Explainaibility for Graph Networks

What is Explainaibility?
Explainability refers to methods and techniques in the application of machine learning or deep learning models such that the results of the solution can be understood by human experts.

Problem Statement
Many people say machine learning models are "black boxes", in the sense that they can make good predictions but you can't understand the logic behind those predictions. This statement is true in the sense that most data scientists don't know how to extract insights from models yet. However, explainability techniques helps to extract the following insights from sophisticated machine learning models.
1.What features in the data did the model think are most important?
2.For any single prediction from a model, how did each feature in the data affect that particular prediction?
3.How does each feature affect the model's predictions in a big-picture sense (what is its typical effect when considered over a large number of possible predictions)?
4.Which neurons get activated towards this prediction?

Different Methods of Explainaibility

1.Contrastive gradient-based saliency maps is perhaps the most straight-forward and well-established approach. In this approach, one simply differentiates the output of the model with respect to the model input. The negative values in the gradient are discarded to only retain the parts of input that positively contribute to the solution.
2.Class Activation Mapping provides an improvement over saliency maps for convolutional neural networks, including GCNNs, by identifying important, class-specific features at the last convolutional layer as opposed to the input space. It is well-known that such features tend to be more semantically meaningful (e.g., faces instead of edges).
3.*Layer-wise Relevance Propagation (LRP)* is a method that identifies important pixels by running a backward pass in the neural network. The backward pass is a conservative relevance redistribution procedure, where neurons that contribute the most to the higher-layer receive most relevance from it.

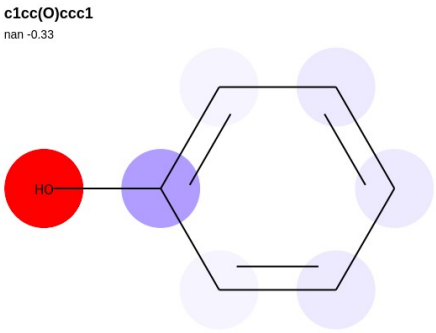
Need for Explainability for Graph Networks
Explainability can be particularly helpful for graphs, even more than for images, because non-expert humans cannot intuitively determine the relevant context within a graph, for example, when identifying groups of atoms (a sub-graph structure on a molecular graph) that contribute to a particular property of a molecule.
Though,there is not much methods to explain a graph network ,we have tried to modify GradCAM method for images and use in our graph networks.

Methodology
1.Get the gradient value before the activation function of the required layer and the respective weight param.
2.Do dot product of the gradient value matrix and respective weight matrix to get a node value tensor containing the logits of each atom in the molecule.
3.Map the node value to visualize which atom relatively contribute how much towards the prediction.

Visualisation of effects of each atom in a molecule towards Solubility(property)

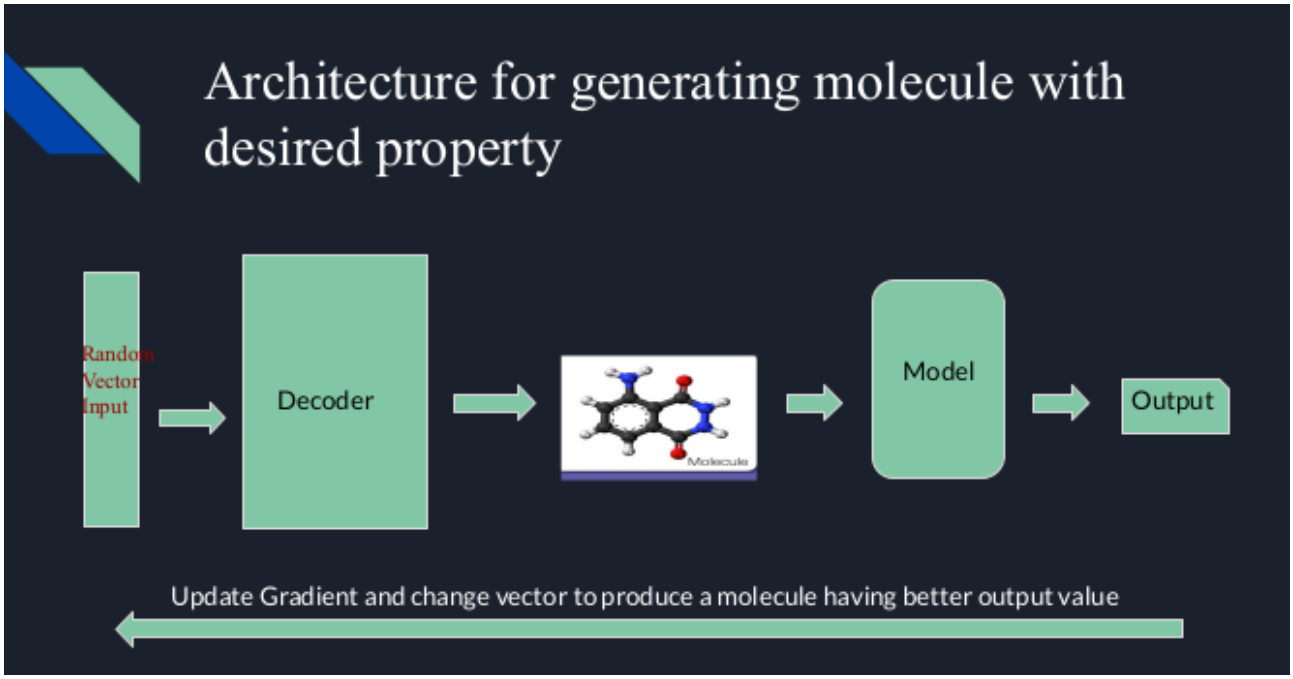
COLOR INDEX:

Range-
Dark Red(Highly positive)
to
Dark Blue(Highly negative



Making graphs features understandable

We have seen which atoms are responsible for your prediction.Now, we are also trying to get which neurons or channels gets activated towards this prediction.For this we came up with a model idea which is discussed below.



Hierarchical Generation of Molecular Graphs using Structural Motifs

Motif Extraction for making Vocabulary

We define a motif Si = (Vi , Ei) as a subgraph of molecule G induced by atoms in Vi and bonds in Ei . Given a molecule, we extract its motifs S1, · · · , Sn such that their union covers the entire molecular graph: V = $S_i V_i$ and E = $S_i E_i$. To extract motifs, we decompose a molecule G into disconnected fragments by breaking all the bridge bonds that will not violate chemical validity (illustrations in the appendix).

1. Find all the bridge bonds (u, v) ∈ E, where both u and v have degree Δu, Δv ≥ 2 and either u or v is part of a ring. Detach all the bridge bonds from its neighbors.

2. Now the graph G becomes a set of disconnected subgraphs G1, · · · , GN . Select Gi as motif in G if its occurrence in the training set is more than Δ = 100.
3. If Gi is not selected as motif, further decompose it into rings and bonds and select them as motif in G.

Hierarchical Graph Generation

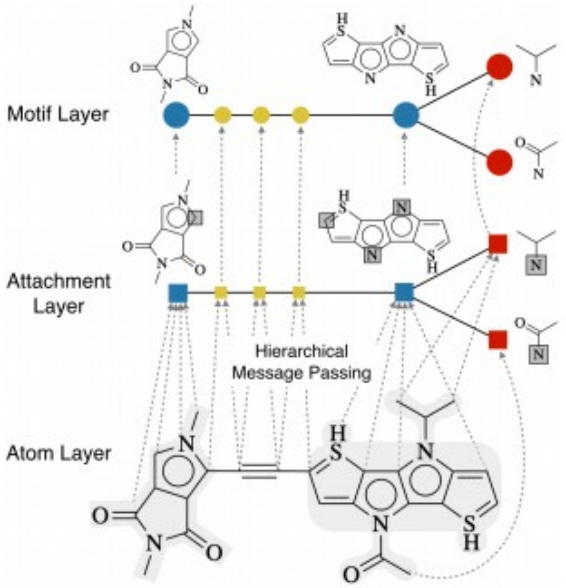


Figure 2. Hierarchical graph encoder. Dashed arrows connect each atom to the motifs it belongs. In the attachment layer, each node $\mathcal{A}_i$ is a particular attachment configuration of motif $\mathcal{S}_i$. The atoms in the intersection between each motif and its neighbors are highlighted in faded block.

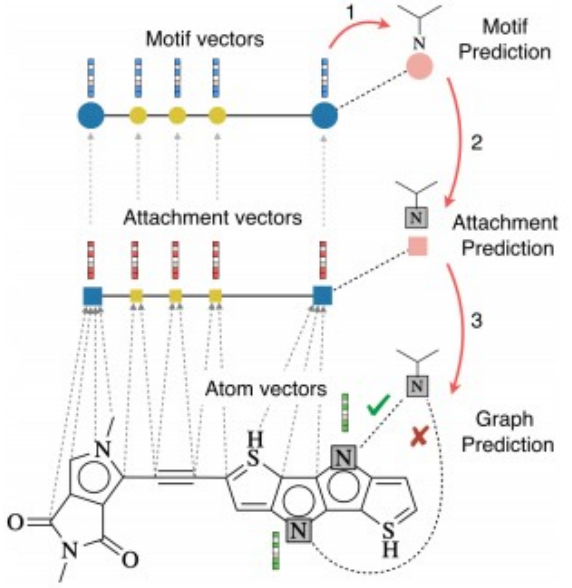


Figure 3. Hierarchical graph decoder. In each step, the decoder first runs hierarchical message passing to compute motif, attachment and atom vectors. Then it performs motif and attachment prediction for the next motif node. Finally, it decides how the new motif should be attached to the current graph via graph prediction.

Our approach extends the variational autoencoder (Kingma & Welling, 2013) to molecular graphs by introducing a hierarchical decoder and a matching encoder. In our framework, the probability of a graph G is modeled as a joint distribution over structural motifs S1, · · · , Sn constituting G, together with their attachments A1, · · · , An. Each attachment Ai = {vj | vj ∈ S k Si ∩ Sk} indicates the intersecting atoms between S_i and its neighbor motifs. To capture complex dependencies involved in the joint distribution of motifs and their attachments, we propose an auto-regressive factorization of P(G):

$$P(\mathcal{G}) = \int_{\boldsymbol{z}} P(\boldsymbol{z}) \prod\nolimits_k P(\mathcal{S}_k, \mathcal{A}_k \mid \mathcal{S}_{<k}, \mathcal{A}_{<k}, \boldsymbol{z}) d\boldsymbol{z} \quad (1)$$

As illustrated in Figure 3, in each generation step, our decoder adds a new motif Sk (motif prediction) and its attachment configuration Ak (attachment prediction). Then it decides how the new motif should be attached to the current graph (graph prediction).
To support the above hierarchical generation, we need to design a matching encoder representing molecules at multiple resolutions in order to provide necessary information for each decoding step. Therefore, we propose to represent a molecule G by a hierarchical graph HG with three layers (see Figure 2):
1. Motif layer: This layer represents how the motifs are coarsely connected in the graph. This layer provides essential information for the motif prediction in the decoding process. Specifically, this layer contains n nodes S1, · · · , Sn and m edges {(Si , Sj) | Si∩Sj != ∅} for all intersecting motifs Si , Sj . This layer is tree-structured due to our way of constructing motifs.
2. Attachment layer: This layer encodes the connectivity between motifs at a fine-grained level. Each node Ai = (Si , {vj}) in this layer represents a particular attachment configuration of motif Si ,

where {vj} are atoms in the intersection between Si and one of its neighbor motifs (see Figure 2). This layer provides crucial information for the attachment prediction step during decoding, which helps reducing the space of candidate attachments between Si and its neighbor motifs. Just like the motif vocabulary V_S , all the attachment configurations of Si form a motif-specific vocabulary $V_A(S_i)$, which is computed from the training set.

3. Atom layer: The atom layer is the molecular graph G representing how its atoms are connected. Each atom node v is associated with a label av indicating its atom type and charge. Each edge (u, v) in the atom layer is labeled with buv indicating its bond type. This layer provides necessary information for the graph prediction step during decoding.

We further introduce edges that connect the atoms and motifs between different layers in order to propagate information in between. In particular, we draw a directed edge from atom v in the atom layer to node Ai in the attachment layer if $v \in S_i$. We also draw edges from A_i to S_i in the motif layer. This gives us the hierarchical graph HG for molecule G, which will be encoded by a hierarchical message passing network (MPN). During encoding, each node S_i is represented as a one-hot encoding in the motif vocabulary V_S . Likewise, each node A_i is represented as a one-hot encoding in the attachment vocabulary $V_A(S_i)$.

Hierarchical Graph Encoder

Our encoder contains three MPNs that encode each of the three layers in the hierarchical graph. For simplicity, we denote the MPN encoding process as MPNψ(·) with parameter ψ, and denote MLP(x, y) as a multi-layer neural network whose input is the concatenation of x and y.

Atom Layer MPN We first encode the atom layer of H_G (denoted as $H^{g}{}_{G}$). The inputs to this MPN are the embedding vectors $\{e(a_u)\}$, $\{e(b_{uv})\}$ of all the atoms and bonds in G. During encoding, the network propagates the message vectors between different atoms for T iterations and then outputs the atom representation hv for each atom v:

$$\boldsymbol{c}_{\mathcal{G}}^{g} = \{\boldsymbol{h}_v\} = \text{MPN}_{\psi_1}\left(\mathcal{H}_{\mathcal{G}}^{g}, \{\boldsymbol{e}(a_u)\}, \{\boldsymbol{e}(b_{uv})\}\right) \quad (2)$$

Attachment Layer MPN The input feature of each node Ai in the attachment layer Ha G is an concatenation of the embedding e(A_i) and the sum of its atom vectors $\{h_v \mid v \in Si\}$:

$$\boldsymbol{f}_{\mathcal{A}_i} = \text{MLP}\left(\boldsymbol{e}(\mathcal{A}_i), \sum_{v \in \mathcal{S}_i} \boldsymbol{h}_v\right) \quad (3)$$

The input feature for each edge (A_i , A_j) in this layer is an embedding vector e(d_{ij}), where d_{ij} describes the relative ordering between node A_i and A_j during decoding. Specifically, we set d_{ij} = k if node Ai is the k-th child of node A_j and d_{ij} = 0 if A_i is the parent. We then run T iterations of message passing over $H^{a}{}_{G}$ to compute the motif representations:

$$\boldsymbol{c}_{\mathcal{G}}^{a} = \{\boldsymbol{h}_{\mathcal{A}_i}\} = \text{MPN}_{\psi_2}\left(\mathcal{H}_{\mathcal{G}}^{a}, \{\boldsymbol{f}_{\mathcal{A}_i}\}, \{\boldsymbol{e}(d_{ij})\}\right) \quad (4)$$

Motif Layer MPN Similarly, the input feature of node Si in this layer is computed as the concatenation of embedding e(Si) and the node vector h_{Ai} from the previous layer. Finally, we run message passing over the motif layer $H^{s}{}_{G}$ to obtain the motif representation:

$$\boldsymbol{f}_{\mathcal{S}_i} = \text{MLP}\left(\boldsymbol{e}(\mathcal{S}_i), \boldsymbol{h}_{\mathcal{A}_i}\right) \quad (5)$$

$$\boldsymbol{c}_{\mathcal{G}}^{s} = \{\boldsymbol{h}_{\mathcal{S}_i}\} = \text{MPN}_{\psi_3}\left(\mathcal{H}_{\mathcal{G}}^{s}, \{\boldsymbol{f}_{\mathcal{S}_i}\}, \{\boldsymbol{e}(d_{ij})\}\right) \quad (6)$$

Finally, we represent a molecule G by a latent vector zG sampled through reparameterization trick with mean μ(hS1) and log variance Σ(h_{S1}):

$$\boldsymbol{z}_{\mathcal{G}} = \boldsymbol{\mu}(\boldsymbol{h}_{\mathcal{S}_1}) + \exp(\boldsymbol{\Sigma}(\boldsymbol{h}_{\mathcal{S}_1})) \cdot \epsilon; \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (7)$$

where S_1 is the root motif (i.e., the first motif to be generated during reconstruction)

Hierarchical Graph Decoder

As illustrated in Figure 3, our graph decoder generates a molecule G by incrementally expanding its hierarchical graph. In t th generation step, we first use the same hierarchical MPN architecture to encode all the motifs and atoms in $H^{(t)}_G$, the (partial) hierarchical graph generated till step t. This gives us motif vectors h_{Sk} and atom vectors hvj for the existing motifs and atoms.

During decoding, the model maintains a set of frontier nodes F where each node Sk ∈ F is a motif that still has neighbors to be generated. F is implemented as a stack because motifs are generated in their depth-first order. Suppose Sk is at the top of stack F in step t, the model makes the following predictions conditioned on latent representation z_G:

Motif Prediction: The model predicts the next motif S_t to be attached to S_k. This is cast as a classification task over the motif vocabulary Vs:

$$\boldsymbol{p}_{\mathcal{S}_t} = \text{softmax}(\text{MLP}(\boldsymbol{h}_{\mathcal{S}_k}, \boldsymbol{z}_{\mathcal{G}})) \qquad (8)$$

Attachment Prediction: Now the model needs to predict the attachment configuration A_t of motif S_t (i.e., what atoms $v_j \in S_t$ belong to the intersection of St and its neighbor motifs). This is also cast as a classification task over the attachment vocabulary

$$\boldsymbol{p}_{\mathcal{A}_t} = \text{softmax}(\text{MLP}(\boldsymbol{h}_{\mathcal{S}_k}, \boldsymbol{z}_{\mathcal{G}})) \qquad (9)$$

This prediction step is crucial because it significantly reduces the space of possible attachments between S_t and its neighbor motifs.

Graph Prediction: Finally, the model must decide how St should be attached to Sk. The attachment between St and Sk is defined as atom pairs $M_{tk} = \{(u_j, v_j) \mid u_j \in A_k, v_j \in A_t\}$ where atom u_j and v_j are attached together. The probability of a candidate attachment M is computed based on the atom vectors h_{uj} and h_{vj}

$$\boldsymbol{p}_M = \text{softmax}(\boldsymbol{h}_M \cdot \boldsymbol{z}_{\mathcal{G}}) \qquad (10)$$

$$\boldsymbol{h}_M = \sum_j \text{MLP}(\boldsymbol{h}_{u_j}, \boldsymbol{h}_{v_j}) \qquad (11)$$

The number of possible attachments are limited because the number of attaching atoms between two motifs is small and the attaching points must be consecutive.

The above three predictions together give an autoregressive factorization of the distribution over the next motif and its attachment. Each of the three decoding steps depends on the outcome of previous step, and predicted attachments will in turn affect the prediction of subsequent motifs.

Training During training, we apply teacher forcing to the above generation process, where the generation order is determined by a depth-first traversal over the ground truth molecule. Given a training set of molecules, we seek to minimize the negative ELBO:

$$-\mathbb{E}_{\boldsymbol{z}\sim Q}[\log P(\mathcal{G}|\boldsymbol{z})] + \lambda_{\text{KL}} \mathcal{D}_{\text{KL}}[Q(\boldsymbol{z}|\mathcal{G})||P(\boldsymbol{z})] \qquad (12)$$

References

1.https://github.com/baldassarreFe/graph-network-explainability/tree/master/notebooks
2.https://graphreason.github.io/papers/25.pdf
3.https://arxiv.org/pdf/2002.03230.pdf
4.https://github.com/wengong-jin/hgraph2graph