# Junction Tree Variational Autoencoder for Molecular Graph Generation

* Abstract: - the researchers’ primary contribution is the direct realization of molecular graphs, a task previously approached by generating linear SMILES stings instead of graphs.
* The researcher’s *junction tree variational autoencoder* generates molecular graphs in two phases, in first it generates a tree-structured scaffold over chemical substructures and then combines them into a molecule with a graph message passing network.
* Introduction: - the key challenge of drug discovery is to find target molecules with desired chemical properties.
* From a computational perspective, the researchers decomposed the challenge into two complementary subtasks: learning to represent molecules in a continuous manner that facilitates the prediction and optimization of their properties; and learning to map an optimized continuous representation back into a molecular graph with improved properties.
* Prior work on drug design formulated that graph generation task as a string generation problem in an attempt to side-step direct generation of graphs.
* These models start by generating SMILES, and these SMILES strings can be translated into graphs via deterministic mappings.
* However, this design has two critical limitations, first, the SMILES representations are not designed to capture molecular similarity. Second, essential chemical properties such as molecule validity are easier to express on graphs rather than in linear SMILES representations.
* The researchers hypothesize that operating directly on graphs improves the generative modeling of valid chemical structures.
* The researchers’ primary contribution is a new generative model of molecular graphs.
* The overall generative approach, cast as a *junction tree variational autoencoder*, first generates a tree-structured object whose role is to represent the scaffold of subgraph components and their coarse relative arrangements.
* In the second phase, the subgraphs are assembled into a molecular graph.
* The researchers’ model demonstrates that the model produces 100% valid molecules when sampled from a prior distribution, outperforming the top-performing baseline by a significant margin.
* They also show that their model excels in discovering molecules with desired properties, yielding a 30% relative gain over the baselines.
* Junction Tree Variational Autoencoder: - the researchers’ model extends the variational autoencoder to molecular graphs by introducing a suitable encoder and a matching decoder.
* The key advantage of this view is that the decoder can realize a valid component and how they interact, rather than trying to build the molecule atom by atom through chemically invalid intermediaries.
* The vocabulary of components is chosen to be large enough so that a given molecule can be covered by overlapping components or *clusters* of atoms.
* The clusters serve the role analogous to cliques in graphical models, as they are expressive enough that a molecule can be covered by overlapping clusters that serve as cliques in a triangulation of the molecular graph.
* The researchers form a junction tree of such clusters and use it as the tree representation of the molecule.



* The original molecular graph and its associated junction tree offer two complementary representations of a molecule.
* The researchers encode the molecule into a two-part latent representation z = [] where encodes the tree without fully capturing how exactly the clusters are mutually connected.
* zG encodes the graph to capture the final-grained connectivity. Both parts are created by tree and graph encoders *q*() and *q*().
* The latent representation is then decoded back into a molecular graph in two stages. The first reproduces the junction tree using a tree decoder *p*() based on the information in .
* Second, they predict the fine grain connectivity between the clusters in the junction tree using a graph decoder *p*(G|) to realize the full molecular graph.
* Notation: - A molecular graph is defined as *G* = (*V, E*) where *V* is the set of atoms (vertices) and *E* is the set of bonds (edges). Let *N*(*x*) be the neighbor of *x*. The researchers’ sigmoid function as () and ReLU function as . They use *i, j, k* for nodes in the tree and *u, v, w* for nodes in the graph.
* Junction Tree: - A tree decomposition maps a graph *G* into a *junction tree* by contracting certain vertices into a single node so that *G* becomes cycle-free.
* Given a graph *G*, a junction tree ) is a connected labeled tree whose node set is = {C1,…, Cn} and edge set is . Each node or *cluster Ci* = (*Vi, Ei*) is an induced subgraph of *G¸*satisfying the following constraints:
  + The union of all clusters equals *G*. That is, = *V* and = E.
  + Running intersection: for all clusters *Ci, Cj,* and *Ck, Vi Vj*  Vk is on the path from *Ci* to *Cj.*
* Tree Decomposition of Molecules: - the researchers' cluster vocabulary includes chemical structures such as bonds and rings.
* They first find all the simple cycles, of graph *G*, and its edges not belonging to any cycles.
* Two simple rings are merged if they have more than two overlapping atoms, as they constitute a specific structure called bridged compounds.
* Next, a cluster graph is constructed by adding edges between all intersecting clusters.
* Finally, they select one of its spanning trees as the junction tree of *G*.
* Graph Encoder: - the researchers first encode the latent representation of *G* by a graph message passing network.
* Each vertex has a feature vector indicating the atom type, valence, and other properties.
* Likewise, each edge (*u, v*) *E* has a feature vector indicating its bond type, and two hidden vectors and denoting the message from *u* to *v* and vice versa.



* Here is the message computed in *t*-th iteration, initialized with = 0. After *T* steps of iteration, the researchers aggregate those messages as the latent vector of each vertex, which captures its local graphical structure:



* The final graph representation is . The mean and log variance log of the variational posterior approximation are computed from h*G* with two separate affine layers. z*G* is sampled from a Gaussian ().
* Tree Encoder: - the researchers also encode the with a tree message passing network. Each edge (*Ci, Cj*) is associated with two message vectors and.
* They picked an arbitrary leaf node as the root and propagate the message in two phases. In the first bottom-up phase, messages are initiated from the leaf nodes and propagated iteratively towards the root.
* In the top-down phase, messages are propagated from the root to all the leaf nodes. Message is updated as:





* The message passing follows the schedule where is computed only when all its precursors {} have been computed.
* After the message passing, the researchers obtain the latent representation of each node by aggregating its inward message:



* The final tree representation is = , which encodes a rooted tree ().
* z is sampled in a similar way as in the graph encode.
* This tree encoder plays *two* roles in the researchers’ framework. First, it is used to compute , which only requires the bottom-up phase of the network. Second, after a tree is decoded from , it is used to compute a message over the entire , to provide essential contexts of every node during graph decoding.
* Tree Decoder: - the researchers decode a junction tree from its encoding with a tree-structured decoder.
* The tree decoder traverses the entire tree from the root and generates nodes in their depth-first order.
* For every visited node the decoder first makes a *topological prediction:* whether this node has children to be generated.
* If a new child node is created, then it predicts its label and recurses its process. The decoder backtracks when a node has no more children to generate.
* The information is propagated through message vectors when trees are incrementally constructed.
* Let, = {(),…,()} be the edges traversed in a depth-first traversal over = (), where *m* = 2|| as each edge is traversed in both directions.
* The model visits the node at time *t*. Let be the first *t* edges in . The message is updated through previous messages:



* Topological Prediction: - when the model visits the node , it makes a binary prediction on whether it still has children to be generated. The researchers compute this probability by combining , node feature and inward messages via a one hidden layer network followed by a sigmoid function:



* Label Prediction: - When a child node *j* is generated from its parent *i*, the researchers predict its node label with



* Here is a distribution over label vocabulary . When *j* is a root node, its parent *i* is a virtual node and = 0.
* Learning: - the tree decoder aims to maximize the likelihood . Here let and be the ground truth topological and label values, the decoder minimizes the following cross-entropy loss:



* The researchers perform *teacher forcing:* after topological and label predictions at each step, they replace them with their ground truth so that the model makes predictions given the correct history.



* Decoding & Feasibility Check: - the above algorithm shows how a tree is sampled from .
* The tree is constructed recursively guided by topological predictions. To ensure the sampled tree could be realized into a valid molecule, they define set to be cluster labels that are chemically compatible with node *i* and its current neighbors.
* When a child node *j* is generated from node *i*, the researchers sample its label from with a renormalized distribution over by masking out invalid labels.
* Graph Decoder: - the final step of the model is to reproduce a molecular graph *G* that underlies the predicted junction tree = .
* The underlying degree of freedom pertains to how neighboring clusters and are attached as subgraphs.
* Researchers' goal is to assemble the subgraphs into the correct molecular graph.
* They take to be the set of graphs whose junction tree is . Decoding graph from = () is a structured prediction:



* Here is a scoring function over candidate graphs. Each term in the scoring function depends only on how a cluster , *j* (*i*) in the tree .
* The problem of finding the highest scoring graph – the assembly task – could be cast as a graphical model inference task in a model induced by the junction tree.
* They start the sampling of the assembly of the root and its neighbors according to their scores. Then they proceed to assemble the neighbors and their associated clusters, and so on.
* It remains to be specified how each neighborhood realization is scored. Here be the subgraph resulting from a particular merging of cluster in the tree with its neighbors .
* The researchers score the as a candidate subgraph by first deriving a vector representation and then using as the subgraph score.
* Let *u, v* specify atoms in the candidate subgraph and let if and if .
* The indices are used to mark the position of the atoms in the junction tree, and to retrieve messages summarizing the subtree, under *i* along the edge (*i,j*) obtained by running the tree encoding algorithm.
* The neural messages about the atoms and bonds in the subgraph are obtained and aggregated into similarly to the encoding step, but with different parameters:



* The major difference from Eq. (1) is that the researchers augment the model with tree messages derived by running the tree encoder over the predicted tree . provides a tree-dependent positional context for bond (*u, v*).
* Learning: - the graph decoder parameters are learned to maximize the log-likelihood of predicting correct subgraphs of the ground true graph *G* at each tree node:



* Here is the set of possible candidate subgraphs at tree node *i*. the researchers feed the graph decoder with ground truth trees as input.
* Complexity: - any two clusters share at most two atoms, so we only need to merge at most two atoms or one bond.
* By pruning chemically invalid subgraphs and merging isomorphic graphs, || 4 on average when tested on a standard ZINC drug dataset. The computational complexity of JT-VAE is therefore linear in the number of clusters, scaling nicely to large graphs.
* Experiments: - Molecule reconstruction and validity – the researchers test the VAE models on the task of reconstructing input molecules from their latent representations, and decoding valid molecules when sampling from the prior distribution.
* Bayesian optimization: - the researchers test how the model can produce novel molecules with desired properties. They perform Bayesian optimization in the latent space to search for molecules with specified properties.
* Constrained molecule optimization: - it modifies the given molecules to improve specified properties while constraining the degree of deviation from the original molecule.
* It is the most realistic scenario in drug discovery, where the development of new drugs usually starts with known molecules such as existing drugs.
* Data: - the researchers use the ZINC molecule dataset for their experiment.
* Baseline: - they compare their approach with a SMILES-based baseline, (1) Chapter VAE (CVAE) which generates SMILES strings character by character; (2) Grammar VAE (GVAE) that generates SMILES following syntactic constraints given by a context-free grammar; (3) Syntax-directed VAE (SD-VAE) that incorporates both syntactic and semantic constraints of SMILES via attribute grammar.
* For molecular generation task, they compare with Graph VAE which directly generates atom labels and adjacency matrices of graphs, as well as an LSTM-based autoregressive model that generates molecular graphs atom by atom.
* Model Configuration: - the researchers set the latent space dimension as 56, i.e., the tree and graph representation and have 28 dimensions each.
* Molecule Reconstruction and Validity: - Setup – the first task is to reconstruct and sample molecules from latent space.
* The encoding and decoding process is stochastic, and the researchers estimate reconstruction accuracy by the Monte Carlo method: each molecule is encoded 10 times and each encoding is decoded 10 times.
* They report that the portion of the 100 decoded molecules that are identical to the input molecule.



* To compute the validity, they sample 1000 latent vectors from the prior distribution , and decode each of these vectors 100 times.
* They also report the percentage of decoded molecules that are chemically valid, and also the validity of their model without a validity check in the decoding phase.
* The above table shows that JT-VAE outperforms previous models in molecule reconstruction, and always produces valid molecules when sampled from the prior distribution.
* The atom-by-atom-based generation only achieves 89.2% validity as it needs to go through invalid intermediate states.
* The researchers' model bypasses this issue by utilizing valid substructures as building blocks.
* They further sampled 5000 molecules from prior and found they are *all distinct* from the training set.
* Analysis: - they follow the method of Kusner et al. (2017) to construct a grid visualization of its neighborhood.
* The researcher’s model shows that their neighborhood does not contain molecules with huge rings, which rarely occur in the dataset.
* They also highlight two groups of closely resembling molecules that have identical tree structures but vary only in how clusters are attached,
* Bayesian Optimization: - following the researchers’ chemical property is octanol-water partition coefficients (logP) penalized by the synthetic accessibility (SA) score and several long cycles.
* To perform the Bayesian optimization (BO), the researchers first train a VAE and associate each molecule with a latent vector, given by the mean of the variational encoding distribution.
* After VAE is learned, they train a sparse Gaussian process (SGP) to predict given its latent representation.
* The comparisons (1) the predictive performance of SGP trained on latent encoding learned by different VAEs, measured by log-likelihood (LL) and root mean square error (RMSE) with 10-fold cross-validation. (2) the top-3 molecules found by BO under different models.
* Results: - JT-VAE finds molecules with significantly better scores than previous methods. JT-VAE finds over 50 molecules with scores over 3.50, moreover, the SGP yields better predictive performance when trained on JT-VAE embeddings.
* Constrained Optimization: - the third task is to perform molecule optimization. Given a molecule *m*, the task is to find a different molecule *mʹ* that has the highest property value with the molecular similarity *sim*(*m, mʹ*) for some threshold .
* The researchers used Tanimoto similarity with Morgan fingerprint as the similarity metric and penalized logP coefficient as their target chemical property.
* They jointly train a property predictor *F* with JT-VAE to predict from the latent embedding of *m*.
* To optimize a molecule *m*, they start from its latent representation and apply gradient ascent in the latent space to improve the predicted score *F*().
* They use *K* = 80 gradient steps, *K* molecules are decoded from resulting latent trajectories, and then report the molecule with the highest *F*() that satisfies the similarity constraint.
* To check the performance of the model, they selected 800 molecules with the *lowest* property score from the test set.
* Results: - the unconstrained scenario () has the best average improvement but often proposes dissimilar molecules.
* When the constraints are , about 80% of the time model finds similar molecules, with an average improvement of 0.84. It also demonstrates the smoothness of the learned latent space.
* Related Work: - previous work on molecule generation mostly operates on SMILES strings. Unfortunately, these models could generate invalid SMILES that do not result in any molecules.
* To counter these issues Kusner et al (2017); Dai et. al (2018) complemented the decoder with syntactic and semantic constraints of SMILES by context-free and attributes grammars, but these grammars also do not fully capture the chemical validity.
* The researchers’ techniques such as active learning and reinforcement learning encourage the model to generate valid SMILES through additional training signals.
* Graph-structured Encoders: - the neural network formulation on graphs was first proposed by Gori et al. (2015); Scarelli et al (2009), and later enhanced by Li et al (2015) with gated recurrent units.
* For recurrent architectures over graphs, Lei designed the Weisfeiler-Lehman kernel network inspired by graph kernels.
* Dai considered a different architecture where graphs were viewed as latent variable graphical models and derived their model from message passing algorithms.
* The researchers’ model’s tree and graph encoder is closely related to this graphical model perspective, and neural message passing networks.
* For convolutional architectures, Duvenaud et al. (2015) introduced a convolutional-like propagation on molecular graphs, which was generalized to other domains by Niepert.
* For applications, graph neural networks are used in semi-supervised classification, computer vision, and chemical domains.
* Tree-structure Models: - the researchers’ tree encoder is related to recursive neural networks and tree-LSTM.
* These models encode tree structures where nodes in the tree are bottom-up transformed into vector representations. In contrast, their model propagates information both bottom-up and top-down.
* On the decoding side, tree generation naturally arises in natural language parsing. The natural language parsers have access to input words and only predict the topology of the tree.
* The researchers’ model is closely related to Alvarex-Melis & Jaakkola’s architecture that is constructed on trees top-down from the root, it disentangles topological prediction from label prediction, but the researchers generate nodes in a depth-first order and have additional steps that propagate information bottom-up.
* Conclusion: - the researchers’ method significantly outperforms previous work in molecule generation and optimization.
* Supplementary Material: - the second algorithm presents researchers’ tree decomposition of molecules. *V*1 and *V*2 contain non-ring bonds and simple rings respectively. Simple rings are extracted via RDKit’s **Get SymmSSSR** function.
* Then the rings merge, that share three or more atoms as they form bridge compounds.
* The junction tree of the molecule is not unique when its cluster graph contains cycles. It introduces additional uncertainty for researchers' modeling.
* The researchers add intersecting atoms as a cluster and remove the cycle connecting them in the cluster graph.
* At last, they construct a junction tree as the maximum spanning tree of a cluster graph (). They also assign a large weight over edges involving clusters in *V*0 to ensure no edges in any cycles will be selected into the junction tree.
* Stereochemistry: - molecules are defined not only by their atom types and bond connections but also by the spatial configuration between atoms.



* *Stereoisomers* are molecules that have the same 2D structure but differ in the 3D orientations of their atoms in space.
* They found that it is more efficient to predict the stereochemical configuration separately from the molecule generation.
* The JT-VAE first generates the 2D structure of a molecule *m*, then the researchers generate all its stereoisomers using RDKit’s **EnumerateStereoisomers** function, it identifies atoms that could be chiral.
* For each isomer *mʹ*, they encode its graph representation with the graph encoder and compute their cosine similarity *(mʹ*) = cos().
* They reconstruct the final 3D structure by picking the stereoisomer = arg.
* Combining this tree and graph generation, the molecule reconstruction loss becomes



* Training Details: - for the graph encoder, the initial atom features include its atom type, degree, its formal charge, and its chiral configuration.
* For the tree encoder, they represent each cluster with a neural embedding vector, similar to word embedding for words.
* The tree and graph decoder use the same feature settings as encoders. The graph encoder and decoder run three iterations of neural message passing.
* More Experimental Results: - Sampled Molecules – to prove that the model does not converge to trivial solutions, they randomly sample and plot 250 molecules from prior distribution .
* Neighborhood Visualization: - they encode a molecule into the latent space and generate two random orthogonal unit vectors as two axes of a grid. Moving in combinations of these directions yields a set of latent vectors and we decode them into corresponding molecules.
* Bayesian Optimization: - they train a sparse Gaussian process with 500 inducing points to predict the properties of molecules.
* Five iterations of batch BO with an expected improvement heuristic are used to propose new latent vectors.
* In each iteration, 50 latent vectors are proposed, from which molecules are decoded and added to the training set for the next iterations.
* The scores reported are **normalized** to zero mean and unit variance by the mean and variance computed from the training set.
* Constrained Optimization: - a property predictor *F* is trained jointly with VAE to predict from the latent embedding of *m.* *F* is a feed-forward network with one hidden layer of dimension 450 followed by tanh activation.
* To optimize a molecule *m*, they start with its mean encoding and apply 80 gradient ascent steps: = + with = 2.0.
* 80 molecules are decoded from latent vectors {} and their property is calculated. Molecular similarity *sim*(*m,mʹ*) is calculated via Morgan fingerprint of radius 2 with Tanimoto similarity.
* For each molecule m, the researchers report the best-modified molecule mʹ with the *sim*(*m,mʹ*) > for some threshold .