Figure 2 provides an example of the feature extraction procedure of atom 2 in the Butyramide molecule considering interested radius of 1. In particular, this algorithm involves three stages:

1. Initial Stage: Each atom in the molecule is assigned with a unique integer identifier which is generated by hashing a set of chemical properties.
2. Updating Stage: After initialization, each atom's identifier will be updated iteratively. Starting by radius $r = 1$:
   (a) An array will be formed by collecting the radius and the core atom's current identifier.
   (b) Next, the neighboring information of the atom that is r hops away from the core atom will be incorporated into the array. Ranked by the bond order (single, double, triple, and aromatic), the bond order and the current identifier of the interested atom are appended to the array.
   (c) Then the same hash function used in the initial stage is applied again to convert the array into a new integer identifier.
   (d) The above procedure is repeated for each atom in the molecule.
   (e) The radius will be updated as $r := r + 1$ and another iteration to update identifiers for all atoms will be started, unless r has already met the user's interested radius.
3. Reduction Stage: Eventually, for each atom, all the identifiers ever generated in the updating stage are converted into a 64-bit binary vector by calculating the modulus of the decimal integer and concatenated to form the final atom feature vector of length $64 \times (\text{radius} + 1)$.In the typical ECFP algorithm, the updating stage is aimed at discovering the unique substructures in the molecule, which will be consequently integrated into the fingerprint serving as a molecular-level representation. Thus, after the updating iterations, there will be a duplicate structure removal stage to eliminate the identical features which encapsulates the same surrounding environment of atoms. However, on the contrary to a molecular-level representation, in our case we require an atom-level feature where the structural duplication amongst atoms is not hampering feature reduction. Besides, the original algorithm only considers the last generated identifiers upon reducing them into the fingerprint, which is effective in producing the unique fingerprint of the molecule, whereas we preserve all the ever-generated identifiers to produce the atom-level features. This is because the last identifier is always computed considering a relatively large radius. The surrounding substructure might thus be identical for different atoms. Therefore, if merely considering the last identifier, certain atom-level features may be duplicated, and the corresponding atoms will be indistinguishable. Under such circumstances, identifiers generated at all radius levels are appended to form the atom-level feature.

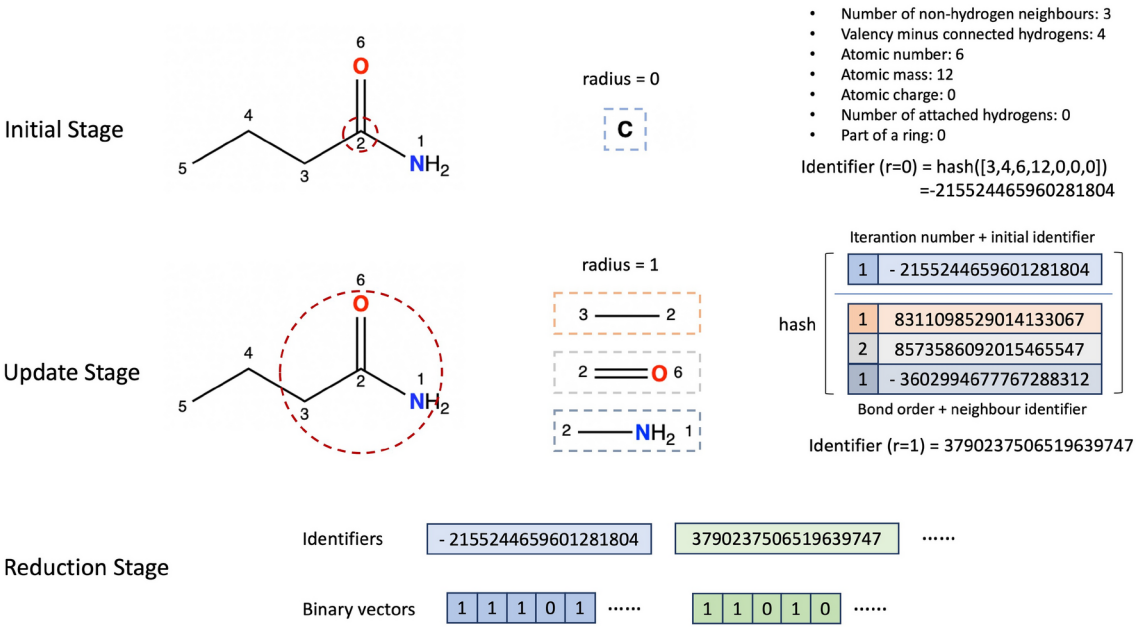

Fig. 2. Illustration of feature extraction procedure of atom 2 in the Butyramide molecule. In the initial stage, chemical features including number of non-hydrogen neibours, valency, atomic number, etc., are extracted and hashed to compute the initial identifier of each atom. In the update stage, starting from radius of 1, the bond orders and identifiers of the surrounding atoms (atom 3, 6 and 1) are combined and concatenated with the iteration number and the initial identifier of the focused atom (atom 2). The hash function is applied again on the concatenated feature to form the new identifier. Finally, in the reduction stage, each of the identifiers of atom 2 generated in the update stage are converted to a 64-bit binary vector and concatenated to form the final atom feature. In our implementation, we combined the binary vectors of radius 0, 1, 2 and 3, which forms a the final feature vector of length $4 \times 64 = 256$.

Computational framework

We utilize Graph Neural Networks (GNN) to learn the latent features of drugs' molecular graphs and Convolutional Neural Networks (CNN) to learn the representation of the gene expression data, and combine them together to predict the response level as shown in Fig. 1. Instead of concatenating latent features of drug and cell line as tCNN[11] and GraphDRP[25], we propose to leverage multi-head attention mechanism introduced by Transformer[41] to integrate the drug and cell line features effectively.

Each head H_i in the multi-head attention module can be formulated as

$$H_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{1}$$

where *Q*, *K* and *V* stand for the query, key and value used in an attention layer. To obtain the drug embed influenced by gene expressions, we use drug features encoded by GNN as *Q* and cell line features encoded by CNN as *K* and *V*. On the contrary, to learn the gene embed, we use cell line features as *Q* and drug features as *K* and *V*. Eventually, the integrative features are combined and fed into a predictor composed of a dense layer for drug response prediction.

After developing the model, we adopt Integrated Gradients[34] and GNNExplainer[33] to explore the saliency of inputs, i.e., atoms and bonds of the drug molecule and transcriptomic features of the cell line, which reveals the reaction mechanism of cancer cell lines and drugs.

Graph neural networks (GNN)

After constructing the molecular graphs and extracting atom-level features for the drugs, we develop GNN models to further learn the latent representation of the drugs. In this work, we take advantage of four types of GNN models and compare their performance in drug response and mechanism prediction: Graph Convolutional Networks (GCN)[42], Graph Attention Networks (GAT)[43], Relational Graph Convolutional Networks (RGCN)[44], and Relational Graph Attention Networks (RGAT)[45]. Similarly, the idea for such GNN models is to aggregate the information from a node itself and its neighborhood.

If we define the atom set in a drug's molecule as *V* and the bond set as *E*, the molecular graph of this drug can be given by $G = (V, E)$. Then we use an adjacent binary matrix $A \in \mathbb{R}^{N \times N}$ to represent the edge connection between nodes where *N* is the number of atoms, $a_{i,j} = 1$ denotes a connection between node *i* and *j*, and $a_{i,j} = 0$ denotes no connection. Additionally, a feature matrix $X \in \mathbb{R}^{N \times M}$ is used for representing the node features of atoms where *M* is the dimension of feature vector that has been extracted by the algorithm aforementioned.

The GCN layer is defined by

$$h_i = W \sum_{j \epsilon N_i \bigcup \{i\}} \frac{\widetilde{a}_{ij}}{\sqrt{\widetilde{d}_i \widetilde{d}_j}} x_i \tag{2}$$

where $\widetilde{A}$ is the adjacent matrix adding a self loop, $\widetilde{D}$ is the diagonal degree matrix with $\widetilde{d}_{i,i} = \sum_j \widetilde{a}_{i,j}$. x_i denotes the node feature vector, *W* is the weight matrix, and N_i is the neighbor node set of node *i*.

For the GAT layer, the attention coefficient of node *i* and *j* is defined as $e_{i,j} = a(Wx_i, Wx_j)$ according to[43], and is only computed when node *j* is in the neighborhood of node *i*. Then the GAT layer can be given by

$$h_i = \alpha_{i,i} W x_i + \sum_{j \epsilon N_i} \alpha_{i,j} W x_j \tag{3}$$

where x_i is the node feature vector, *W* is the weight matrix, N_i is the neighbor node set of node *i*, and $\alpha_{i,j}$ is the normalized attention coefficients with softmax function.

Notably, one drawback when adopting the typical GCN and GAT layers on molecular graphs is that they both dismiss the edge properties of the graph whereas the varying chemical bond types in a molecule could also impose a crucial impact on the drug's functional mechanism. In order to tackle this problem, we encode the chemical bond type (single, double, triple, and aromatic) into the edge features, which can be used for updating edges in the message passing procedure in GNN models. Edge features are directly supported by GAT layer and GATv2 layer which is designed to fix the static attention problem of original GAT layer[46].

To further investigate the effectiveness of edge features, we look into RGCN and RGAT models, which consider edge types as relations and differentiate the message passing patterns according to various relation types. In molecular graphs, edges represent chemical bonds that naturally possess disparate characteristics and should be treated accordingly. Therefore, we attempt to leverage RGCN and RGAT models to represent a molecule more precisely. Considering there are *R* relations in total, the RGCN layer can be defined as

$$h_i = W^{(root)} x_i + \sum_{r \epsilon R} \sum_{j \epsilon N_i^{(r)}} \frac{1}{|N_i^{(r)}|} W^{(r)} x_j \tag{4}$$

where $N_i^{(r)}$ denotes the neighbor node set of node *i* under relation *r*. Unlike general GCN layer, RGCN layer learns different weights specific to relation types. $W^{(r)}$ represents the weights corresponding to relation *r*, and $W^{(root)}$ corresponds to a special self-connected relation that is not included in *R*.

Similar to the way RGCN creates relation-specific transformations to update node representations, RGAT also proposes relation-specific attention weights for message aggregation. If we compute the attention coefficient of node *i* and *j* under relation *r* as $e_{i,j}^{(r)} = a(Wx_i^{(r)}, Wx_j^{(r)})$, RGAT layer can be formulated as

$$h_i = \sum_{r \epsilon R} \sum_{j \epsilon N_i^{(r)}} \alpha_{i,j}^{(r)} x_j^{(r)} \tag{5}$$

where $N_i^{(r)}$ denotes the neighbor node set of node *i* under relation *r* and $\alpha_{i,j}^{(r)}$ is the normalized attention coefficients with softmax function. Notably, the softmax function can be applied either over only attention coefficients under single relation type or all attention coefficients regardless of relation types, which result in within-relation GAT (WIRGAT) and across-relation GAT (ARGAT), respectively. In our experiments, ARGAT is found to outperform WIRGAT and is thereby used in the subsequent analysis.

Hyperparameters such as the radius in atom feature extraction, number of neural network layers, hidden sizes and dropout rates are searched to develop the best model. We first implemented GCN and GAT-based XGDP with the features extracted with radius of 0, 1, 2 and 3, and found radius of 3 obtained the best and most stable performace on the validation set. Grid search is then conducted to find the optimal parameters for number of layers of both GNN and CNN in [1, 2, 3, 4, 5], hidden sizes in [128, 256, 512] and dropout rates in [0, 0.1, 0.2, 0.3, 0.4, 0.5] . Finally, the number of layers is set to 2 for the GNN module and 3 for the CNN module. The hidden size is set to 128. And the dropout rate is configured as 0.5.

Model interpretability

To explore our proposed model's interpretability, we leverage on GNNExplainer[33] to identify the functional groups of molecular graphs and Integrated Gradients[34] implemented by Captum[47] to track the attributes of the genes in cancer cell line data.

Integrated gradients

Integrated Gradients is a gradient-based attribution method proposed by Subdararajan et al.[34]. Integrated Gradients is designed to satisfy two fundamental axioms, i.e., sensitivity and implementation invariance, and thus generate more reasonable explanations of neural network models than previous approaches such as Gradient * Input[48], Layer-wise Relevance propagation (LRP)[49], and DeepLIFT[50].

A prerequisite for a reasonable attribution using Integrated Gradients is to identify a baseline input. Take image networks as an example, the baseline inputs can be pixels equal to zero, constituting a black image. In our case, however, baseline cannot be simply set to zeros, since genes are seldomly expressed as zeros and picking zeros as baseline will render the explanation biased to certain genes with relatively high expression values. Our intention is to compute the average normal expression level as background for each gene, and study the effect when a gene is differentially expressed. Hypothesising that genes are normally expressed in most cell lines, we propose to identify suspiciously abnormal values with an interquartile range (IQR) filter. For each gene, we calculate IQR of its expression values on all the cell lines. Then expression values that are more than 2.22 times the IQR away from the median of the data are considered as outliers, which is roughly equivalent to remove the data points that have a z-score larger than 3 in the normal distribution. Thereafter we remove the outliers and compute the average of preserved expression values as the baseline of each gene.

After deciding the baseline input, Integrated Gradients computes the integral of the gradients along the path from the baseline input to the actual input. If we denote the actual input as *x* and the baseline input as $x\prime$, the integrated gradients can be defined by

$$Attribution_i\,(x) = (x_i - x'_i) \times \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \times (x - x'))}{\partial x_i} d\alpha \tag{6}$$

where *i* refers to the interested dimension of inputs, and α is the interpolated value from $x\prime$ to *x*.

GNNExplainer

Gradient-based methods (e.g., Integrated Gradients) is suitable to explain models built on grid-like data in the text or image domain. However, GNN models built in the graph domain are developed to capture the structural information of graphs, and interpreting such models requires to explore how messages are passed through the graph structures[51]. GNNExplainer is one of the explanation methods aiming at analyzing models built in the graph domain. Comparing with gradient-based methods, GNNExplainer has been testified to be capable of capturing reasonable substructures of graphs such as functional groups of molecules, in the task of molecular

property prediction[33]. Therefore, in this work, we adopt GNNExplainer to interpret the graph convolutional layers and identify the active functional groups of molecular graphs.

The theory of GNNExplainer is to identify the most salient subgraph and subset of node features for the model's prediction. It can be formulated in an optimization problem:

$$\max_{G_S} MI(Y, (G_S, X_S)) = H(Y) - H(Y|G = G_S, X = X_S) \tag{7}$$

where the mutual information *MI* reflects the change of model's output when using a subgraph G_S and subset of node features X_S. The prediction of the model can be given by $Y = \Phi(G, X)$, where Φ represents the function of the model, and *G* and *X* denote the input graph and node feature matrix. Although it is infeasible to retrieve the optimal subgraphs and feature subsets to solve the above problem directly, GNNExplainer has proposed their optimization framework to identify high-quality explanations in an empirical way.

Drug response prediction

In this section, we present the results of drug sensitivity prediction and the saliency maps of inputs. We experiment with various GNN models with and without involving edge features, and compare their performance with four baseline models, i.e., tCNN[11], GraphDRP[25], DeepCDR[26] and TGSA[27]. Particularly, gene expression data are used as cell line features in place of CNV data used in the original research of tCNN and GraphDRP. DeepCDR and TGSA focused on incorporating multi-omics profiles, whereas our work intends to investigate better profiling of drug features. For a fair comparison, we used the gene expression only version of DeepCDR and TGSA to explore if our proposed method properly represents the drugs and leads to better prediction. In addition, we decode the developed models to explore the salient functional groups of drug molecules and biomarkers that are potentially responsible for the biochemical activities.

Our models were implemented with PyTorch[52] and PyTorch Geometric[53] libraries. The performance of our experiments are evaluated by Root Mean Square Error (RMSE), Pearson Correlation Coefficient (PCC) and Coefficient of Determination (R^2). We performed a 3-fold cross-validation on our dataset. The mean and the standard deviation of the evaluation metrics obtained on the validation set are reported in the following sections.

Rediscovery of known drug and cell line responses

To test XGDP with the task of rediscovering response levels of known drugs and cell lines, we randomly shuffle all the pairs of drug and cell line and divide the dataset as described above. This strategy ensures one combination of drug and cell line can present only once in training, validation or testing set, but each drug or cell line can emerge simultaneously in all sets. The rediscovery task is designed to evaluate if the model is able to learn the reaction pattern of a drug from its response data with several cell lines, and predict the response levels between the drug and other unknown cell lines.

Table 1 presents the performances of the proposed method with different GNN layer and compares them with the baseline models. In the table, GAT_E and GATv2_E refer to GAT and GATv2 convolution with incorporation of bond types as edge features. It is shown that XGDP with GAT achieves the highest PCC and R^2 values, and all XGDP variants and the tCNN model achieve the lowest RMSE. DeepCDR and TGSA with only expression data obtain the worst RMSE, which is sensible since their research focus lie on incorporation of multi-omics profiles for drug response prediction. Compared with GraphDRP, our method extends the node features with the circular atomic descriptor as illustrated in Algorithm 1, and introduces multi-head attention to integrate the hidden features of drug and cell line rather than simple concatenation. It is evident in Table 1 that our refinement in the architecture leads to a better performance, especially on models with GAT convolution. Compared with tCNN, GAT- and GAT_E-based XGDP outperform tCNN on both PCC and R^2. Moreover,

Method	Conv type	RMSE (↓)	PCC (↑)	R^2 (↑)
tCNN[11]	CNN	0.026 ± 0.000	0.920 ± 0.001	0.846 ± 0.001
GraphDRP[25]	GCN	0.027 ± 0.000	0.917 ± 0.001	0.840 ± 0.003
	GAT	0.042 ± 0.002	0.828 ± 0.011	0.609 ± 0.034
DeepCDR (exp)[26]	GCN	1.496 ± 0.018	0.841 ± 0.003	0.532 ± 0.057
TGSA (exp)[27]	GraphSAGE	1.072 ± 0.014	0.919 ± 0.002	0.845 ± 0.004
XGDP	GCN	0.026 ± 0.000	0.918 ± 0.001	0.843 ± 0.002
	GAT	0.026 ± 0.000	**0.923 ± 0.000**	**0.851 ± 0.001**
	GAT_E	0.026 ± 0.000	0.922 ± 0.001	0.849 ± 0.001
	GATv2_E	0.026 ± 0.000	0.921 ± 0.001	0.846 ± 0.001
	RGCN	0.026 ± 0.000	0.920 ± 0.001	0.845 ± 0.001
	RGAT	0.026 ± 0.000	0.920 ± 0.001	0.846 ± 0.002

Table 1. Performance of proposed and baseline models in the task of rediscovering known drug and cell line responses. All the models, except GraghDRP-GAT, achieve similar RMSE (~0.26). Best PCC and R^2 (marked in bold) is achieved by XGDP-GAT.