Alsalem, A. O.; Thomas, S.; Li, H.; Aggarwal, V.; Bhargava, B.; and Kejriwal, M. 2021. Learning Monopoly Gameplay: A Hybrid Model-Free Deep Reinforcement Learning and Imitation Learning Approach. *ArXiv: 2103.00683*.

Hundman, K.; Constantinou, V.; Laporte, C.; Colwell, I.; and Soderstrom, T. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 387–395.

Malhotra, P.; Vig, L.; Shroff, G.; Agarwal, P.; et al. 2015. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89, 89–94.

Munir, M.; Siddiqui, S. A.; Dengel, A.; and Ahmed, S. 2018. DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *Ieee Access*, 7: 1991–2005.

Schermerhorn, P.; Kramer, J.; Brick, T.; Anderson, D.; Dingler, A.; and Scheutz, M. 2007. DIARC: A testbed for natural human-robot interaction. In *Mobile Robot Competition and Exhibition - Papers from the 2006 AAAI Workshop, Technical Report*, AAAI Workshop - Technical Report, 45–52. ISBN 9781577353201.

Smaldone, R. A.; Thompson, C. M.; Evans, M.; and Voit, W. 2017. Teaching science through video games. *Nature chemistry*, 9(2): 97–102.

Xu, J.; Wu, H.; Wang, J.; and Long, M. 2021. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. *arXiv preprint arXiv:2110.02642*.