Scaling Graphically Structured Diffusion Models

Christian Weilbach [1] **William Harvey** [1] **Hamed Shirzad** [1] **Frank Wood** [1]

Abstract

Applications of the recently introduced graphically structured diffusion model (GSDM) family show that sparsifying the transformer attention mechanism within a diffusion model and meta-training on a variety of conditioning tasks can yield an efficiently learnable diffusion model artifact that is capable of flexible amortized inference in probabilistic graphical models. While extremely promising in terms of applicability and utility, implementations of GSDMs prior to this work were not scalable beyond toy graphical model sizes. We overcome this limitation by describing and and solving two scaling issues related to GSDMs; one engineering and one methodological. We additionally propose a new benchmark problem of weight inference for a convolutional neural network applied to 14×14 MNIST.

1. Introduction

Diffusion models have proven wildly successful as generative models of sensory data like images (Ho et al., 2020; Rombach et al., 2022), audio (Kong et al., 2020), and video (Ho et al., 2022; Harvey et al., 2022). Conditional variants of these (Tashiro et al., 2021), such as image completion artifacts, can be understood as performing amortized inference in these spaces. While these domains are complex and high-dimensional, it seems that the structure they possess is amenable to learning through gradient descent in architectures with simple inductive biases like convolutions.

Now consider performing a factorization of a 100×100 matrix with rank 10. This problem is trivial with a hand-designed algorithm but almost impossible for a naively-applied neural network (Weilbach et al., 2022). This problem space exhibits much more explicit structure than image completion, in that we know that the product of the factors should be exactly equal to the input matrix, but the dependencies are very non-local and so this structure is not amenable to learning with standard architectures. In addition, permutation invariances in valid factorizations mean that there are multiple solutions and so it is a difficult problem for conventional methods.

The recently introduced graphically structured diffusion model (GSDM) skirts around these issues by (a) using a stochastic diffusion model to allow for the possibility of multiple solutions, and (b) explicitly integrating known problem structure via a graphical model specification to better model complex dependencies, as we describe in Section 2. By doing so, it exhibits gracefully scaling in performance with problem size and can be trained on 30×30 rank 8 matrices towards low error. Training does not scale, though, to significantly larger graphical models than this. For larger models the runtime requirements are high, meaning that training is slow and quickly turns infeasible due to accelerator memory limitations. As a first, engineering, contribution, we modify the implementation of GSDM's attention mechanism in order to improve its scaling with respect to both of these. We describe this in Section 3.

As a second, methodological, contribution, we introduce a technique which we call plate sampling (PS). This technique stochastically selects only a few graphical model nodes at training time, so that we may save memory and computation by only considering a small fraction of the total number of graphical model nodes. Intuitively, this may be helpful if there is a lot of redundant computation in a graphical model. As an example of this, our graphical model in Figure 1 contains a plate in which the same computation is repeated N times. We describe a heuristic method for selecting such "informative" subgraphs for any given graphical model in Section 4.

Our final contribution is a new benchmark for testing amortized inference in highly structured domains. Specifically, the challenge is to infer a Bayesian posterior over convolutional neural network weights given an observed set of inputs and outputs. We describe this in Section 5 and, to enable experimentation with varying problem sizes, we release our code[1] which maps from a neural network specification

[*]Equal contribution [1]Department of Computer Science, University of British Columbia, Vancouver, Canada. Correspondence to: Christian Weilbach <weilbach@cs.ubc.ca>.

Accepted to ICML workshop on Structured Probabilistic Inference & Generative Modeling. 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]https://github.com/plai-group/gsdm

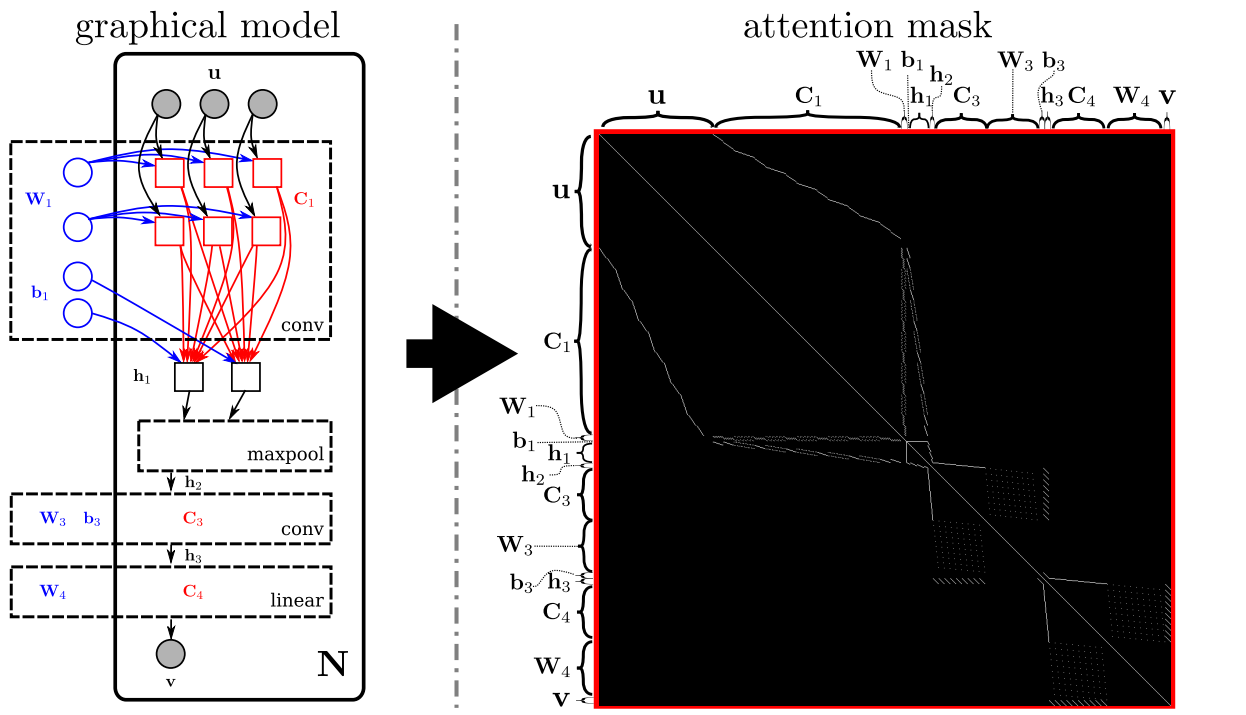

Figure 1. **Left:** A simplified version of the graphical model we propose for a new amortized inference benchmark, with latent variables $\mathbf{x} = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_3, \mathbf{b}_3, \mathbf{W}_4\} \cup \{\mathbf{C}_1^i, \mathbf{h}_1^i, \mathbf{h}_2^i, \mathbf{C}_3^i, \mathbf{h}_3^i, \mathbf{C}_4^i\}_{i=1}^N$ and observed variables $\mathbf{y} = \{\mathbf{u}^i, \mathbf{v}^i\}_{i=1}^N$. It corresponds to the compute graph of a convolutional neural network, with $\{\boldsymbol{W}_1, \mathbf{b}_1, \boldsymbol{W}_3, \mathbf{b}_3, \boldsymbol{W}_4\}$ being the model weights. The plate represents N separate network passes, each of which takes an input $\mathbf{u}$ and transforms it into an output $\mathbf{v}$ via the hidden states $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$. The intermediate variables denoted $\boldsymbol{C}_1$, $\boldsymbol{C}_3$, and $\boldsymbol{C}_4$ are intermediate quantities computed during tensor multiplication; including them makes the graphical model graph more sparse. For visual clarity the graphical model shown is significantly simplified: we show only 3 input pixels; we show all hidden states as having only one channel; we show the first convolution as having a 1×1 kernel; and we omit the internals of all layers but the first. **Right:** The resulting structured attention mask for GSDM with dimensions annotated. The mask is binary, where black entries are ignored and only white entries of the mask can be atteneded to. We set the number of network passes (i.e. number of input images) $N = 1$ and the first and second layer channel sizes to 1 to simplify the mask for this exposition.

to the corresponding compute graph and all graphical model edges.

2. Graphically Structured Diffusion Models

Overview GSDM (Weilbach et al., 2022) is a technique for constructing and training a conditional diffusion model for approximate inference in a given probabilistic graphical model. It translates the graphical model's connectivity structure to the adjacency matrix of a graphical neural network (GNN) which parameterizes the diffusion model. It also uses side-information extracted from the graphical model to share certain embeddings within the neural network and further enhance performance. We provide some background on conditional diffusion models before discussing the key features of GSDM, structured attention and embedding sharing. In the following we denote GSDM's latent variables $\mathbf{x}$ and observed variables $\mathbf{y}$, but note that these may each contain a multitude of different tensors or types of data; e.g., in Figure 1, $\mathbf{x}$ refers to all neural network weights and hidden states. The division of nodes into $\mathbf{x}$ and $\mathbf{y}$ can also vary between training examples if there are variables which we wish to sometimes condition on and sometimes infer.

Conditional diffusion models We provide a brief description of the conditional diffusion framework underlying GSDM which should be sufficient to understand this paper's constributions. See Weilbach et al. (2022) or Tashiro et al. (2021) for a more complete description. Let $\mathbf{x}_0$ be the latent variables we wish to model, and $\mathbf{y}$ denote observed variables we wish to condition on. GSDM is a conditional diffusion model which fits to the posterior under the data distribution, $q(\mathbf{x}_0, \mathbf{y})$. Underlying it is a diffusion process which progressively adds noise to $\mathbf{x}$, yielding a chain of increasingly noisy copies of the latent variables, which we call $\mathbf{x}_1$, $\mathbf{x}_2$, and so on until $\mathbf{x}_T$. These have the joint distribution $q(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})$ where $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a diagonal Gaussian distribution and, consequently, other marginals including $q(\mathbf{x}_t|\mathbf{x}_0)$

are Gaussian for any $t \in \{1, \ldots, T\}$. Under the conditional diffusion framework, we invert this process as $p_\theta(\mathbf{x}_{0:T}|\mathbf{y}) = p(\mathbf{x}_T) \prod_{i=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$. where $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$ is typically approximated as a Gaussian with non-learned diagonal covariance. Ho et al. (2020) obtain its mean as an affine function of the conditional expectation $\mathbb{E}[\mathbf{x}_0|\mathbf{x}_t, \mathbf{y}]$ and so reduce the problem to one of fitting this conditional expectation. We follow them in using the simple mean-squared error loss

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_t, \mathbf{y})} \left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t, \mathbf{y}, t) - \mathbf{x}_0\|_2^2 \right]. \quad (1)$$

Structured attention In this section we describe how GSDM maps from a probabilistic graphical model to a GNN adjacency matrix. If the graphical model is directed, the procedure to do so is simple: we make a symmetrized graph such that, if node i connects to node j in the directed graphical model, node i connects to node j and node j connects to node i in the symmetrized graph. The adjacency matrix of this symmetrized graph is then applied to the GNN. We show an example of this process in Figure 1, where the graphical model on the left is transformed into the adjacency matrix on the right. See Weilbach et al. (2022) for a proof that this symmetrization is necessary for the GNN to be able to faithfully model the data distribution and for the procedure for dealing with undirected graphical models like factor graphs. The GNN adjacency matrix, which we denote $\boldsymbol{M}$, is then applied to mask the transformer attention mechanism within GSDM's architecture such that node i can attend to node j iff. $\boldsymbol{M}_{i,j}$ is 1.

Compared to a dense transformer attention mechanism, the resulting extremely sparse attention mechanism reduces the computational cost and ensures that it scales with the number of graphical model edges e as $\mathcal{O}(e)$ instead of with the number of nodes n as $\mathcal{O}(n^2)$. The structured attention also provides an inductive bias which Weilbach et al. (2022) show is necessary for learning at all on some graphical models, and improves the scaling law between model performance and problem size in others.

Embedding sharing Known permutation invariances of the graphical model can be enforced in the distribution learned by GSDM through structured parameter sharing with a technique that Weilbach et al. (2022) call "exchangeable embeddings", or EE. Weilbach et al. (2022) also demonstrate "array embeddings", or AE, which do not enforce permutation invariances but instead use information about which graphical model nodes are within the same multi-dimensional arrays in a generative model. For example, in the CNN model in Figure 1, this approach would provide correlated embeddings to all nodes that make up the input $\mathbf{u}$, to all nodes that make up the weight matrix $\boldsymbol{W}_1$, and so on. This information is easier to automatically extract from a model than permutation invariances and Weilbach et al. (2022) show that it works similarly well or sometimes better than EE. Finally, Weilbach et al. (2022) also compared against independent embeddings, or IE, in our experiments. In this setting, embeddings are learned per-node with no weight sharing. This is a baseline which we will use in our experiments to verify that the benefits of embedding sharing are maintained as we scale to larger problem sizes.

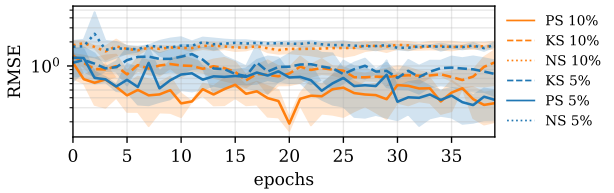

Figure 2. Different marginalization strategies on matrix factorization for subgraphs of sizes up to 5% and 10%.

3. Efficient Implementation

Given a graphical model with n nodes, the GSDM attention mechanism implemented by Weilbach et al. (2022) had computational cost $\mathcal{O}(nm)$, where m is the maximum degree of any node. With this mechanism, it is possible to train on matrix factorization problem instances with size up to roughly 50 000 nodes (enabling rank-10 factorization of 65×65 matrices) on an A5000 GPU. We replace it with a functionally equivalent attention mechanism from the GNN literature (Kreuzer et al., 2021) that instead scales as $\mathcal{O}(e)$, where e is the number of graph edges. The number of edges is a lower bound on nm, being equal only if all nodes have the same degree. This implementation allows us to scale to matrix factorization problems with as many as $190,000$ nodes, enabling rank-10 factorization of 130×130 matrices, over four times as many as with the original implementation.

4. Plate Sampling

A fundamental issue with scaling further is that, on every training iteration, we must operate on every graphical model node. For graphical models with many symmetries and millions of nodes, this may involve repeating a lot of near-identical computations, increasing computational cost without meaningfully improving the gradient estimate. We therefore propose to train on each training example by sampling a "subgraph" which includes the nodes to keep and include in $\mathbf{x}$ or $\mathbf{y}$, and then discarding the rest of the nodes. Using a distribution over subgraphs which discards 95% of nodes, we can scale to matrix factorization problems with 2 800 000 nodes, enabling rank-10 factorization of 500×500

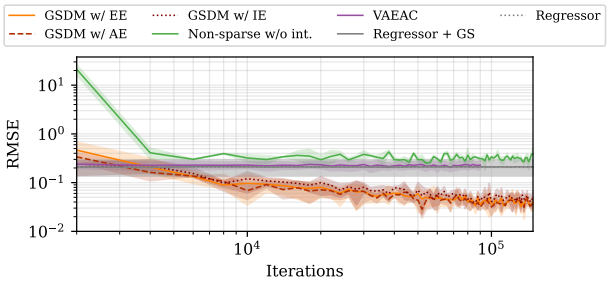

Figure 3. Reconstruction error on the CNN benchmark problem.

matrices, over fifty times as many as with the original implementation.

Multiple approaches to sampling subgraphs for GNNs already exist, which we briefly describe before introducing our novel approach to incorporating known graphical model structure into the sampling. Node sampling (NS) Zeng et al. (2019) independently samples a set of nodes in each step and trains the GNN on the subgraph spanned by these nodes. While being simple this approach has the downside that it captures few edges of the graph and can sample a lot of isolated nodes in each step. Following GraphSAGE (Hamilton et al., 2017), k-hop subgraph sampling (KS) alleviates this problem by uniformly sampling one or multiple nodes and then iteratively expanding the neighborhood until a threshold is reached. See Liu et al. (2021) for a comprehensive survey of other techniques. Our proposed method, plate sampling (PS), is similar to KS but instead of incrementally expanding the subgraph we leverage our additional knowledge of the graphical model to ensure that we sample all nodes along one plate dimension first and then subsample the remaining dimensions to be below the desired size. We call this property of PS the plate maximization invariant (PMI). Since PS has the additional PMI constraint which can be impossible to match exactly we accept graphs in the range down to half the maximum allowed size. PS addresses the problem that just sampling dense subgraphs without considering plates can create a lot of training examples in which some plate dimensions are rarely or never maximized. In this case the corresponding nodes cannot attend to the same number of edges connecting to a plate as will be observed during inference, turning them out of distribution. Take the matrix factorization example of Weilbach et al. (2022) where the goal is to find a rank k factorization of a matrix $\boldsymbol{E}$ into its factors $\boldsymbol{A}$ and $\boldsymbol{R}$ with intermediate variables $\boldsymbol{C}_{ijk} = \boldsymbol{A}_{ik}\boldsymbol{R}_{jk}$ and $\boldsymbol{E}_{ij} = \sum_k \boldsymbol{C}_{ijk}$. The rank k is typically much smaller than the ranges for i and j and therefore its plate gets sampled more often by KS while PS ensures that all plates are fully sampled with the same probability. The problem gets more pronounced in bigger models such as the proposed neural network benchmark in Section 5 where most nodes are close to the input of the neural network and KS hence would rarely reach all network output dimensions for the plate over data points.

In Figure 2 we compare PS to NS and KS on the matrix factorization problem for $i = 100, j = 100, k = 10$. Due to the $i \times j \times k$ intermediate variable nodes necessary for this problem, the full graphical model has more than $100,000$ nodes. We sample 1000 subgraphs for each strategy before training, sample one marginal in each training step and fill it with fresh values. We explore maximum sizes of 5 and 10 % of nodes. We evaluate three training runs for each strategy on a full sized problem after each epoch and plot mean RMSE with a confidence band over min and max values. NS performs poorly and saturates early in training. PS consistently outperforms KS and keeps improving during training.

5. Bayesian neural network benchmark

We propose amortized Bayesian neural network weight inference as a benchmark problem. We denote the function that the neural network parameterizes f_θ and its weights θ. Given the architecture in Figure 1, we have $\theta = [\boldsymbol{W}_1, \mathbf{b}_1, \boldsymbol{W}_3, \mathbf{b}_3, \boldsymbol{W}_4]$. As a prior $p(\theta)$ over the weights we simply use `pytorch`'s default weight initialization distribution, which is independent between all dimensions of θ. For a prior over $\mathbf{u}$, which we denote $p(u)$, we use the empirical distribution of MNIST training images at 14×14 resolution. All other graphical model nodes can be deterministically computed as a function of θ and $\mathbf{u}$ and so we can write the distribution of each given its parents as a Dirac in the graphical model. For the weight inference problem, we observe the network inputs and outputs for network passes $1, \dots, N$, which together we call $\{\mathbf{u}_i, \mathbf{v}_i\}_{i=1}^N$, and wish to infer the posterior over its weights, $p(\theta|\{\mathbf{u}_i, \mathbf{v}_i\}_{i=1}^N)$. We do not evaluate the fidelity of the inferred $\mathbf{h}$ and intermediate variables $\boldsymbol{C}$ but include them in Figure 1 because modeling the compute graph in this fine-grained manner can be extremely beneficial for inference (Weilbach et al., 2022). To summarise, given N network passes, we have $\mathbf{x} = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_3, \mathbf{b}_3, \mathbf{W}_4\} \cup \{\mathbf{C}_1^i, \mathbf{h}_1^i, \mathbf{h}_2^i, \mathbf{C}_3^i, \mathbf{h}_3^i, \mathbf{C}_4^i\}_{i=1}^N$ and $\mathbf{y} = \{\mathbf{u}^i, \mathbf{v}^i\}_{i=1}^N$. We use the architecture described in Table 1 and $N = 2$ network passes, giving rise to a total of 4814 graphical model nodes, 4794 of which are latent and 20 of which are observed. The benchmark can be incrementally scaled up both by increasing the neural network size and the number of network passes, rendering it suitable to explore scaling properties of different inference methods.

We evaluate methods on sets of observations $\{\mathbf{u}_i, \mathbf{v}_i\}_{i=1}^2$ sampled from the data distribution (via sampling 2 MNIST images $\mathbf{u}_1$, $\mathbf{u}_2$, sampling network weights θ from the prior,

Table 1. Network architecture for the convolutional network we model. No padding is applied. The 10 dimensional output of the network could be used to parametrize a softmax distribution.

layer	in channels	out channels	kernel size	stride	activation	input shape	output shape
conv	1	4	3	2	-	$1 \times 14 \times 14$	$4 \times 6 \times 6$
maxpool	4	4	2	2	-	$4 \times 6 \times 6$	$4 \times 3 \times 3$
conv	4	10	3	1	ReLU	$4 \times 3 \times 3$	$10 \times 1 \times 1$
linear	10	1	1	1	-	$10 \times 1 \times 1$	$10 \times 1 \times 1$

Table 2. Final reconstruction RMSE achieved by each method on the neural network weight inference benchmark.

GSDM w/ EE	GSDM w/ AE	GSDM w/ IE	Non-sparse.	VAEAC	Regr. + GS	Regressor
0.039 $\pm$ 0.008	**0.039** $\pm$ 0.012	0.052 $\pm$ 0.015	0.391 $\pm$ 0.116	0.228 $\pm$ 0.044	0.210 $\pm$ 0.073	0.210 $\pm$ 0.065

and then setting $\mathbf{v}_1 := f_\theta(\mathbf{u}_1)$ and $\mathbf{v}_2 := f_\theta(\mathbf{u}_2)$). Then we use each method to estimate θ given each pair of observations. Calling the estimate $\hat{\theta}$, we estimate the error for the method as $\sum_{i=1}^{2} ||f_\theta(\mathbf{u}_i) - f_{\hat{\theta}}(\mathbf{u}_i)||_2^2$. We take the square root of this and average over all sets of observations to obtain the final RMSE. We evaluate a range of approaches, showing their training progress in Figure 3 and final RMSEs in Table 2. Specifically, we evaluate: **GSDM** with each of EE, AE, and IE as described in Section 2; **Non-sparse** is an ablation of GSDM with the structured attention replaced by a dense transformer mechanism; **VAEAC** (Ivanov et al., 2019) is a VAE-based approach; **Regressor + GS** is an ablation which uses the same architecture as GSDM w/ EE but is trained with a mean-squared error loss to deterministically predict $\mathbf{x}$ given $\mathbf{y}$ instead of parameterizing a diffusion model.; **Regressor** also makes a deterministic mean-squared error prediction, but uses the same architecture as "Non-sparse". Both deterministic methods learn to predict constants almost independent of $\mathbf{y}$ because they cannot account for the problem's permutation invariances. The "non-sparse" ablation and VAEAC both fail to learn the complex relationships between variables and so do even worse than these deterministic baselines. The EE and AE variations of GSDM perform best, both achieving a final RMSE of 0.039.

6. Conclusion

We scale GSDM by improving its implementation, providing a benchmark and suggesting a better plate base subgraph method. Future work will explore plate sampling on the proposed benchmark and large scale scientific simulators.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada CIFAR AI Chairs Program, and the Intel Parallel Computing Centers program. Additional support was provided by UBC's Composites Research Network (CRN), and Data Science Institute (DSI). This research was enabled in part by technical support and computational resources provided by WestGrid (www.westgrid.ca), Compute Canada (www.computecanada.ca), and Advanced Research Computing at the University of British Columbia (arc.ubc.ca).

References

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Harvey, W., Naderiparizi, S., Masrani, V., Weilbach, C., and Wood, F. Flexible diffusion modeling of long videos. *Advances in Neural Information Processing Systems*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video Diffusion Models. *arXiv:2204.03458 [cs]*, April 2022.

Ivanov, O., Figurnov, M., and Vetrov, D. Variational Autoencoder with Arbitrary Conditioning. *arXiv:1806.02382 [cs, stat]*, June 2019.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

Liu, X., Yan, M., Deng, L., Li, G., Ye, X., and Fan, D. Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, 9(2):205–234, 2021.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Tashiro, Y., Song, J., Song, Y., and Ermon, S. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*, 2021.

Weilbach, C., Harvey, W., and Wood, F. Graphically structured diffusion models. *arXiv preprint arXiv:2210.11633*, 2022.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.