Automatic Quantum Optics experimental design with sequential graph generative models

Written by Anon [1*]
Anon
[1]Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303
publications20@aaai.org

Abstract

With the rise of quantum computing and the continuous development of quantum information technologies, there is a large demand for producing evermore exotic entangled quantum mechanical states. However, it is an extremely difficult task to design quantum optical systems that generate these states. Therefore, we turn to recent work on sequential generative models of graphs to help us generate and search through the space of possible experimental quantum optical setups. Our model constructs experiments by sequentially adding devices to the setup. We demonstrated that our model can build valid possible experimental setups that are unique and novel from the training data.

Introduction

Over the second half of the 20th century, we have witnessed major breakthroughs in quantum mechanics from experimental verification of quantum entanglement to the development of groundbreaking quantum information technologies such as quantum computing and quantum cryptography. Among these, quantum entanglement is an important phenomena for development of these technologies. Thus, building experimental setups to produce target entangled states is an essential research task.

With recent advances in automated design procedures, large datasets of experimental setups can be procured, the MELVIN algorithm is the first of such (Krenn et al. 2016). MELVIN searches through the large combinatorial space of possible experiments by a simple Genetic Algorithm – checking if random configurations of optical elements produce interesting quantum mechanical states. For states with just a few photons, it is generally infeasible for humans to attempt to solve the inverse problem of determining the setup required to produce a specific state.

On the other hand, over the past few years, there has been considerable interest and advancement in building generative models of graphs with many important applications such as automatic and goal directed molecular design. Multilayered models can be trained sequentially to generate valid molecular graphs that match interesting data distributions of, for example, drug like molecules.

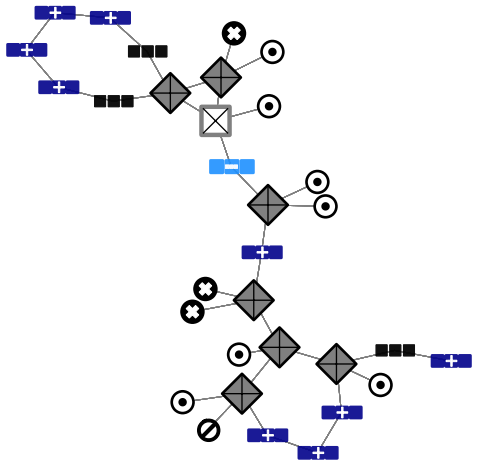


Figure 1: A setup for a quantum optics experiment represented as a graph. Specific nodes correspond to specifc optical devices. More details in the Dataset section

In this work, we apply these recent advances to the task of generating experimental setups for quantum optics research and define the following contributions:

- We introduce a new bench mark dataset for the generation of semantically constrained graphs consisting of 1 million QO experimental setups.
- We build a sequential generative model of graphs specifically for the generation of QO experimental setups by specifying a simple generation procedure that avoids computing intractable likelihoods over complicated generation traces.
- We test our model by assessing validity, uniqueness and novelty on random samples from the trained model.

*Primarily Mike Hamilton of the Live Oak Press, LLC, with help from the AAAI Publications Committee

Table 1: Possible Optical Devices

Device	Description
	spontaneous parametric down conversion
	**Leach interferometer**
	**Beam Splitter**
	**Hologram that decrease mode #**
	**Hologram that increase mode #**
	**Dove prism**
	Reflection
	**Detector**
	**Detector at trigger**
	**Path without Detector**
	empty path

The Dataset

We train on a dataset of around 1 million experimental setups using the graph representation of each setup. Each graph is represented by its device and path features.

Node representations A setup S consists of devices which we treat as nodes in the graph. They come from a set of possible devices to include in the setup, each with feature vector $\boldsymbol{d} \in \mathbb{R}^D$. There are $D = 11$ possible devices which are displayed in table 1. We consider setups of at most $N = 61$ devices including different kinds of optical devices like holographhms and detectors.

Edge representation: The experiments involve using three photons with orbital angular momentum (OAM) cite. They traverse a sequence of optical devices including beam splitters, and holographic plates and obtain a final output state. They can travel along both directions of the setup and the graphs are undirected. There are two types of paths giving path feature $\boldsymbol{p}_{dd'} \in \mathbb{R}^2$ for all possible $N \times N$ paths between devices on the graph, meaning that $\boldsymbol{p}_{dd} = [0, 1]$ if device d is connected to device d' with path type 2.

Constraints : Not all possible arrangements of devices are valid experimental setups that would produce an entangled state. For example a beam splitter must have 4 paths connecting to it while a detector can only detect one photon and must have one path connected to it. Similar constraints exist for all other devices.

Related Work Model Design

In recent years there have been few advancements in building generative models of graphs. Sequential generative models like graph rnn (You et al. 2018b) autoregressively generate a graph by decomposing the generation process into a sequence of node and edge constructions, conditioned on the generated graph structure. These are powerful models since they can account for correlations between edges which are crucial in real life applications.

In particular, the following three Sequential Generative Models of Graphs, are specified for constrained graph generation, supporting multiple edge types and condition on past generation by using learned representations from graph neural networks. We construct our model using components of these three with consideration of our domain.

1) DGM Learning deep generative models of graphs (Li et al. 2018) Is the first and most general approach for learning generative models over graphs : it uses a simple message passing neural network to parameterize a distribution over nodes and edges for any arbitrary graph. They sequentially generate each graph using a fixed ordering or uniform random ordering during training. The generation process consists of three main modules `1) add node 2) add edge 3) nodes` which 1) decide to add another node or terminate 2) decide to add an edge (or not) to the newly created node and 3) pick another node in the graph to connect to with a specific edge type. They initialize every added nodes representation using a separate MLP module with both graph level and node level embeddings.

2) CGVAE Constrained Graph Variational Auto-Encoders for Molecule Design (Liu et al. 2018) propose a graph structured variational autoencoder by building gated graph neural networks (Li et al. 2015) into the encoder and decoder. They assume a sequential ordering of graph generation steps and constrain their model so it can design molecules that are optimal in desired properties. Their model first initializes a set of nodes using a sampled latent variable and cycles between a edge selection plus labeling step and Node update/ Termination stage until their model flags to globally stop.

3) GCPN Graph convolution policy network (You et al. 2018a) Most related to our work is GCPN, which uses a graph convolutional network (Kipf and Welling 2016) for goal-directed graph generation through reinforcement learning. The model is trained to optimize domain-specific rewards and adversarial loss through proximal policy gradient optimization (Schulman et al. 2017), and acts in an environment that incorporates domain-specific rules. The generation process starts with a single carbon atom and at every step uses four modules to 1) select a node on the current graph 2) select a second node from the union of current graph nodes and set of all possible scaffold graphs 3) sample a bond type between the two selected nodes and 4) choose whether to terminate graph construction or continue GCPN does not initialize node representations.

Graph Generative Model

We use the base graph construction framework similar to GAR and GCPN – we want to construct setups by sampling a device and path one at a time. We avoid computing complicated likelihoods based on complex generation histories by specifying an adversarial loss. Our generative process is path oriented: first selecting a device on the current generated setup to build a path from, then choosing a double or single photon path and deciding whether or not connect the path back to the setup or add a new device to the setup. We initialize setups with a single device either 1) a random device 2) Beam splitter or 3) Detector

Setup Construction Modules

The `select device` module

This module selects a device from current experimental setup S including an off setup *terminate device* $\varnothing$ which tell us to globally stop adding devices. This is done by MLP module $f_{\texttt{select}}$ which takes in setup representation $\mathbf{h}_S$ and outputs the probability of selecting a specific device.

$$f_{\texttt{select}}(S) = \sigma(m_{\texttt{MLP}}(\boldsymbol{h}_S))$$

The `path` module

This module simply chooses which path type to build onto the experimental setup from the selected device. Once again we use an mlp $f_{\texttt{path}}$ that takes in the representations of the device selected and the full setup S to determine the probability of building a single or double path.

$$f_{\texttt{path}}(S, s) = \sigma(m_{\texttt{MLP}}(\boldsymbol{h}_S, \boldsymbol{h}_s))$$

The `close` module

This module decides whether to connect the new path to a device on the current setup or add a new device to the path. Once again we use an mlp $f_{\texttt{close}}$ that takes in the representations of the device selected and the full setup S to determine the probability of closing a cyle or adding a new device

$$f_{\texttt{close}}(S, s, p) = \sigma(m_{\texttt{MLP}}(\boldsymbol{h}_S, \boldsymbol{h}_s, \boldsymbol{p}))$$

The `connect or add device` modules

Depending on the choice of the close module the connect module either 1) samples a new device from the set of all possible devices to connect to the selected device or 2) chooses another device on the current setup to connect to the selected device s ie close the loop This is done by MLP modules $f_{\texttt{connect}}$ or $f_{\texttt{add}}$ which takes in setup representation $\mathbf{h}_S$ and selected device representation $\mathbf{h}_s$ then outputs the probability of choosing which device to connect.

$$f_{\texttt{connect}}(S, s) = \sigma(m_{\texttt{MLP}}(\boldsymbol{h}_S, \boldsymbol{h}_s, \boldsymbol{p}))$$

Algorithm 1 A single Generation Step of a QO Setup

Input current setup S
$\{\boldsymbol{h}_d^\top\}_{d\in S} = \texttt{MessagePassing}(\boldsymbol{h}_d^0, \boldsymbol{p}_{dd'})$ ▷ device level propagation
$\boldsymbol{h}_S^\top = \texttt{Readout}(\{\boldsymbol{h}_d^\top, \boldsymbol{x}_d\})$ ▷ setup level aggregation
$s \sim \texttt{Cat}(f_{\texttt{select}}(\boldsymbol{h}_S))$ ▷ select
$\boldsymbol{p} \sim \texttt{Ber}(f_{\texttt{path}}(\boldsymbol{h}_S, \boldsymbol{h}_s))$ ▷ build path
$\texttt{Close} \sim \texttt{Ber}(f_{\texttt{path}}(\boldsymbol{h}_S, \boldsymbol{h}_s))$
if `Close` **then**
    $d \sim \texttt{Cat}(f_{\texttt{connect}}(\boldsymbol{h}_S, \boldsymbol{h}_s, \boldsymbol{p}))$ ▷ close cycle
    Return $S \to S \cup (\boldsymbol{p}_{sd})$ ▷ return new setup
else
    $d \sim \texttt{Cat}(f_{\texttt{add}}(\boldsymbol{h}_S, \boldsymbol{h}_s, \boldsymbol{p}))$ ▷ add new device
    Return $S \to S \cup (d, \boldsymbol{p}_{sd})$
end if

Propagation and Aggregation model

To learn a representation of the constructed graphs at each construction step we use a message passing neural network (Gilmer et al. 2017) that uses graph attention. (Veličković et al. 2017)

$$\boldsymbol{m}_d = \underset{s\in\mathcal{N}_d}{\texttt{Attn}}([\boldsymbol{h}_d^t, \boldsymbol{h}_s, \boldsymbol{p}_{sd}]), \quad U_t = \sigma([\boldsymbol{h}_d^t, \boldsymbol{m}_d^t,])$$

as well as the message function from the model in interaction networks (Battaglia et al. 2016), which is a simple concatenation of node and edge features. The node update function concatenates incoming messages with the current node state and feeds it through a dense layer. And we get graph level representations

$$\boldsymbol{h}_S = \underset{d\in S}{\texttt{Set2Set}}(\{\boldsymbol{h}_s^T, \boldsymbol{x}_s\})$$

After propagation through message passing layers, we use the set2set model (Vinyals, Bengio, and Kudlur 2015) as the readout function to combine the node hidden features into a fix-sized hidden vector.

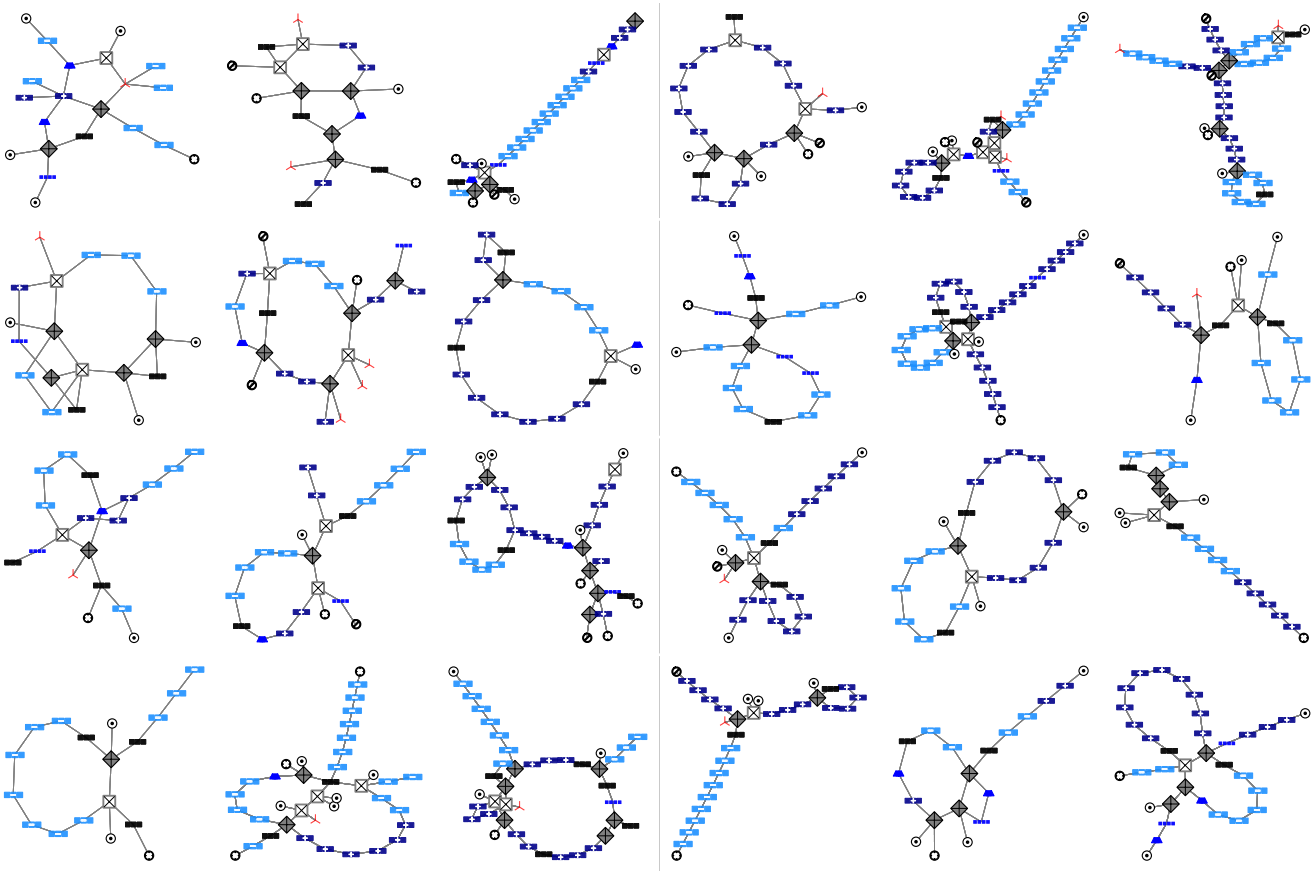
Likelihood free training

In order for our model to match the data distribution $p_{\texttt{data}}(S)$ of QO experimental setups we make use of a GAN framework with adversarial loss $\ell(\boldsymbol{\theta}, \boldsymbol{\phi})$

$$\begin{aligned}\ell(\boldsymbol{\theta}, \boldsymbol{\phi}) = &\mathbb{E}_{S\sim p_{\texttt{data}}(S)}[\log D_{\boldsymbol{\phi}}(S)] \\ &+ \mathbb{E}_{S\sim p_{\texttt{G}}(S)}[\log(1 - D_{\boldsymbol{\phi}}(S))]\end{aligned}$$

where D is the discriminator. We can not differentiate through the generative process which produces a discrete graph, instead we employ the reinforce gradient estimator.

Initialization We initialize each devices learned representation using the concatenation of prior noise and device features.



Samples from training set **Random Samples from the Model**

Figure 2: A comparison of samples from the data distribution and samples from the generative model, more samples in the appendix

Experiments

We conduct some simple basic preliminary experiments with our model

- We sample 10K experimental setups from our model and compute validity, novelty, uniqueness as in (Simonovsky and Komodakis 2018).
- we plot random samples and compare to samples from the training set.

Figure two shows samples from the model and data: demonstrating we reasonably match the data distribution.

For Validity, uniqueness and Novelty we score $100, 96, 100$ percentages respectively.

References

Battaglia, P.; Pascanu, R.; Lai, M.; Rezende, D. J.; et al. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, 4502–4510.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks.

Krenn, M.; Malik, M.; Fickler, R.; Lapkiewicz, R.; and Zeilinger, A. 2016. Automated search for new quantum experiments. *Physical review letters* 116(9):090405.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; and Battaglia, P. 2018. Learning deep generative models of graphs.

Liu, Q.; Allamanis, M.; Brockschmidt, M.; and Gaunt, A. 2018. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems*, 7795–7804.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms.

Simonovsky, M., and Komodakis, N. 2018. Graphvae: To-

wards generation of small graphs using variational autoencoders.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

You, J.; Liu, B.; Ying, R.; Pande, V.; and Leskovec, J. 2018a. Graph convolutional policy network for goal-directed molecular graph generation.

You, J.; Ying, R.; Ren, X.; Hamilton, W. L.; and Leskovec, J. 2018b. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*.