nature portfolio

https://doi.org/10.1038/s42256-021-00325-y

Supplementary information

Integration of multiomics data with graph convolutional networks to identify new cancer genes and their associated molecular mechanisms

In the format provided by the authors and unedited

Supplementary Materials

January 27, 2021

1 Results

1.1 EMOGI Hyper-Parameter Optimization

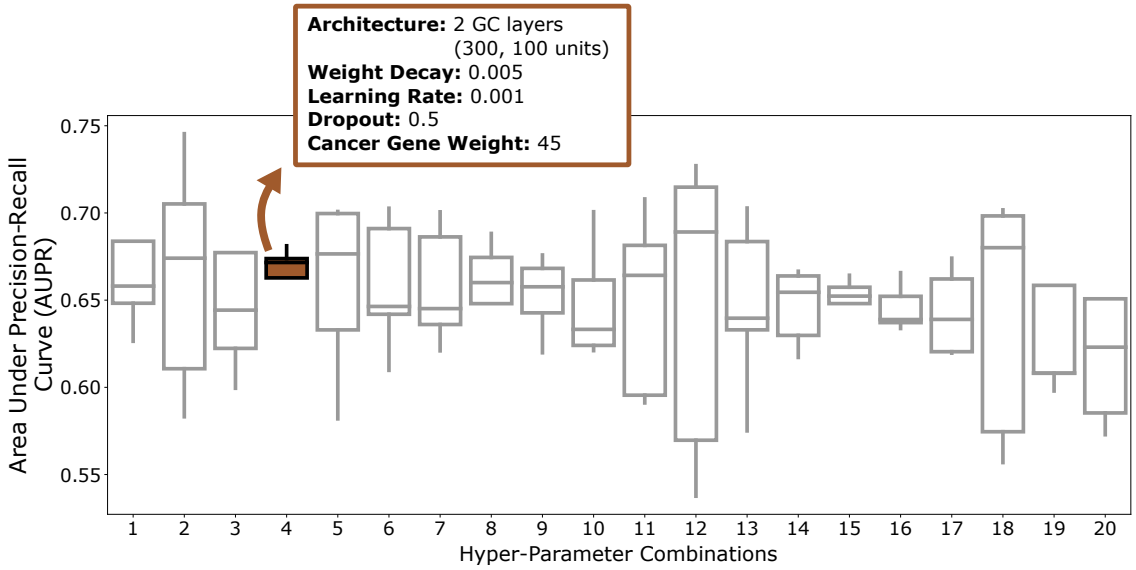


Figure S1: **Best-performing combinations of hyper-parameters.** We conducted a grid search over a reasonable number of values for each hyper-parameter. We then trained an EMOGI model on the CPDB PPI network for every possible combination of those values, corresponding to 288 combinations in total. To obtain robust results independent of the random initialization of the network weights, we used 5-fold cross validation (see figure S20). To determine the best-performing hyper-parameters, we used the combination that yielded the highest median area under the precision-recall curve (AUPRC) values after 2000 training epochs, but was also robust, i.e. had low variance across CV folds. We selected the combination with the 4^{th} highest average AUPRC (marked in orange) because it yielded the best compromise between robustness and performance. For the EMOGI pan-cancer model the optimal hyper-parameters resulted in a dropout rate of 0.5, a learning rate $\eta = 0.001$, a weight factor of 45 for the positive class to compensate for class imbalance, a weight decay of 0.005 and two graph convolutional layers with 300 and 100 filters, respectively. Hyper-parameter optimization for the cancer type specific models yielded only 50 and 10 filters per layer and $\eta = 0.0005$. For breast cancer, we multiplied the loss for positives by 60 and for thyroid cancer by 600 to compensate for the higher class imbalance. Averaged and patient-wise models used the same training and test data.

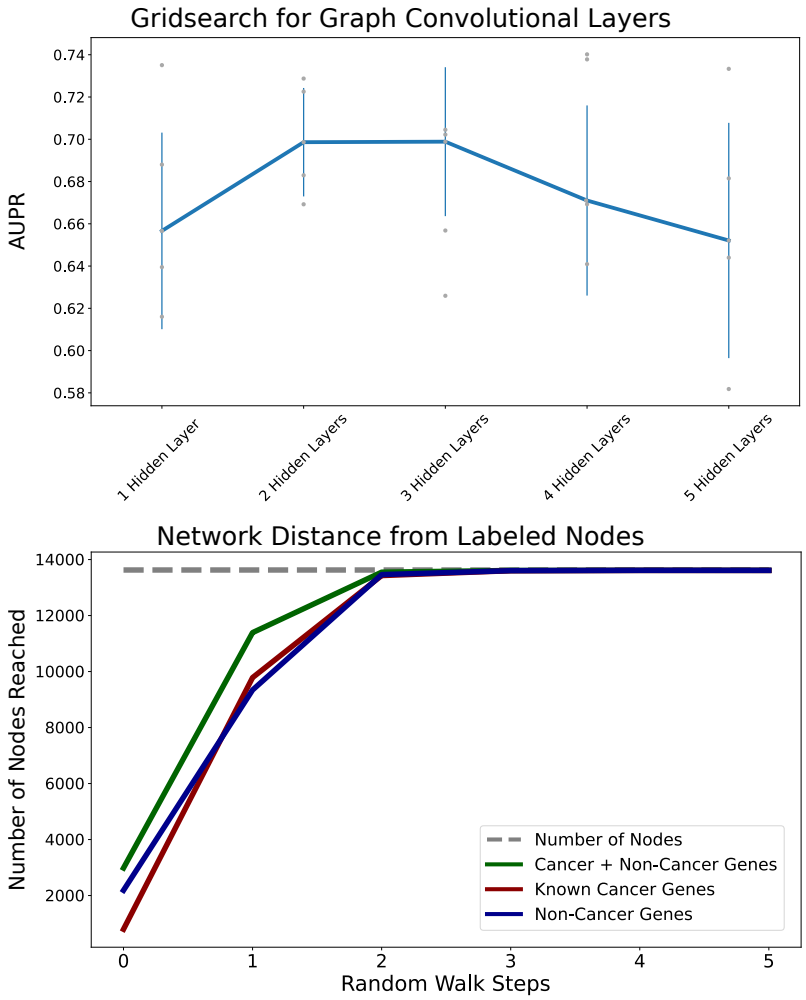


Figure S2: **Investigating the optimal number of graph convolutional layers.** After selecting the optimal hyper-parameters, we conducted a separate grid search to assess the influence of the number of graph convolutional layers specifically. We fixed all hyper-parameters, apart from the number of layers, using the optimal values according to Fig. S1. As in the general hyper-parameter search, we used 5-fold cross-validation and assessed the aggregated performance using the median AUPRC values. We find that 2 or 3 graph convolutional layers yielded the best performance. To further understand this finding, we investigated the distance from the set of labeled genes to any other gene in the CPDB PPI network using random walks. In detail, we performed random walks for three different seed sets of genes namely the known cancer genes (positives), non-cancer genes (negatives) and the union of both gene sets. Initially we assign a probability value of 1 to seed nodes and a probability value of 0 to all other nodes. We found that within two hops, the cancer gene labels can be propagated to the entire graph (bottom figure), making two graph convolutional layers a sensible choice in this context. The random walk analysis was performed independently from model selection and hyperparameter tuning of the GCN model, therefore providing an independent validation for the selected number of layers

1.2 EMOGI Recovery of Known and Candidate Cancer Genes

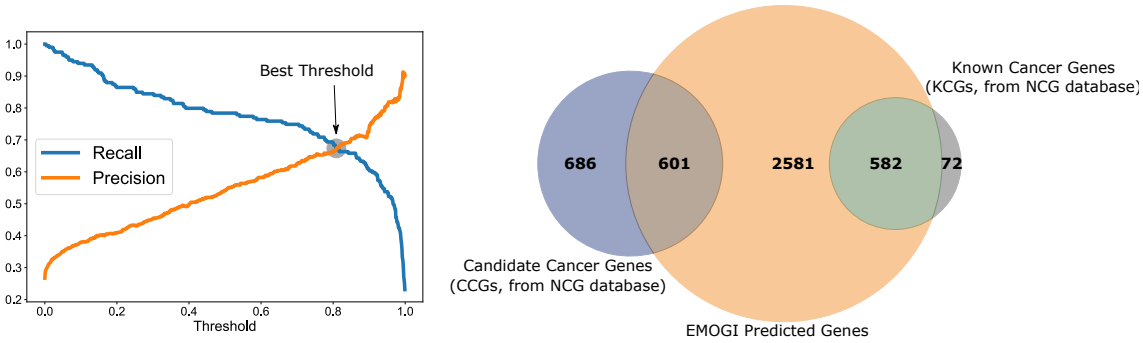


Figure S3: **Overlap of EMOGI's positive predictions with known cancer genes (KCGs) and candidate cancer genes (CCGs) from the NCGs for the CPDB PPI network.** To quantify the overlap between gene sets, we had to choose a threshold on the EMOGI probability score. We chose the cutoff such that it balances precision and recall as depicted on the left. Recall decreases when the cutoff increases because fewer genes are considered positive predictions. Precision, on the contrary, increases when the cutoff increases because the predictions are more and more conservative and only high-confidence predictions are left for high thresholds. The optimal selected threshold, at the intersection between precision and recall, is 0.809 on the CPDB PPI network.

1.3 Performance comparison

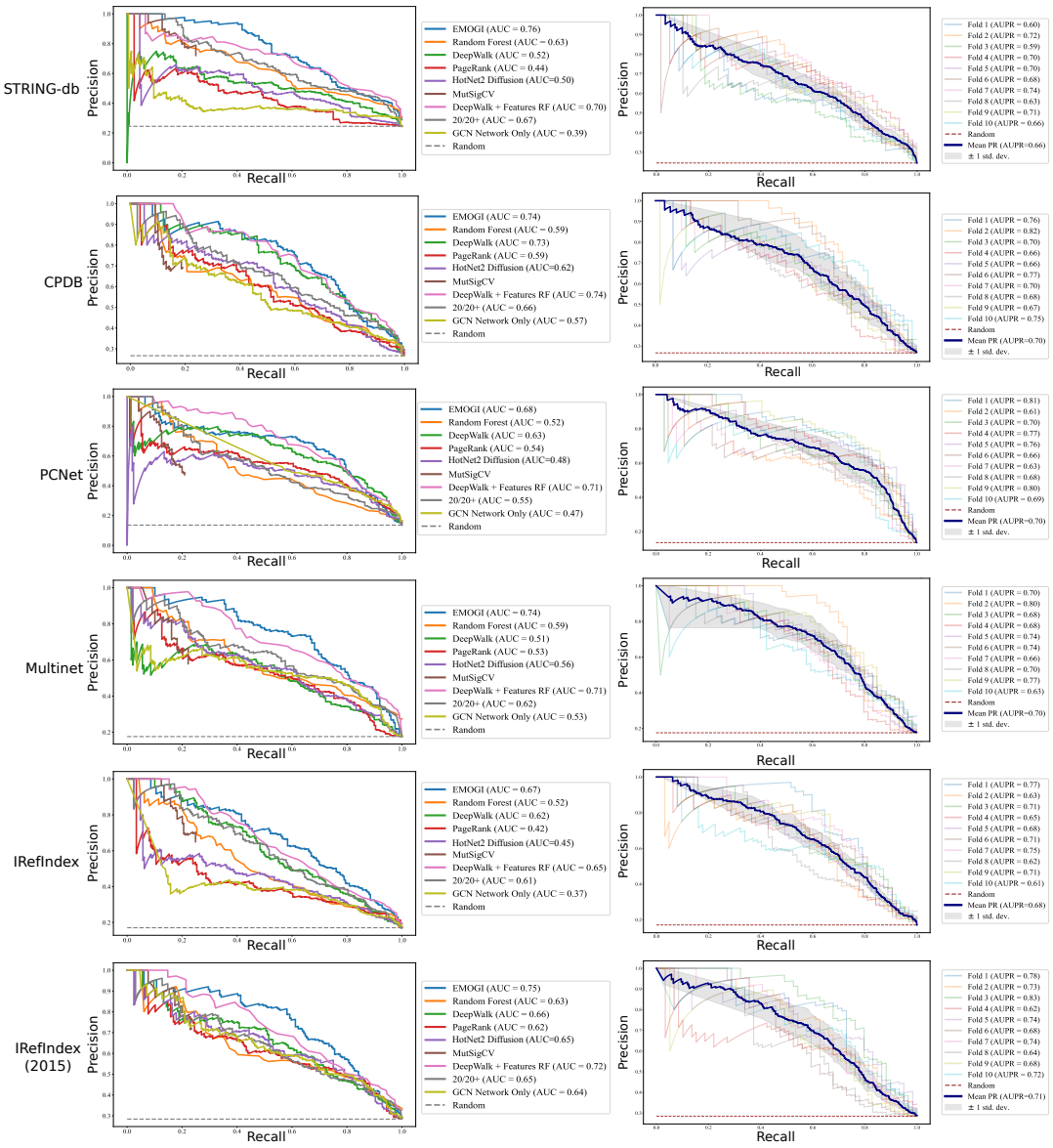


Figure S4: **Precision-recall (PR) curves of different methods for different PPI networks.** We evaluated EMOGI's performance on the test set for 6 different PPI networks and show the comparison to a Random Forest, the PageRank algorithm, the DeepWalk algorithm coupled to an SVM, a GCN trained only on the PPI graph, the DeepWalk algorithm in combination with omics feature and a random forest classifier, the HotNet2 diffusion process, MutSigCV and the 20/20+ method (see subsection 2.6). On the right, PR curves from the individual cross-validation runs are depicted for EMOGI only. In blue, we denote the average PR curve across all 10 cross-validation runs. The gray shading denotes ±1 standard deviation.

1.4 Additional Validation on Independent Cancer Gene Sets

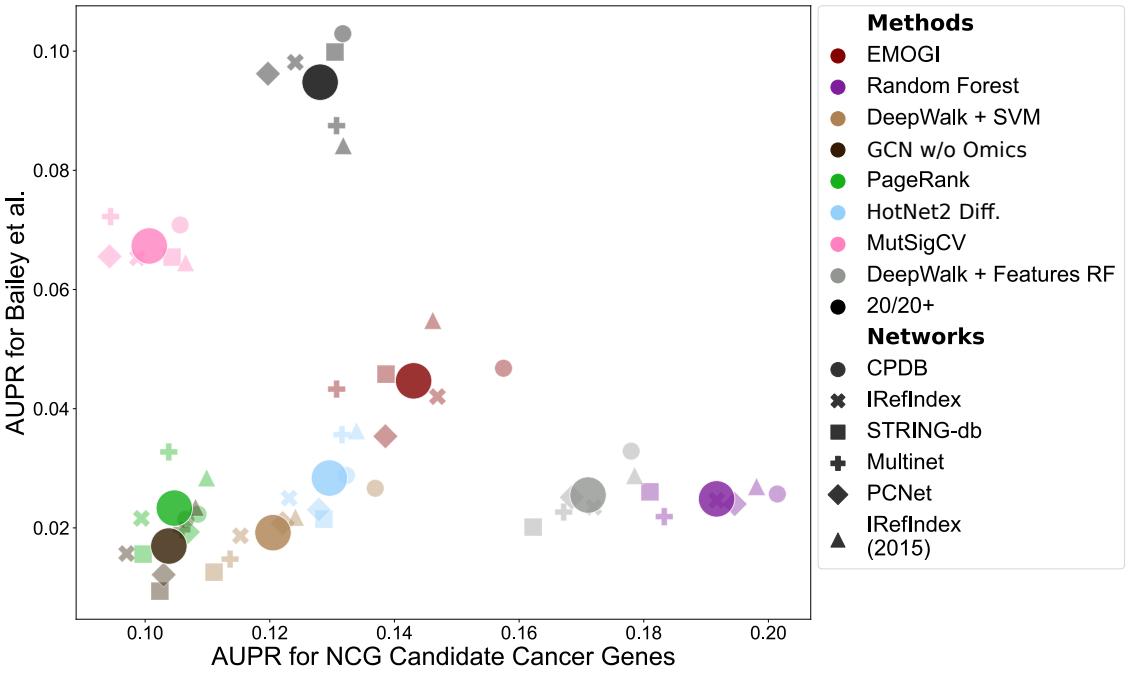


Figure S5: **Performance of EMOGI and the other methods on independent cancer gene sets.** We used the candidate cancer genes from the network of cancer genes (NCG) [1] and the list of predicted cancer genes from Bailey et al. [2] as gene sets for additional validation. Both axes show the area under the precision-recall (AUPRC) curve values of all methods on different PPI networks for both data sets. We only consider hits in the respective gene set as true positives and all other cancer gene predictions as false positives. Therefore, the AUPRC values are smaller compared to Fig. 2a in the manuscript (see section 2.2 for a detailed description of the metrics). The large circles correspond to average AUPRC values across PPIs for each method.

1.5 EMOGI's Performance on Subsets of Features

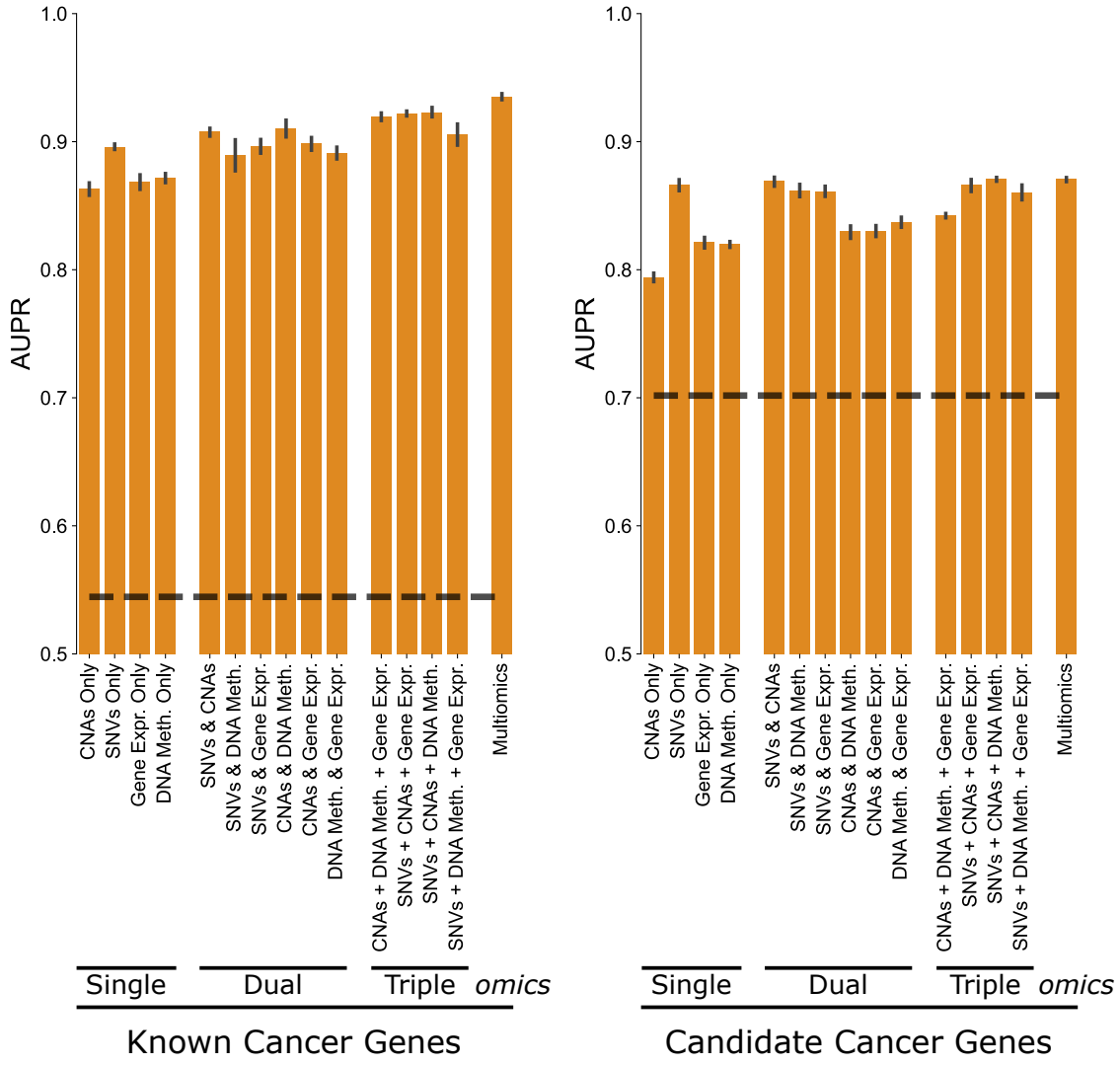


Figure S6: **Performance of EMOGI on subsets of the data types.** We investigated the performance of EMOGI (reported as AUPRC) when only considering subsets of the data for training on the CPDB PPI network. Models are grouped according to using one data type (single), two types of *omics* levels (dual), three types of *omics* (triple) or all four (multi-omics). We assessed performance in correctly recovering known and candidate cancer genes from the NCG. The black dotted line denotes the random performance for both gene sets. Error bars denote standard deviation across the different folds of the cross-validation.

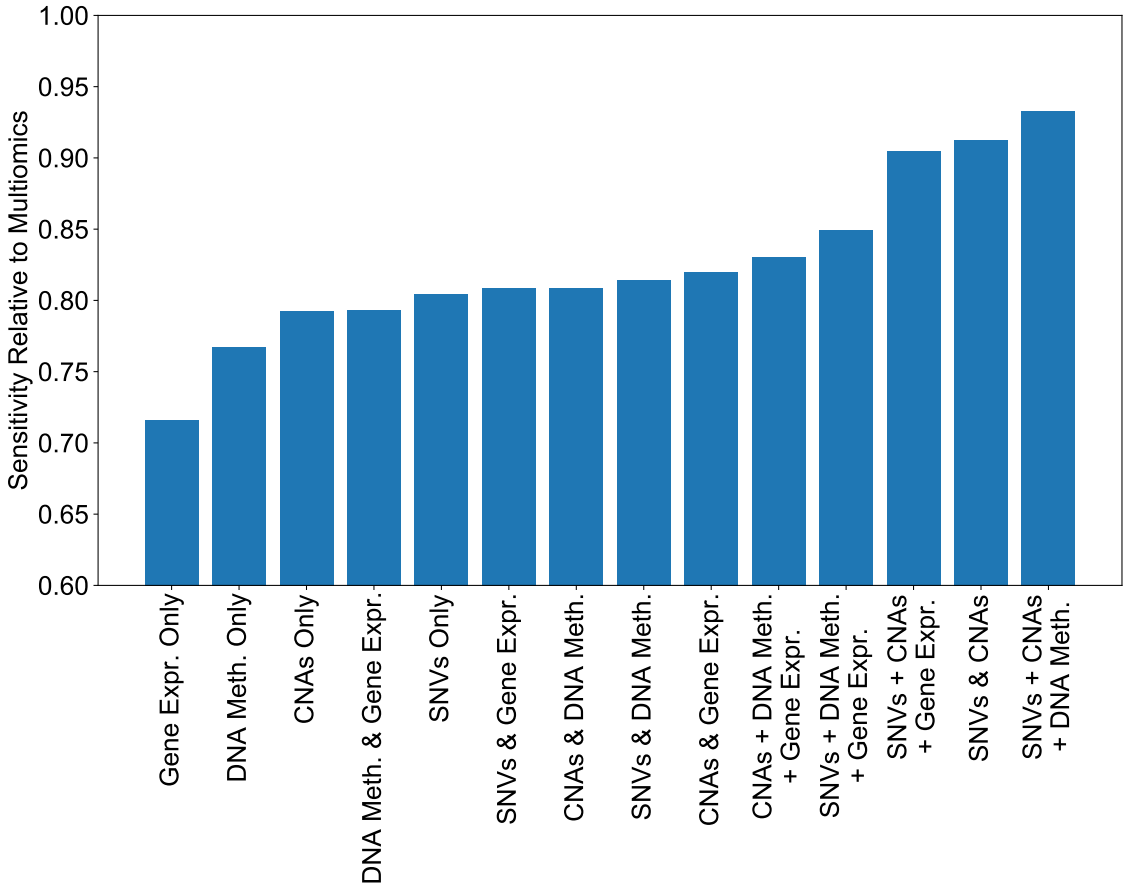


Figure S7: **Sensitivity of EMOGI on different *omics* subtypes, relative to the multi-omics setting.** We used EMOGI models trained on subsets of the original features and the CPDB PPI network to compute the fraction of known cancer genes contained in the predictions (sensitivity) for each of the models. Cutoffs were determined as for Fig. S3, based on the intersection between precision and recall. Bars represent the fraction of sensitivity for each model with respect to the full multi-omics setting and are ranked according to sensitivity. A value of e.g. 0.85 means that EMOGI trained on that subset of feature was able to achieve 85% of the sensitivity of the full model.

1.6 EMOGI is Superior in a Pan-Cancer Setting

Breast Cancer

Sensitivity Analysis

Method	BRCA Training Set	BRCA Test Set	DriverDB Breast Cancer Genes
Pan-Cancer	68%	46%	31%
BRCA Averaged	54%	39%	22%
BRCA Patient-Wise	68%	46%	20%

Thyroid Cancer

Sensitivity Analysis

Method	THCA Training Set	THCA Test Set	DriverDB Thyroid Cancer Genes
Pan-Cancer	85%	59%	46%
THCA Averaged	6%	0%	3%
THCA Patient-Wise	3%	6%	1%

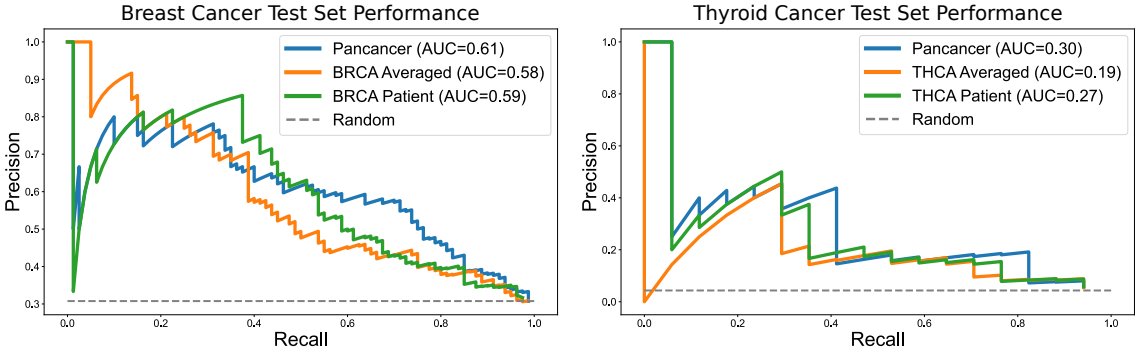


Figure S8: **Comparison of an EMOGI pan-cancer model versus cancer-specific models.** We trained EMOGI on cancer type-specific models for breast and thyroid cancer in two ways: 1) averaging the values of each *omic* across the patients of the specific cancer type and 2) on patient-specific *omics* values as input (see section 2.1.1 and methods section in the main article for details). We collected cancer type-specific cancer genes (positives) for training and testing (see methods, main article for details). We then compared the performance of the *averaged* and the *patient-wise* cancer-specific models with the pan-cancer model. We selected the probability cutoff for all three methods the same way as depicted in Fig. S3. Both *averaged* and *patient-wise* models were trained on the same train/test splits for each cancer type and overlap of the cancer type specific test sets with the pan-cancer training set was removed for the analysis. The pan-cancer EMOGI model recovered the same amount of breast cancer-specific training and test genes as the *patient-wise* breast cancer model, while being more sensitive on an independent set of breast cancer driver genes from DriverDB [3] (table on the top, left side). For thyroid cancer, both cancer-specific models were hard to train due to the low number of labeled examples available for training, and retrieved only a limited number of thyroid-specific cancer specific genes (table on top, right side). The lower panel shows precision-recall curves of the pan-cancer model compared to the cancer-specific models for both breast (left) and thyroid cancer (right). In all cases the pan-cancer model was the best in terms of AUPRC, but only slightly outperformed the *patient-wise* cancer-specific models.

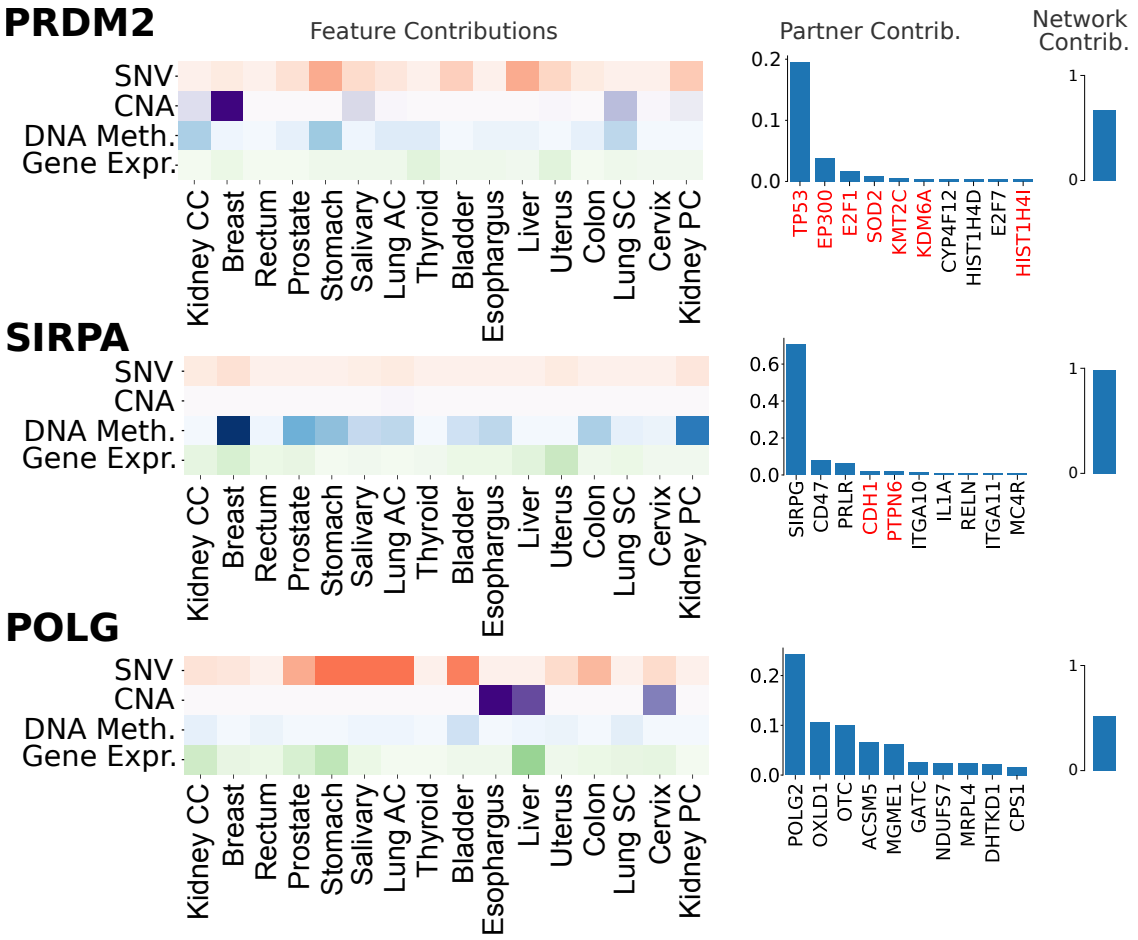


Figure S9: **LRP contributions for selected breast cancer genes from the pan-cancer EMOGI model on the CPDB PPI network.** The depicted genes were predicted by the pan-cancer model but failed to be recovered by both, the *averaged* and *patient-wise* breast cancer models. Contributions were computed using LRP on the pan-cancer model (see section 2.5). Heatmaps depict the importance of individual cancer types for different *omics* levels (darker colors indicate higher importance). The bar plots (middle column) depict the 10 most important interaction partners for the gene of interest and the right column depicts the importance of the overall interactome, compared to the features.

1.7 Explanations for additional cancer genes

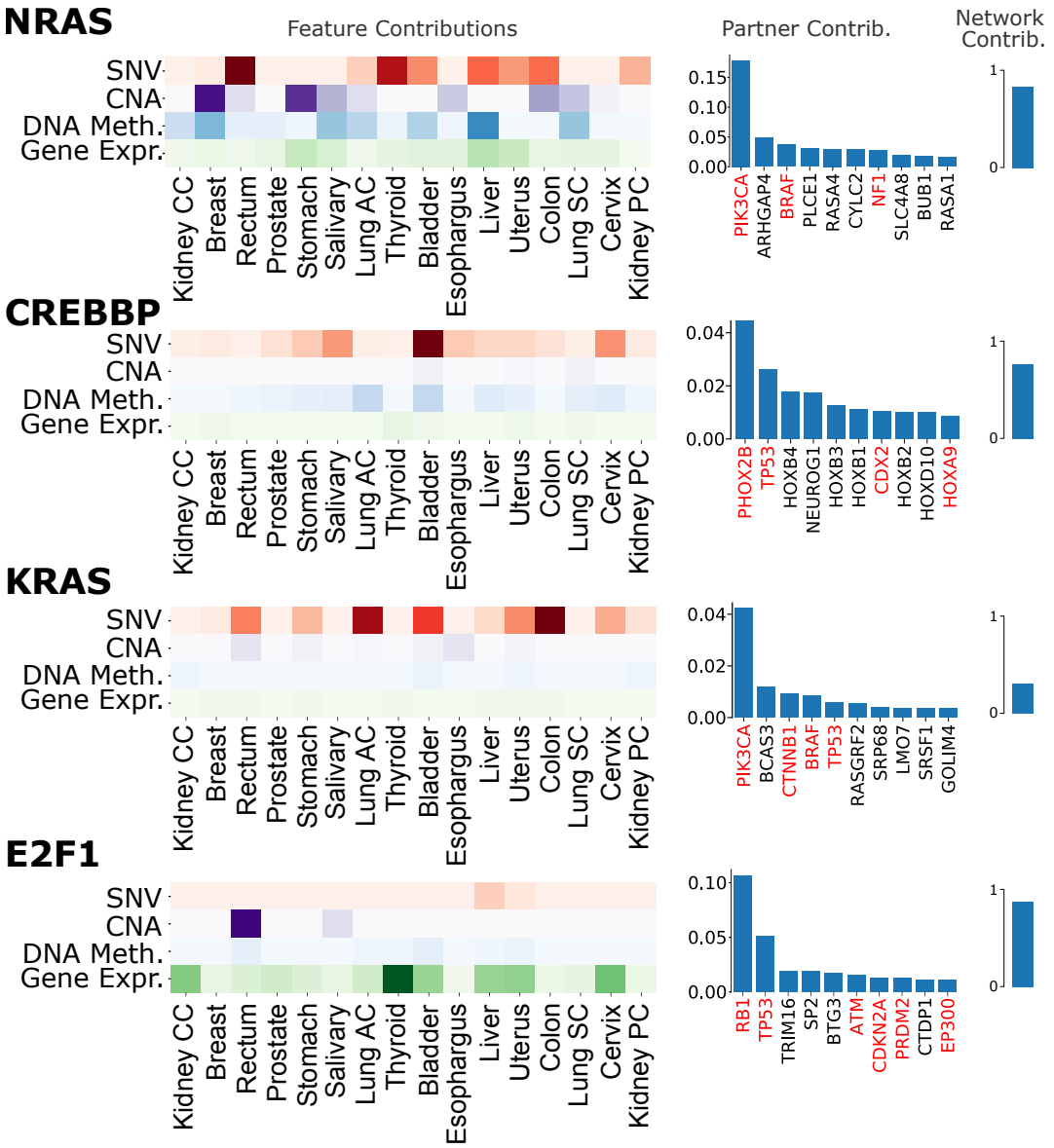


Figure S10: **EMOGI's LRP explanations for four representative genes.** We show the important *omics* features, as well as the most important PPI interaction partners computed through the LRP technique for four selected genes, namely *NRAS*, *CREBBP*, *KRAS* and *E2F1* (see section 2.5 for details). Heatmaps depict the importance of individual cancer types for different *omics* levels (darker colors indicate higher importance). The bar plots (middle column) depict the 10 most important interaction partners for the gene of interest. The scale of the bars denotes how much of the overall EMOGI score originates from the interaction with that gene. The right column depicts the amount that the interactome contributes to the classification of the gene of interest and ranges from 1 (only the PPIs was important) to 0 (only the features were important).

1.8 Relative Contributions of the different *Omics* to Classification

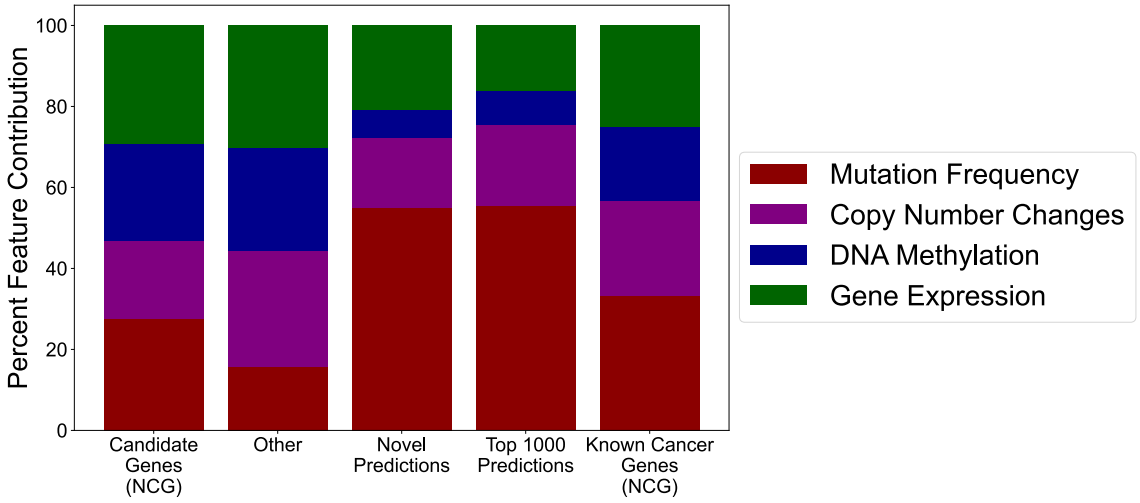


Figure S11: **Contribution of different *omics* data types to the classification.** We assessed the total contributions of single *omics* data types (mutation rates, copy number aberrations, DNA promoter methylation and gene expression) for classification of different gene sets. To compute the total contribution of each individual *omic* type for each gene set (computed as fraction of the total contribution from all *omics*) we summed up the LRP values across all genes in the set and applied min-max normalization to bring LRP values from different *omics* features on the same scale.

1.9 Newly Predicted Cancer Genes are classified as such because of their Interactions with Known Cancer Genes

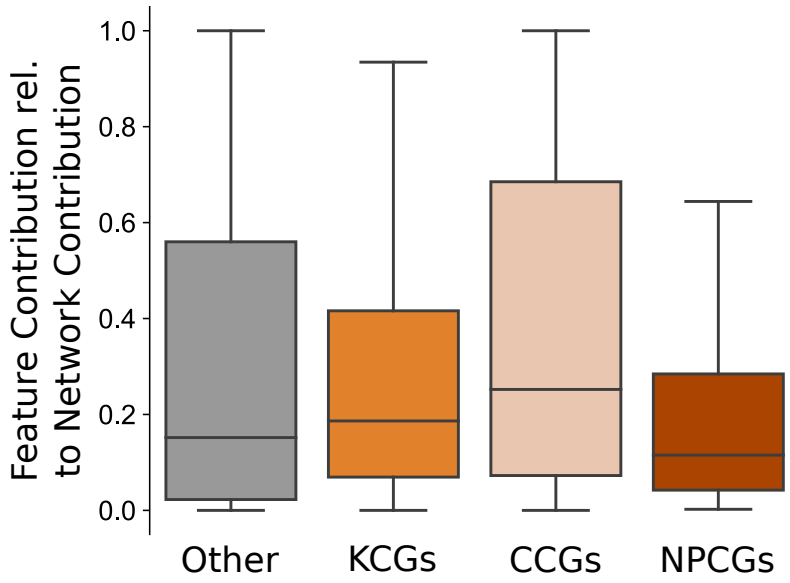
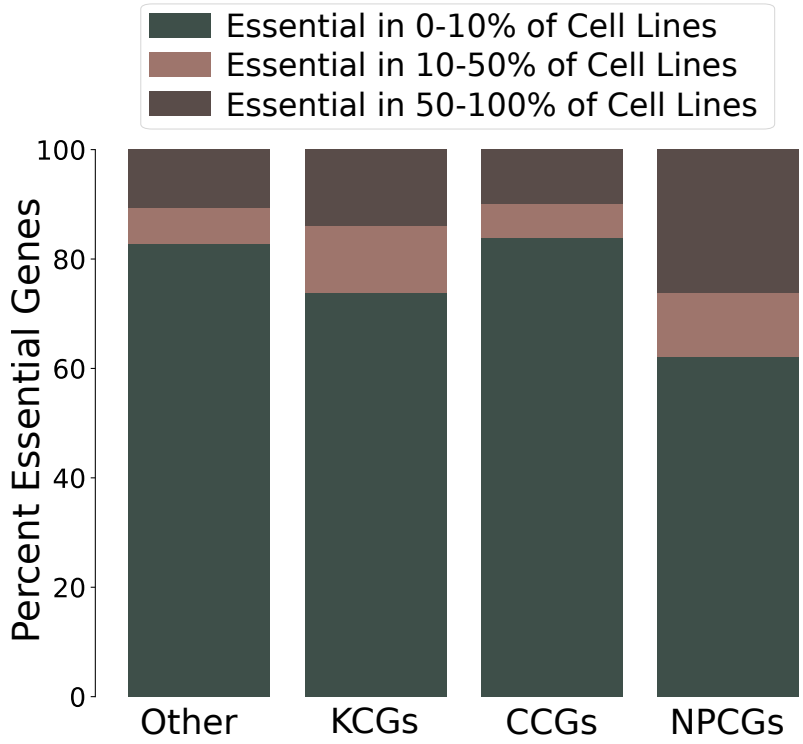


Figure S12: **Feature versus interactome contribution for different gene sets.** Shown is the average fraction of the contribution of the multi-omics versus the network features, computed through the LRP framework (section 2.5), for Known Cancer Genes (KCGs), Candidate cancer Genes (CCGs), NPCGs and Others, where this last group refers to genes which are not present in any of previous three sets.

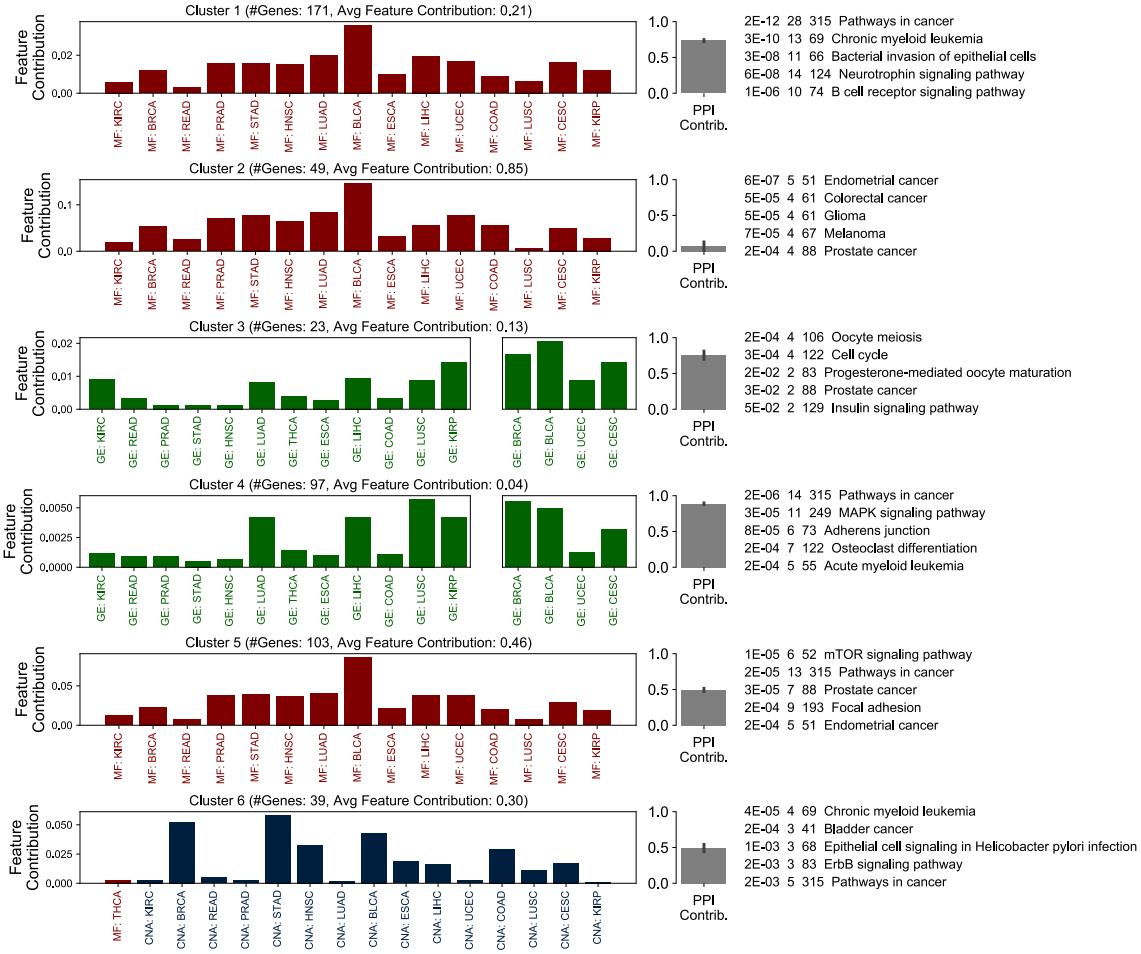
1.10 Newly Predicted Cancer Genes are not merely Housekeeping Genes



Term	P-Value	Count	Size
Cell cycle	$6.6e^{-12}$	18	122
Neurotrophin signaling	$9.1e^{-11}$	17	124
Shigellosis	$16.2e^{-10}$	12	59
Oocyte meiosis	$8.3e^{-10}$	15	106
Chronic myeloid leukemia	$5e^{-8}$	11	69
T cell receptor signaling	$7.2e^{-8}$	13	106
Pathways in cancer	$2.8e^{-7}$	21	315
Focal adhesion	$5.2e^{-7}$	16	193
GnRH signaling pathway	$1.1e^{-6}$	11	92
Osteoclast differentiation	$2.6e^{-6}$	12	122

Figure S13: **Scales of essentiality in tumor cell lines and pathway enrichment analysis.** Upper panel: shown is the proportion of genes essential in less than 10% of tumor cell lines, in more than 10% but less than 50% and in more than 50% based on the Achilles cancer dependency map [4] for the different gene sets. A CERES score ≤ -0.5 [5] was used to define essentiality of a gene in a cell line. On the bottom, the top 10 enriched KEGG pathways for the 165 NPCGs are shown (a complete list of enriched pathways can be found in table S3). In the table, *Count* denotes the number of NPCGs included in the pathway and *size* denotes the pathway size.

1.11 Selected bi-clusters for the top 1000 predicted cancer genes



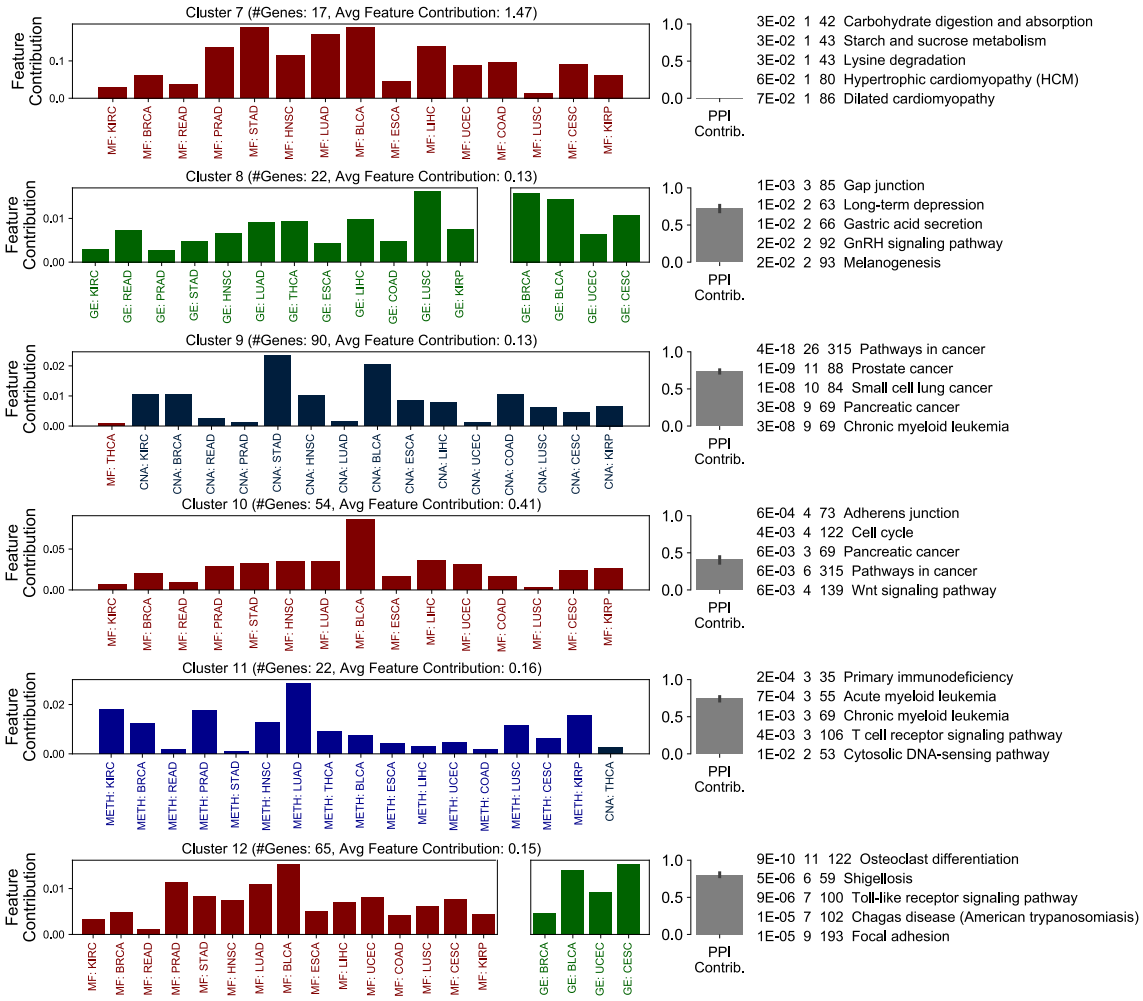


Figure S14: **Contributions of different features to the bi-clusters (continued from previous page)** Each plot panel corresponds to a different bi-cluster. Columns in each bar plot show average *omics* feature contributions to the classification of the genes in each cluster for the respective cancer and *omic* types. The middle column shows the overall importance of *omics* features relative to the network contribution for each cluster, and error bars denote the standard deviation across genes in the cluster. The right column shows enriched KEGG pathways in each cluster with corresponding p-values, number of cluster genes belonging to specific pathway and the total number of genes in the pathway. Split clusters (denoted with *a* and *b* in Fig. 5) are marked by a gap in the bar plot (e.g. Biclusters 3a and 3b).

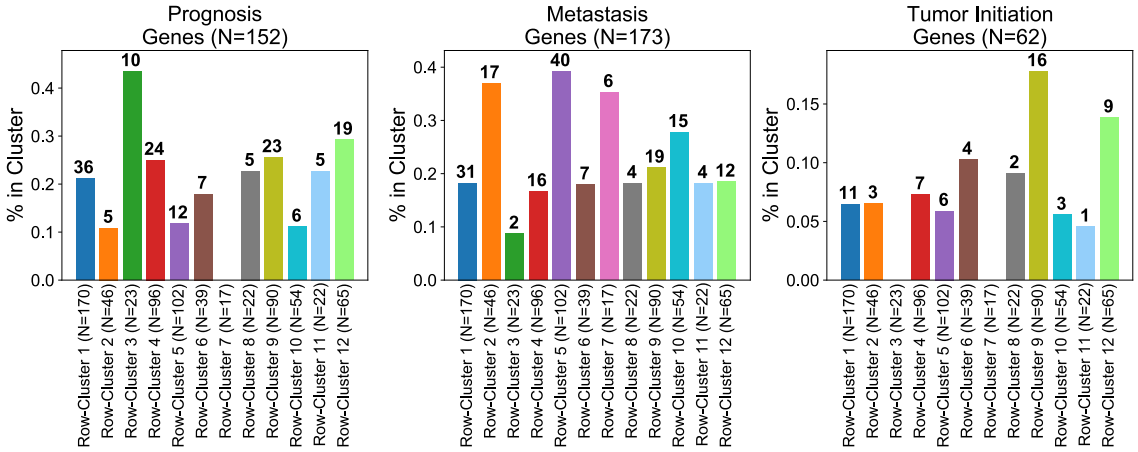


Figure S15: **Enrichment of different classes of cancer genes in the bi-clusters.** We extracted prognostic cancer genes from Wee et al. [6], genes involved in metastasis from Priestley et al. [7] and genes associated with tumor initiation from CIGene [8]. From the bi-clustering analysis in Fig. 5 (main article) we computed the percentage of prognostic, metastatic and tumor initiation-associated cancer genes contained in each bi-cluster, in order to assess whether any of those categories was over-represented in any of the bi-clusters. The numbers above the bars denote absolute numbers of prognostic, metastatic and tumor initiation genes in each bi-cluster, respectively. Colors match those in Fig. 5. Note that genes can be contained in multiple gene sets, e.g. be associated to both prognosis and metastasis.

1.12 Cutoff selection for interactions in Strongly Connected components (SCCs)

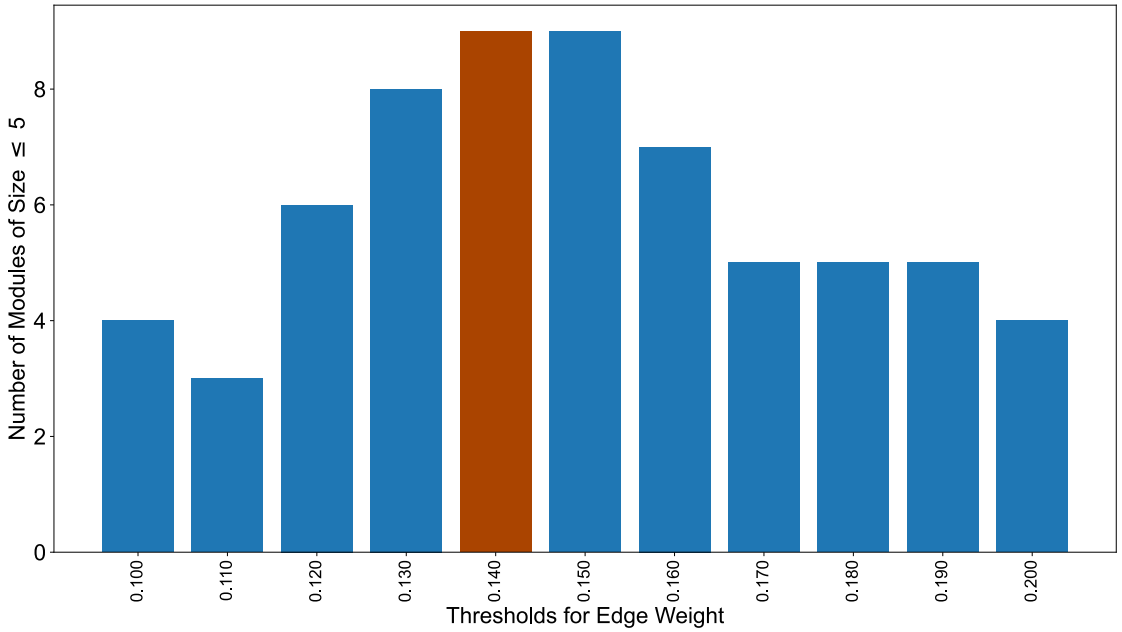


Figure S16: **Selection of a cutoff for module detection.** We examined the number of SCCs for different thresholds on the edge weights of the CPDB PPI network. The higher the edge weight, the higher the importance, computed as LRP value, of the interaction partner for the gene of interest in the network (see section 2.5 and Fig. S22, bottom right). For each putative threshold t, we discarded all edges in the LRP contribution graph with an edge weight $\leq t$. The resulting network corresponded to a subset of the original CPDB PPI network but with directed and weighted connections indicating whether a certain PPI was important for the EMOGI model. To detect SCCs, we used Tarjan's algorithm [9]. Here, we depict the number of SCCs with 5 or more nodes as function of the threshold t. We decided to use $t = 0.14$ because it yielded the most components with 5 or more genes.

1.13 Interactions of General Importance for EMOGI

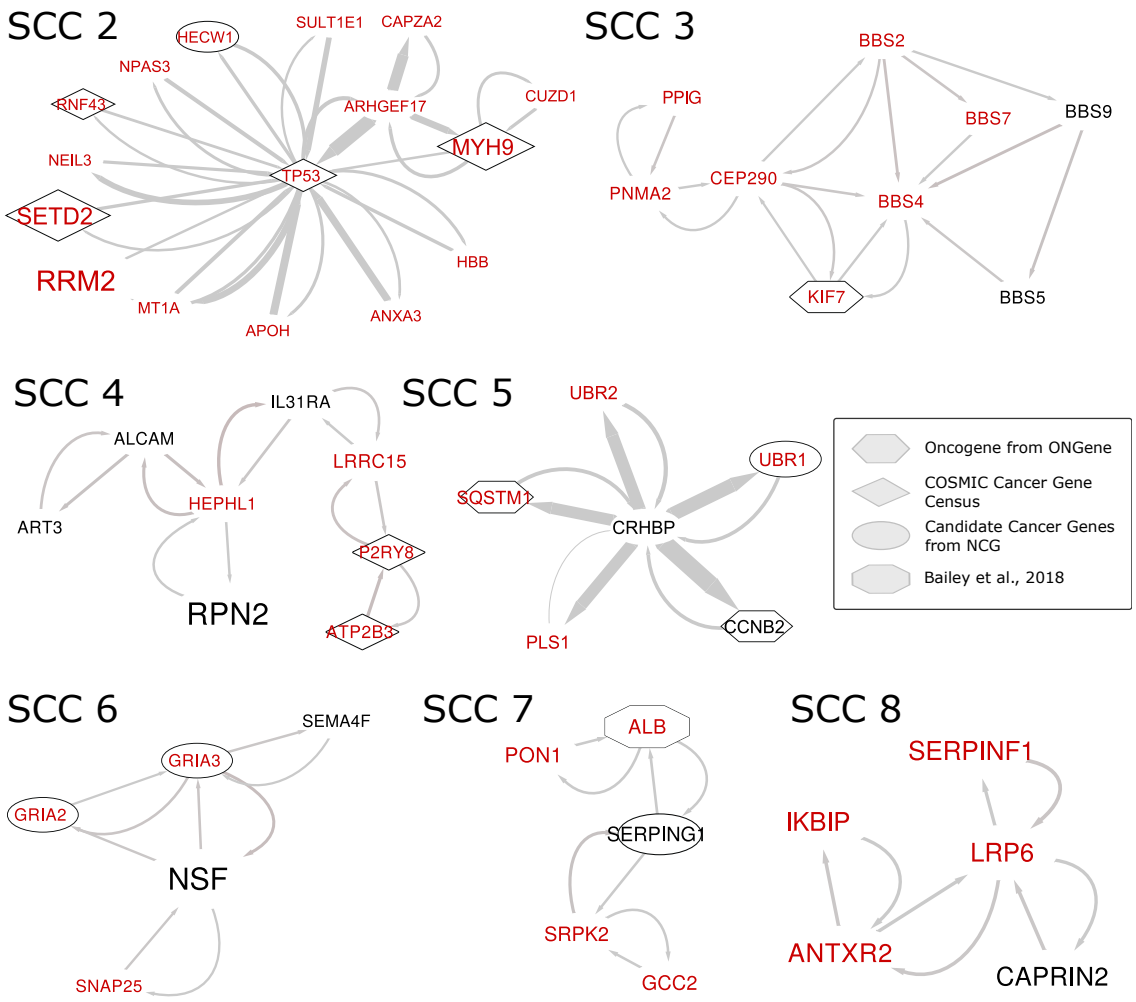


Figure S17: **Strongly Connected Components (SCCs) 2-8 identified from the LRP importance analysis**. Our approach identifies 8 SCCs which are depicted here and in the main text, figure 6. The size of the nodes is proportional to the number of tumor cell lines where the gene was found to be essential, according to the Achilles data. Gene names marked in red correspond to genes which were predicted to be cancer genes by EMOGI, and different node shapes indicate whether and in which database the gene was already annotated as cancer gene.

2 Materials & Methods

2.1 Graph Convolutional Networks

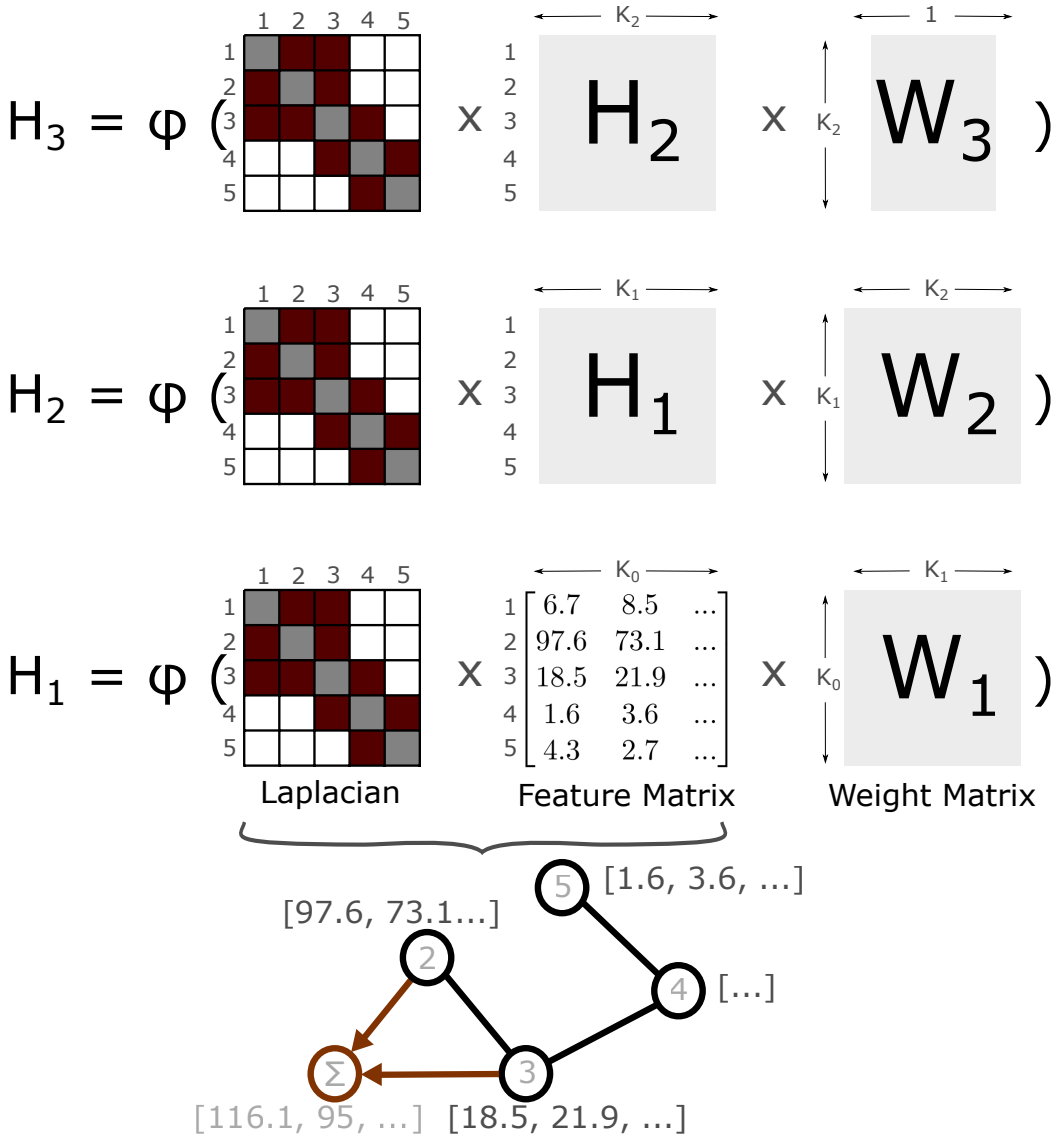


Figure S18: **Graph convolutions.** A graph convolutional network uses layers of graph convolutions. A simplified graph convolutional layer sums together the feature information of neighboring nodes. This is a form of label-smoothing for each of the features separately. Mathematically, the graph laplacian (which is very similar to its adjacency matrix) is multiplied with the feature matrix to do the smoothing. The resulting matrix is transformed through the weight matrix and a non-linear function, as in fully-connected neural networks. The output from one graph convolution can serve as input to the next layer or as output of the GCN. The output layer contains one output neuron and uses sigmoid activations while all other layers use ReLU activations.

Graph convolutional networks (GCNs) are an extension to classical convolutional neural networks and essentially aim to extend the convolution operation to non-regular grids that are unlike images where pixels are arranged in a specific structure. Convolution in images denotes a filtering operation where a filter (a small matrix, usually of size 3×3) is placed on an image and then "slides" over it, aggregating information from neighboring pixels. Filters have been used extensively in image recognition before the advent of deep learning, most often in the form of gaussian and Sobel filters. But they have been integrated in deep learning frameworks as convolutional layers and instead of defining the entries in a filter matrix, this matrix is learned by the algorithm.
For non-regular graphs with variable numbers of neighbors and no ordering among them, convolutions can not be applied in a straight-forward way [10]. However, spectral graph theory is a field in mathematics where graphs are described as frequencies and a Fourier transform of a graph exists, analogously to images. Therefore, convolutions in the spectral domain can be easily defined [11]. Several extensions and simplifications of the original spectral convolution definition [12, 13] have finally led to the GCN framework used in this paper [14].
A GCN transforms the graph it operates on to the spectral domain by graph Fourier transformation. The graph Fourier transformation is defined as:

$$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T \tag{1}$$

where $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix and D the degree matrix of the graph G. U therefore describes the matrix of eigenvectors of G and Λ the eigenvalues to the respective eigenvectors. GCNs then apply a filter g to a signal $x \in \mathbb{R}^N$. The graph convolution of g with x is written as:

$$g * x = U g U^T x \tag{2}$$

For large graphs such as PPI networks with more than $12,000$ nodes, the singular value decomposition is prohibitively expensive to do. Furthermore, filters of the graph convolution are still vectors of dimension N, making learning for large graphs hard when labels are scarce. Several approximations to graph convolutions have been proposed. Especially, Defferrard et al. [13] have proposed to approximate the filter vector g by a Chebychev polynomial $T_k(x)$. The degree of the polynomial k then controls the size of the filter because the polynomial is guaranteed to have a support $\leq k$ around the convoluted node.
A learning algorithm would then learn the coefficients of the Chebychev polynomial instead of the parameters in the Fourier space directly. This alleviates the problem of the locality of filters but does not fix the decomposition problem. Kipf and Welling [14] further simplified graph convolutions by setting $k = 1$, stating that a wider neighborhood (more than 1 hop away from the convoluted node) can be achieved through stacking multiple graph convolutions on top of each other. These (and other) simplifications lead to the definition of a graph convolutional layer:

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X W \tag{3}$$

with $\tilde{A} = A + I$ and $D_{ii} = \sum_j \tilde{A_{ij}}$ where $X \in \mathbb{R}^{N \times p}$ denotes the feature matrix for a p-dimensional signal, $W \in \mathbb{R}^{p \times m}$ denotes the parameters to learn and m corresponds to the number of filters in the graph convolution.

2.1.1 Graph Convolutions for Feature Tensors

We adapted EMOGI to perform gene classification for a specific cancer type at the level of individual patients, without averaging features across the samples of a given cancer type, but using the *omics* values of the single patient samples for training. We adapted the graph convolution operation defined in [14] (and described in the previous sub-section) to work with a feature tensor of rank 3. In detail, we constructed a feature tensor $F \in \mathbb{R}^{N \times S \times C}$ harbouring one dimension for

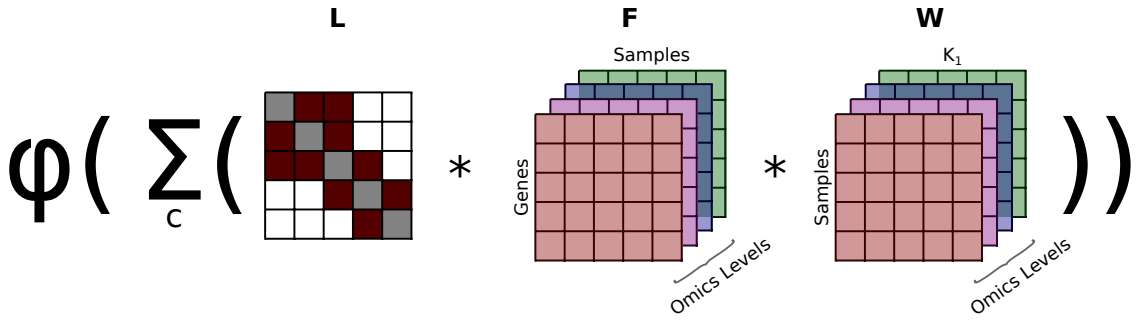


Figure S19: **Graph convolution operation for rank 3 tensors.** The adapted graph convolution operation takes one "slide" (corresponding to a single *omic* level) of the feature tensor F and the weight tensor W along with the graph laplacian L to perform a graph convolution (slides are indicated by different colors). This produces 4 activation maps (for SNVs in red, CNAs in purple, DNA methylation in blue and gene expression in green) that are summed together after the convolution. This procedure is analogous to how rbg channels of color images are convolved in CNN models for image processing.

the genes (N), one for the patient samples (S) and one for the *omics* levels (C), as summarized in Fig. S19, and treated the different *omics* levels similarly to the rgb channels of a color image processed by a CNN model. This led us to the following formulation of the convolution operation for the first model layer, leading to the activation map H^1:

$$H^1 = \sigma\big(\sum_{c=1}^{C} L * F_{::c} * W_{::c}^{(0)}\big) \tag{4}$$

where $W \in \mathbb{R}^{S \times H \times C}$ denotes the weight matrix to be learned (similarly to Equation 3 in the main method), H is the dimension (number of units) of layer 1 and L the graph laplacian. We sum over the activations of the separate *omics* levels $c \in 1, ..., C$ after convolving every *omic* separately with the graph laplacian L and the weight matrix W. This is analogous to what is usually done with the rgb channels of a color image. The extended graph convolution is depicted in Fig. S19. The produced activation map ($H^{(1)}$) has the same dimensionality as the activation map of the first layer defined in the original model (with a two-dimensional feature matrix as input, see subsection 2.1). The successive layers are hence not affected by the use of a rank 3 tensor and the previously defined graph convolution operation (equation 3 main method) holds for all subsequent layers after the first one.

2.2 Metrics

To evaluate the performance of our classifier and the other cancer gene prediction methods we computed the True Positives (TP) as the correctly identified known cancer genes, the False Positives (FP) as genes which are wrongly predicted as cancer drivers, the True Negatives (TN) as those genes which are correctly identified as non-cancer genes and the False Negatives (FN) as genes which are known cancer genes but which are missed by the algorithm. These values are then used to compute several performance metrics in order to compare EMOGI to other approaches. We mainly compute the area under the precision-recall curve (AUPRC) (figure 2 in the main article). The problem for many biological settings and this study especially is that the false positives are hard to evaluate. When new predictions are the goal of a classifier, we assume that the catalogue of

genes that associate with cancer is incomplete. This implies that we cannot assess false positives accurately beyond a test set because we do not know if a new prediction is valid or not. We evaluated EMOGI on a test set of known cancer genes and non-cancer genes in Fig. 2a, c and Fig. S4, S6 and S8.
For the independent cancer gene set evaluations (Fig. 2b and Fig. S5), however, we do not have an obvious choice of a negative set at hand. We still want to use AUPRC because it does not favor sensitive algorithms (i.e. algorithms that predict all genes to have associations with cancer). We therefore measured AUPRC values for the independent cancer gene sets (Fig. 2.b and Fig. S5) where every positive prediction which was not part of the respective gene set was considered a false positive prediction. This results in lower AUPRC values for all methods.

2.2.1 Area under the Precision-Recall Curve (AUPRC)

The area under the precision-recall curve is a performance metric for binary classifiers, independent from the chosen cutoff to classify examples in the positive or the negative class. In particular, when a classifier outputs a probability or any other continuous value, one can compute pairs of precision and recall for all possible cutoffs. This function of the cutoff can be plotted and is commonly referred to as the precision-recall curve (PR-curve). The best possible PR-curve goes from the upper left corner (perfect precision with no recall) straight to the upper right corner (perfect precision with perfect recall), and then it falls to the bottom right corner (perfect recall with no precision). Such an ideal PR-curve has an area under the curve of 1. A random classifier has a AUPRC given by the fraction of positives examples in the dataset.
PR-curves have also the advantage to account for class imbalance. In fact, a classifier that appears to perform well in the balanced setting might be nearly useless in the unbalanced setting, because it is very hard to control the number of false positives in such a situation. To circumvent this, PR-curves plot the fraction of predicted examples that are correct (precision) versus the fraction of all known cancer genes that are predicted as such (recall) at different cutoff scores.

2.3 Hyper-Parameter Optimization

The EMOGI algorithm contains several hyper-parameters that have to be set and which heavily influence the performance of the model. On top of the classical parameters of almost all deep learning models we also require a loss multiplier (a scalar factor by which the loss for known cancer genes is scaled to increase their importance). We employ a grid search over reasonable parameter ranges to optimize those hyper-parameters. Since our labeled data is rather scarce, we use 5-fold cross-validation (CV). Instead of using a validation set of labeled genes to quantify the performance of a combination of hyper-parameters, we split our training set in 5 parts. For every part, we split the training set into training and validation, using 20% of the labeled genes for validation as depicted in figure S20. We then selected the best-performing combination of hyper-parameters based on AUPRC values (see section 2.2.1 for details).

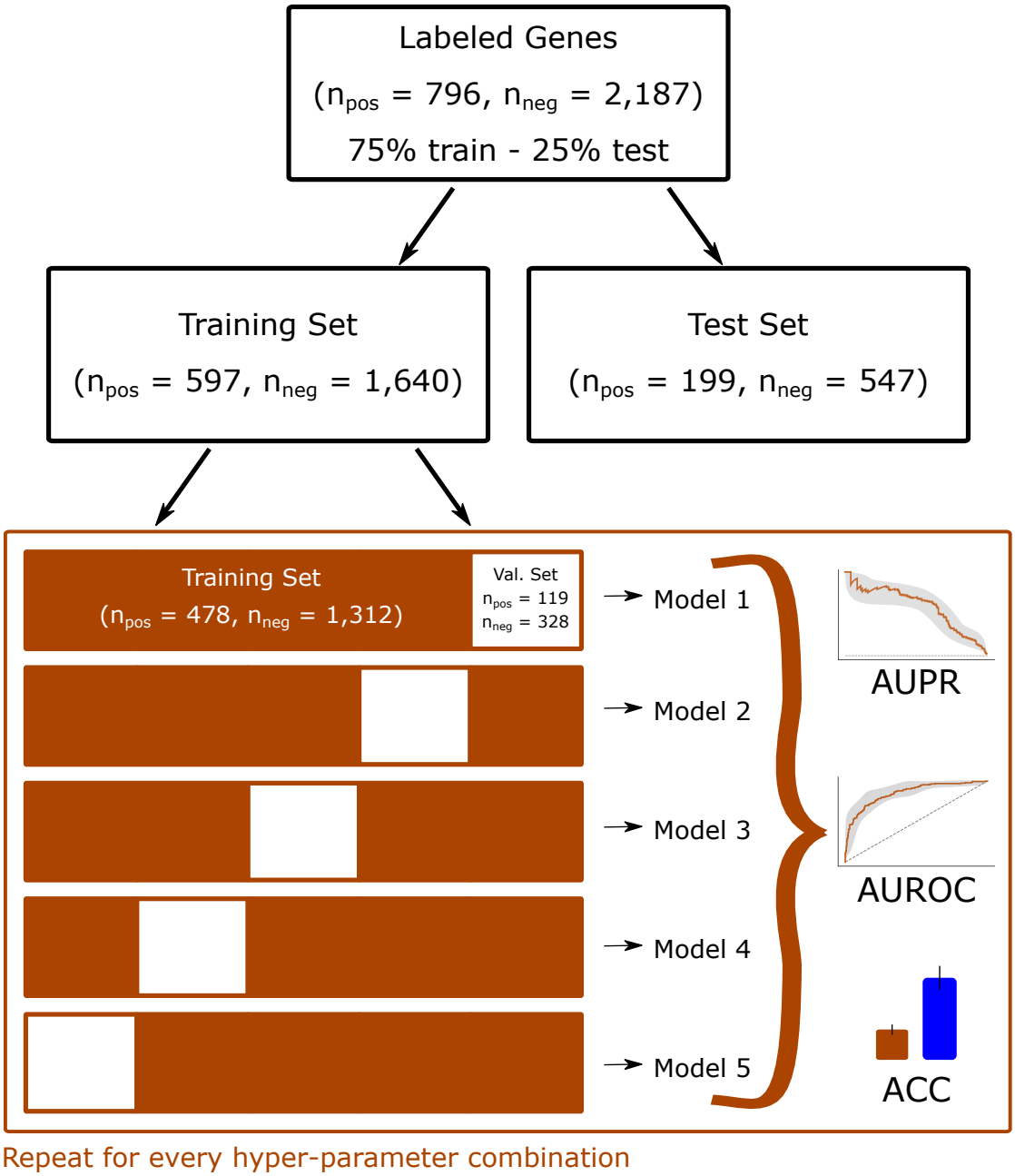


Figure S20: **Data splits for cross-validation and hyper-parameter optimization.** We split all of our labeled genes into training and test set, using 75% of the genes for training and the rest for testing. We then employed 5-fold CV on the training set to optimize hyper-parameters. That is, we used $\frac{1}{5}$ of the training set for validation and the other $\frac{4}{5}$ for training of an EMOGI model. We then repeated that procedure four more times such that every gene was used four times for training and once for validation. We then computed the performance of a hyper-parameter combination by averaging over the five models and selected the best combination as final parameters.

2.4 Perturbation Experiments

We conducted systematic perturbation of both node features and network interactions. For the features, we randomly selected two genes and exchanged their entire feature vectors. We repeated this process for either 25%, 50% or 75% and 100% of the nodes.
The network perturbations were generated using a double-edge swap. Here, two edges are randomly selected (e.g. u — v and x — y) and re-wired, such that two new edges u — x and v — y are obtained while the two original ones are lost. This procedure is also repeated until either 25%, 50%, 75% or 100% of the edges are swapped, while preserving the network's node degree [15]. We further considered a random network scenario, where we did not preserve node degree but preserved the characteristics of most biological networks by imposing a power law distribution on the node degree of the network. For that, we used the algorithm from Holme and Kim [16].

2.5 Layer-wise Relevance Propagation Applied to EMOGI to Extract Explanations for Genes

We used layer-wise relevance propagation (LRP) to compute feature importance for single genes (see main method). LRP is an attribution method for interpreting non-linear models, such as Neural Networks (NNs) or Convolutional Neural Networks (CNNs) exploiting the differential architecture of such models [17]. Unlike model-centric interpretation approaches, which give insights about the inner working of e.g. a CNN by explaining what each layer or network unit has learned, attribution methods aim at explaining why each input object has been classified in a certain way. Unlike linear classifiers, non-linear methods such as CNNs can have different explanations for different data points [17] and therefore, explanation methods centered on a particular data point of interest are very useful.
The essential idea of the LRP algorithm is to understand the contribution of a single component

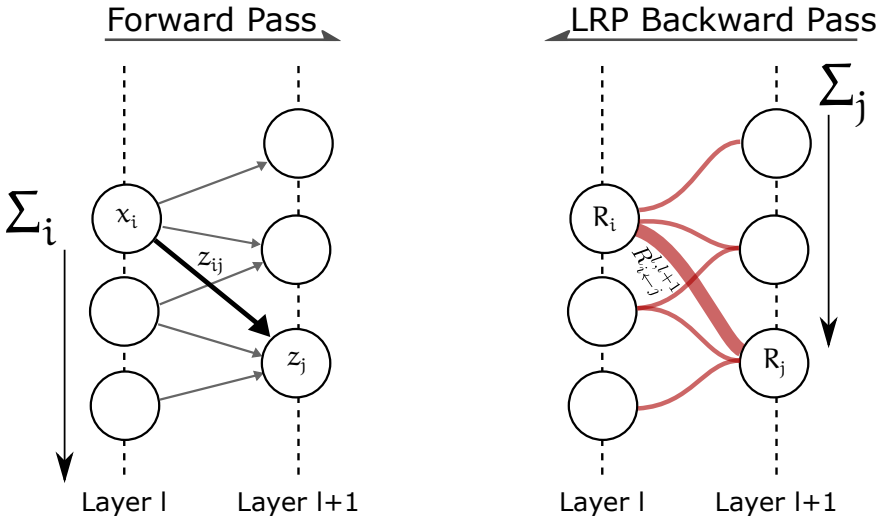


Figure S21: **Schematic workflow of layer-wise relevance propagation (LRP).** Figure reproduced from [18].

or feature of an input x to the model prediction $f(x)$ made by a classifier f for that particular example x. It is possible to decompose the function $f(x)$ into a sum of *relevance scores* R_d, each

one corresponding to a separate input dimension d.

$$f(x) \approx \sum_d R_d \tag{5}$$

The interpretation for relevance scores is that $R_d < 0$ contributes evidence against the structure that the classifier has learned, while $R_d > 0$ contributes evidence for the presence of such structure and $R_d \approx 0$ contributes no evidence to it.
Similarly to the backpropagation algorithm for Neural Networks (NNs), the idea is to redistribute the relevance scores onto the model units of the previous layer and compute intermediate relevance scores for each model unit. When this procedure is applied iteratively, i.e., from the output layer down to the input layer, LRP produces relevance scores for each input dimension or feature. For computer vision applications this results in a relevance score for each pixel in a input image. If we indicate the intermediate representation of the data at the $l-th$ layer as the vector $z = (z_i^l) \in \mathbb{R}^{dim(l)}$, LRP assumes that we have a relevance score R_j^{l+1} for each component z_j^{l+1} at layer $l+1$. The objective is to compute relevance score R_i^l for each component z_i^l at layer l closer to the input layer such that the total relevance contribution is conserved at each layer, i.e. the following equation holds:

$$\sum_d R_d^{input} = ... \sum_i R_i^l = \sum_j R_j^{l+1} = ... = f(x) \tag{6}$$

The relevance R_i^l attributed to each layer l and node i is expressed in terms of relevance messages $R_{i\leftarrow j}^{l,l+1}$ propagated from node j at layer $l+1$ to node i at layer l in a backward pass from output to input units, in contrast to the direction of the forward mapping $z_j^{l+1} = \sum_i z_{ij}^l$ typical of dense layers in ANNs or convolutional layers in CNNs where the intermediate representation of the data is achieved by summing up single contributions z_{ij} to the next layer (Figure S21, let panel). Relevance messages can be written as:

$$R_{i\leftarrow j}^{l,l+1} = \frac{z_{ij}}{z_j} R_j^{l+1} \tag{7}$$

The intuition beyond this formula is that if mapping z_{ij} contributes significantly to the activation of a node in the forward pass, then it will carry high relevance in the backward pass towards input nodes during the application of LRP. Finally, the relevance of node i at layer l can be written as the sum of incoming relevance messages from the direct unit contributions from next layer (Figure S21, right panel):

$$R_i^l = \sum_j R_{i\leftarrow j}^{l,l+1} = \sum_j \frac{z_{ij}}{z_j} R_j^{l+1} \tag{8}$$

To decompose the value $f(x)$ of the output node, the layer-wise propagation procedure in Equation 8 is applied iteratively, layer by layer, initializing it from the model output $R_K^L = f(x)$ with L indicating the last layer, until the input of the model is reached.

The exact propagation rule depends on the model architecture and function $f(x)$ to be learned. The LRP for a fully-connected neural network can be written as:

$$R_i^l = \sum_j \frac{a_i^l w_{ij}^l}{\sum_i a_i^l w_{ij}^l} R_j^{l+1} \tag{9}$$

where R_i and R_j represent the relevance of node i and j, respectively; $\sum_j$sums over the neurons in layer $l+1$ connected to neuron i; $\sum_i$ sums over the neurons in layer l connected to neuron j; a_i is the output of linear (non-linear) activation of neuron i in layer l and w_{ij} is the weight of the connection between neuron i and j.

When applying LRP to GCNs let us remember that each graph convolution layer does the following computation:

$$H^{(l+1)} = \sigma\big(LH^{(l)}W^{(l)}\big) \tag{10}$$

where $L = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ denotes the normalized graph laplacian, $\widetilde{A}$ the adjacency matrix with added self connections, $\widetilde{D}_{ii}$ the degree matrix, W a learnable weight matrix and σ a non-linear function, such as the ReLU activation function (see main method). Similarly to NNs, we can treat the process of LRP on a GCN as the inverse process of the forward reasoning through the GCN. In the forward reasoning, first, each node i calculates the weighted sum of all neighboring nodes' representations, and the weight for each neighboring node j equals the value located in the row i and column j of the normalized graph Laplacian L. Second, each node transforms it's representation through the multiplication with W. In the inverse process, one can compute relevance for the components of node feature vectors first, and then distribute the node's relevance vector computed in the first step to all its neighboring nodes. This is how it is done, for example in [3].

In this paper, we used the *deepexplain* package [19] to compute LRP for genes of interest. In the article corresponding to deepexplain and others, it was shown that the basic LRP propagation rule in Equation 9 can be expressed in terms of a modified gradient rule [19, 20, 18] when only ReLU activation functions are used, making LRP compatible with modern deep learning frameworks such as Tensorflow [21]. *Deepexplain* reformulates the LRP rule through partial derivatives with respect to the internal mappings H^l and computes the values using a propagation rule more similar to the back-propagation algorithm (see Kindermans et al. [20] or Ancona et al. [19] for the mathematical proof). Assuming top layer relevances $R^{(l+1)}$ corresponding to the nodes of $H^{(l+1)}$, the relevance for the nodes in $H^{(l)}$ can be computed as:

$$R^{(l)} = H^{(l)} \cdot \frac{\delta H^{(l+1)}}{\delta H^{(l)}} \cdot \frac{R^{(l+1)}}{H^{(l+1)}} \tag{11}$$

where $H^{(l)}$ denotes the intermediate representation learned by the GCN at layer l. Connecting LRP decomposition steps for consecutive layers yields:

$$R^{(Input)} = X \cdot \frac{\delta H^{(1)}}{\delta X} \cdot \cdots \cdot \frac{\delta H^{(L)}}{\delta H^{(L-1)}} = X \cdot \frac{\delta f(X)}{\delta X} \tag{12}$$

where $H^{(0)} = X$ corresponds to the input data vector or matrix X and $R^{(Input)}$ to the vector or matrix of relevance values for each input feature.

In its original application for image classification, LRP computes a value for each pixel of an input image, indicating the importance of that pixel for the classification. Therefore it returns for each input image a matrix, also called *relevance map*, of the same size of the original image, where each cell visualizes the relevance of a single pixel for classification [22, 19]. For the application of this paper, the input to EMOGI consists of two matrices, namely the graph laplacian L and the feature matrix X. Therefore, when we apply LRP to find the most relevant *omics* features and PPIs in the network for a gene of interest g, we obtain for that gene one matrix for the feature explanations, which we denote with $E^F(g)$ and one for the explanations of interaction partners, denoted with $E^I(g)$. These matrices have the same shape as the input matrices for features and network, namely $(N \times 64)$ and $(N \times N)$, as in our pan-cancer setting each node is characterized by 64 multi-omics feature values, and N is the number of genes (nodes) in the graph (Fig. S22, left panel).

Applying Equation 12 to our task we obtain:

$$E^F(g) = X \cdot \frac{\delta f(X, L)}{\delta X} \tag{13}$$

$$E^I(g) = L \cdot \frac{\delta f(X, L)}{\delta L}, \tag{14}$$

treating both matrices X and L as individual inputs to the GCN.
The intuition behind this is that GCNs aggregate features from neighboring nodes resulting in a setting where the classification of a gene is not solely based on the features of that gene but also the features of surrounding genes in the PPI network. Therefore, explanations for genes are also not only based on the gene of interest but also its surrounding genes in the network. Hence, both $E^F(g)$ and $E^I(g)$ are matrices instead of vectors as one would intuitively assume. In practise, the values of all rows (genes) in $E^F(g)$ and $E^I(g)$ are close to zero or very small except for the values corresponding to the gene of interest g and it can be easily seen from equation 14 that $E^I(g)$ is 0 for non-interacting gene products.

To answer the question about which cancer-type specific *omics* features mostly contribute to the classification of a certain gene, we extract the row of $E^F(g)$ that corresponds to the gene of interest, resulting in a vector with 64 entries, containing the importance of every *omic* feature in each cancer type (Fig. S22, left panel). The sum of the entries of this vector of feature explanations corresponds to the total multi-omics feature contribution for the gene of interest.
Similarly, to answer the question about the most important interaction partners for the classification of gene g we extract the row of $E^I(g)$ that corresponds to that gene (Fig. S22, left panel). This yields a vector with N entries, each denoting the importance of each gene in the network to the classification of the gene of interest.

For the bi-clustering analysis, we simply stack the vectors of feature explanations for all genes on top of each other to create a matrix of shape $(N \times 64)$, denoted as E^F_{total}, that we directly subject to the biclustering algorithm (Fig. S22, right panel).

The module analysis aims at finding core sub-networks which represent the most important parts of the interactome used by EMOGI for cancer gene classification. In order to understand how LRP values for gene interaction partners are used to identify such important modules we need to keep in mind that EMOGI uses a two-layered graph convolutional network which learns and propagates features from a first-order gene-gene interaction neighbourhood to a second-order neighbourhood. This means that higher order interactions, and not only direct interactions between genes, contribute to the classification.
To capture core interactions that are repeatedly important for the classification of multiple genes, and therefore identify cancer-related modules, it is not only sufficient to extract, for a given gene g, the row of $E^I(g)$, corresponding to the importance of its direct interactors, but also to incorporate higher-order neighborhood information from each gene $g_1.....g_N$. To achieve that, we collect the N interaction explanation matrices $E^I(g)$ returned by the LRP for each gene, each of dimension $N \times N$, and sum them up to obtain the matrix E^I_{total}, where the value of each cell represents the total importance of a certain PPI edge for the classification of the input genes (Figure S22, right panel). The higher this value, the higher the chance that the corresponding interaction will be part of a strongly connected component. We further apply min-max normalization to each row of E^I_{total}, such that every gene contributes the same total amount of LRP to the matrix, thereby removing the bias from genes with high node degrees in the PPI network. The final normalized E^I_{total} is then converted to a directed, weighted graph, where nodes correspond to the genes from the original PPI network and edge weights correspond to the entries of the normalized matrix E^I_{total}, i.e. to the importance of each interaction in the EMOGI model. We next selected a threshold and removed all edges with weights below that threshold to obtain a sparse graph, which is then subjected to the SCCs calling algorithm (see main method and Figure S16).

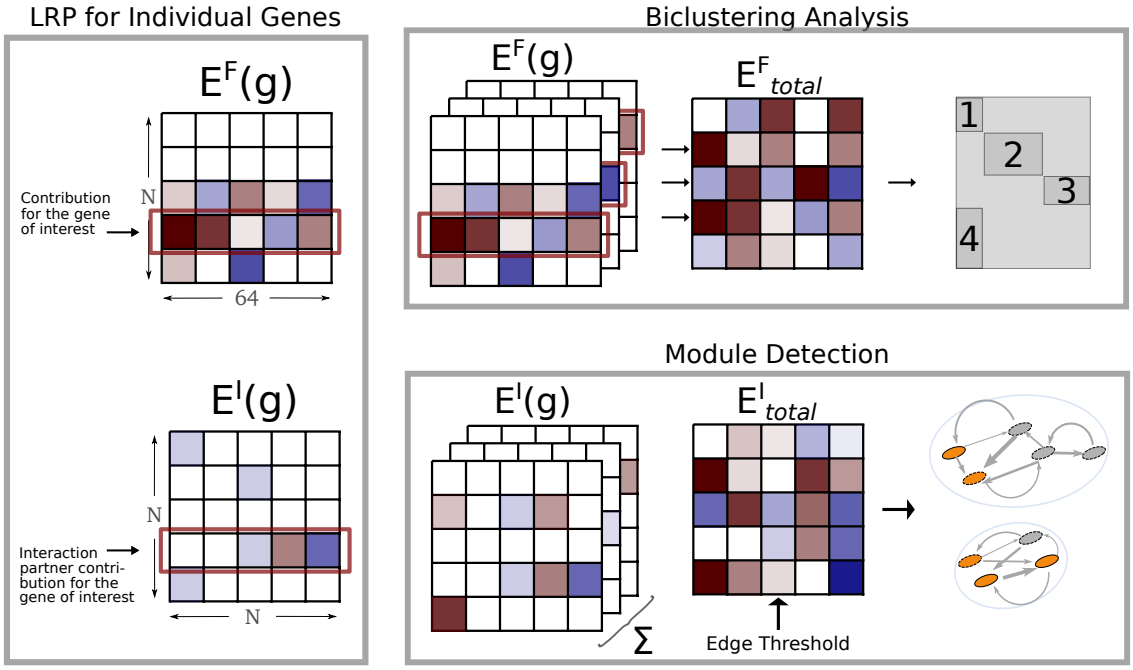


Figure S22: **Extraction of explanations using LRP.** Left panel: for each gene g LRP returns 1) a feature importance matrix $E^F(g)$ containing the *omics* feature contributions for that gene and 2) an interaction importance matrix $E^I(g)$ containing the PPI interaction contributions to the gene of interest. The right panel (top) summarizes how LRP and the obtained feature matrices for each gene $E^F(g)$ are summarized to obtain the E^F_{total} matrix which is subjected to bi-clustering analysis. The right panel (bottom) summarizes how LRP and the obtained interaction importance matrices for each gene $E^I(g)$ are aggregated to obtain the E^I_{total} matrix which is then used for module detection.

2.6 Other Prediction Methods used for Comparison

Random Forest model

A random forest classifier to predict cancer gene was trained on the multi-omics features only, namely genomic (from SNVs and CNAs), epigenomic and transcriptomic features, regardless of the PPI network using the *RandomForestClassifier* function with default parameters from the *scikit-learn* python package.

DeepWalk

This method [23] computes a node embedding in an unsupervised fashion from the network alone and does not incorporate node features or labels. DeepWalk constructs a vector representation of each node, trying to preserve the topological features of that node such that nodes with similar topologies are close to one another in the learned representation. It works by conducting small random walks on the graph iteratively, starting at each node in the network at a time. These walks explore the local structure around the node for which the representation is to be computed and are interpreted by DeepWalk as sentences of natural language. The algorithm then iterates over the random walks starting at node i and optimizes the node embedding to maximize the probability that the other nodes of the random walk are close to node i in the learned representation. This method, called skip-gram model [24], was very successfully used in natural language processing. We used DeepWalk to compute a vector representation of the nodes and then used a support-vector machine with radial basis function kernel on that representation to classify between cancer and non-cancer genes as was suggested by the authors of DeepWalk. We used our training and test set for training and evaluating the SVM that used the DeepWalk embeddings as input.

GCN Without Node Features

We used the same implementation of graph convolutional networks but replaced the feature matrix by a vector of 1s, as suggested by the authors [1]. To obtain reasonable hyper-parameters, we conducted a small grid search and found that the hyper-parameters used by the complete EMOGI's model worked well also in this case. We therefore trained EMOGI based on the PPI network's topology using an architecture of two graph convolutional layers with 300 units in the first layer and 100 units in th esecond layer, a dropout rate of 0.5, weight decay of 0.005, a learning rate of 0.001 and a cancer gene weight of 45.

PageRank

The PageRank algorithm, originally developed by Google to rank web pages in their search engine result, implements random walks on a network to identify highly influential nodes [25]. The PageRank algorithm outputs a probability distribution used to represent the likelihood a certain node in a network will be reached by starting randomly somewhere in the network. Here we used it to rank genes according to their likelihood of being cancer genes based on the PPI network, regardless of their *omics* features. We used the implementation from the *NetworkX* python package with a restart probability of 0.3. The node probabilities of the steady-state distribution were used to compute AUPRC values.

DeepWalk + Features

We also benchmarked EMOGI against a combination of DeepWalk embeddings alongside the multi-omics feature matrix computed for the 16 cancer types. For that, we concatenated the DeepWalk

[1] https://github.com/tkipf/gcn/issues/10

embeddings of the graph's nodes with the *omics* features for each gene and trained a Random Forest (RF) classifier on the combined matrix. We chose a RF because the *omics* values and DeepWalk embeddings are not likely to span the same feature space but RFs are known to work well with differently scaled data. We used the RF implementation from the *scikit-learn* python package with default parameters.

HotNet2 Diffusion

Network propagation amplifies a biological signal based on the assumption that neighbouring genes in an interaction network share the same phenotype [26]. Network propagation forms the basis of HotNet2 [27], where an average mutation score across cancer types is associated to every node in the PPI network and propagated to nearby nodes in an iterative manner using random walk with restart, similarly to PageRank. We used the HotNet2 implementation of the random walk with restart and their precomputed heat scores for the genes to obtain the steady-state probability distribution to compute AUPRC values.

MutSigCV

MutSigCV is one of the most popular tools to identify cancer driver genes [28]. It prioritizes highly mutated genes, i.e. genes with a mutation frequency that is higher than what is expected by chance, based on a background model which takes into account both gene sequence composition and gene length. We used MutSigCV to predict cancer genes from one type of *omics* only, the mutation rate. We computed gene-associated q-values for each of the TCGA samples separately and averaged over all 16 samples to produce gene scores. The $-\log_{10}$ was used to compute the AUPRC values.

20/20+

This method is a widely used machine learning method that predicts cancer genes based on mutations. Tailored specifically to the problem of cancer gene prediction, 20/20+ uses a random forest alongside a ratiometric approach to predict oncogenes and tumor suppressor genes [29].
We applied the tool to the MAF files for all 16 cancer types used in this study obtained from TCGA. Specifically, we concatenated all MAF files using custom code and performed a liftover of the mutations from hg38 back to the hg19 reference genome because 20/20+ is not compatible with the newer genome version. We then used 20/20+ as indicated in the documentation for pan-cancer applications with our concatenated MAF file [2] and extracted the "driver score" for all genes.

2.7 Gene sets used in this study

2.7.1 Gene sets used for training

- Network of Cancer Genes (NCG) v6.0
  (`http://ncg.kcl.ac.uk/download_file.php?file=cancergenes_list.txt`)
- DigSee database
  (`http://210.107.182.61/digseeOld/`)
- COSMIC Cancer Gene Census (CGC) v91 and COSMIC Mutations in Census Genes
  (`https://cancer.sanger.ac.uk/cosmic/download`)

[2] https://2020plus.readthedocs.io/en/latest/tutorial.html#pan-cancer-analysis

2.7.2 Independent Cancer Gene Sets

We collected four additional cancer gene sets from other sources to construct independent test sets for benchmarking of the different methods.

- The first set is represented by candidate cancer genes from the NCG which are non-overlapping with the known cancer gene set used for EMOGI's training.
- The second set comprises a list of 682 genes from the OncoKB database, a manually curated dataset of cancer genes annotated according to validated oncogenic effects. Only high-confidence cancer genes, i.e. with evidence from more than three sources including clinical studies and associated FDA-approved drug, were included in this set (`https://www.oncokb.org/cancerGenes`).
- The third set comprises a list of literature-curated cancer genes from the ONGene database [30] (`http://ongene.bioinfo-minzhao.org/ongene_human.txt`).
- The fourth set a list of high-confidence cancer genes compiled using different computational tools [2] (`https://www.cell.com/cms/10.1016/j.cell.2018.02.060/attachment/cf6b14b1-6af1-46c3-a2c2-91008c78e87f/mmc1.xlsx` Table S1).

Overlap with training and test sets was removed for all four datasets to ensure an unbiased analysis. In addition, DriverDB [3] gene set were downloaded at: `http://driverdb.tms.cmu.edu.tw/cancer`.

2.7.3 Achilles Project data

The Achilles Project is a systematic effort aimed at identifying and cataloging gene essentiality across 625 genetically characterized cancer cell lines [4]. It uses genome-wide RNAi and CRISPR-Cas9 genetic perturbation to silence or knock-down individual genes and compile a dependency map of *essential* genes, i.e. a map of genes which significantly affect cell survival in a cell-type specific way. In detail, each gene is assigned a so-called *CERES* score in each cell line which summarizes the effect of that gene on cell proliferation from the loss-of-function screen [5]. Negative *CERES* scores indicate a reduced cell growth, while positive scores indicate an increased proliferation. Essential genes in a tumor cell line where defined as those genes with $CERES < -0.5$. We extended this definition to define globally essential genes as those genes with a *CERES* score < -0.5 in more than 78 cell lines, where 78 denotes the average amount of cell lines affected by a gene. The Achilles data was used for functional validation of EMOGI's predictions and can be downloaded at: `https://ndownloader.figshare.com/files/22629068`.

2.7.4 Gene sets for different cancer stages

To check for the enrichment of genes involved in different cancer stages in the specific bi-clusters (Fig. S15) the following data sets were downloaded

- Cancer initiation genes from CIGene [8] (`http://soft.bioinfo-minzhao.org/cigene/`)
- Metastasis genes from Priestley et al. [7] (`https://static-content.springer.com/esm/art%3A10.1038%2Fs41586-019-1689-y/MediaObjects/41586_2019_1689_MOESM10_ESM.xlsx`)
- Prognostic genes from Wee et al. [6] (`https://static-content.springer.com/esm/art%3A10.1038%2Fs41598-018-21691-5/MediaObjects/41598_2018_21691_MOESM3_ESM.xlsx`)

Supplementary Tables

Supplementary Table 1 - Overview of TCGA studies and data availability.

This table provides an overview of TCGA studies for 33 cancer types. The first column contains the TCGA study abbreviations, followed by the cancer type in the next column. For mutation profiling, DNA methylation and gene expression, we collected the amount of data available. These numbers can be slightly different today as the data was collected in early 2019. For DNA methylation, we count the samples of tumor and normal adjacent tissue, for mutation data we count the samples in the TCGA repository and additionally check whether there is also a pre-processed data set on synapse. For gene expression, we count tumor samples, normal samples from the same tissue and GTEx samples from the same tissue. The last column summarizes if we can use that cancer type for our EMOGI pan-cancer analysis. In total, we use 16 different cancer types.

Supplementary Table 2 - EMOGI predictions for the CPDB PPI network.

This table contains Ensembl gene IDs in the first column, the official gene symbol and whether the gene was associated with cancer previously (label column). It then states the probability of being a cancer gene across the 10 cross-validation models (called *Prob_pos_i*), the number of times that the gene has received a score ≥ 0.5 (*Num_Pos*) and the mean EMOGI score (*Mean_Pred*) as well as the standard deviation for the predictions (*Std_Pred*).

Supplementary Table 3 - List of NPCGs and KEGG pathways enriched.

The first sheet contains a list of the 165 newly predicted cancer genes (NPCGs). For every gene, we list how often it was present in the top 100 predictions of one of the 6 EMOGI models for different PPI networks (averaged across the 10 cross validations). We further list the presence of the gene in either the network of cancer genes (NCG) candidate cancer genes or the OncoKB data base high confidence list (see methods for details). We further report the number of tumor cell lines for which this gene had a significant negative effect on culture growth. The second sheet lists enriched pathways in the NPCGs according to the KEGG data base. It contains all enriched pathways with a p-value ≤ 0.05.

Supplementary Table 4 - List of genes in the 12 Biclusters.

We performed spectral biclustering of the feature contributions for the top 1000 genes of EMOGI trained on the CPDB PPI network. We selected 12 biclusters (see main article, figure 5). The table lists the genes contained in all of the 12 biclusters in different sheets with their Ensembl gene IDs and official gene symbols.

Supplementary Table 5 - GO enrichment for Biclusters.

Similarly to the previous table, this table contains sheets for the same 12 biclusters. It provides a Gene Ontology enrichment analysis (GOEA) for the biclusters. The analysis contains also p-values corrected for multiple tests (*p_fdr_bh*) and lists enriched terms for biological processes (BP), molecular functions (MF) and cellular components (CC).

Supplementary Table 6 - LRP for top 1000 genes.

This table lists the feature contributions for the top 1000 genes predicted by the CPDB EMOGI model. For every gene (identified by the official gene symbol), the contributions of gene expression, DNA methylation, copy number changes and mutations are summarized such that the overall

feature contribution (sum of the four columns) adds to 1. Furthermore, the table lists the contribution of the PPI network in comparison to the features. Note that the value can be greater than 1 when the feature contribution is negative. This indicates that the multi-omics gene features are evidence against the gene being a cancer gene. Lastly, the top 5 interaction partners according to the LRP of every gene are listed as well.

Supplementary Table 7 - List of genes in SCCs.

This table lists the genes present in strongly-connected components with a size ≥ 5 in the CPDB PPI network. The LRP contributions of the interactome form a directed version of the PPI network where the edge weight corresponds to the importance of the PPI to the final classification. We removed edges with weight > 0.14 and detected strongly-connected components in the network. There are 8 components with size ≥ 5. The table lists Ensembl gene IDs and gene symbols in a sheet per component.

Supplementary Table 8 - GO enrichment for SCCs.

Similarly to the previous table, this table contains a GO enrichment analysis for the SCCs. We only list SCCs with a significant corrected p-value, therefore the table contains only 5 sheets for the largest, third-largest, fifth, sixth and seventh -largest components.

Supplementary Table 9 - KEGG pathway enrichment analysis for SCCs.

Similarly to the previous table, this table contains a KEGG pathway enrichment analysis for the SCCs. We only list SCCs with a significant p-value, therefore the table contains only 7 sheets.

References

[1] Repana, D. *et al.* The Network of Cancer Genes (NCG): a comprehensive catalogue of known and candidate cancer genes from cancer sequencing screens. *Genome Biology* **20**, 1–12 (2019).

[2] Bailey, M. H. *et al.* Comprehensive Characterization of Cancer Driver Genes and Mutations. *Cell* **173**, 371–385.e18 (2018).

[3] Liu, S. H. *et al.* DriverDBv3: A multi-omics database for cancer driver gene research. *Nucleic Acids Research* **48**, D863–D870 (2020).

[4] Tsherniak, A. *et al.* Defining a Cancer Dependency Map. *Cell* **170**, 564–576.e16 (2017).

[5] Meyers, R. M. *et al.* Computational correction of copy number effect improves specificity of CRISPR-Cas9 essentiality screens in cancer cells. *Nature Genetics* **49**, 1779–1784 (2017).

[6] Wee, Y., Liu, Y., Lu, J., Li, X. & Zhao, M. Identification of novel prognosis-related genes associated with cancer using integrative network analysis. *Scientific Reports* **8** (2018).

[7] Priestley, P. *et al.* Pan-cancer whole-genome analyses of metastatic solid tumours. *Nature* **575**, 210–216 (2019).

[8] Liu, Y. *et al.* CIGene: A literature-based online resource for cancer initiation genes. *BMC Genomics* **19** (2018).

[9] Tarjan, R. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing* **1**, 146–160 (1972).

[10] Niepert, M., Ahmed, M. & Kutzkov, K. Learning Convolutional Neural Networks for Graphs. In *ICML* (2016).

[11] Hammond, D. K. & Vandergheynst, P. Wavelets on Graphs via Spectral Graph Theory 1–37 (2009).

[12] Bruna, J., Zaremba, W., Szlam, A. & LeCun, Y. Spectral Networks and Locally Connected Networks on Graphs. *ICLR* 14 (2013).

[13] Defferrard, M., Bresson, X. & Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *NIPS* 1–14 (2016).

[14] Kipf, T. N. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *ICLR 2017* 1–10 (2016).

[15] Frieze, A., Horn, P. & Pralat, P. *Algorithms and Models for the Web-Graph: 8th International Workshop, WAW 2011, Atlanta, GA, USA, May 27-29, 2011, Proceedings*. Lecture Notes in Computer Science (Springer Berlin Heidelberg, 2011).

[16] Holme, P. & Kim, B. J. Growing scale-free networks with tunable clustering. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics* **65**, 026107 (2002).

[17] Lapuschkin, S. *et al.* Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications* **10**, 1096 (2019).

[18] Lapuschkin, S. *Opening the Machine Learning Black Box with Layer-wise Relevance Propagation*. Ph.D. thesis (2019).

[19] Ancona, M., Ceolini, E., Öztireli, C. & Gross, M. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* (International Conference on Learning Representations, ICLR, 2018).

[20] Kindermans, P.-J., Schütt, K., Müller, K.-R. & Dähne, S. Investigating the influence of noise and distractors on the interpretation of neural networks (2016).

[21] Martin, A., Paul, B., Jianmin, C. & Zhifeng, C. TensorFlow: A system for Large-Scale Machine Learning 265–283 (2016).

[22] Bach, S. *et al.* On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* **10**, 1–46 (2015).

[23] Perozzi, B., Al-Rfou, R. & Skiena, S. DeepWalk: Online Learning of Social Representations. *arXiv* 1–10 (2014).

[24] Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings* (International Conference on Learning Representations, ICLR, 2013).

[25] Page, L., Brin, S., Motwani, R. & Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. *World Wide Web Internet And Web Information Systems* 1–17 (1998).

[26] Cowen, L., Ideker, T., Raphael, B. J. & Sharan, R. Network propagation: A universal amplifier of genetic associations. *Nature Reviews Genetics* **18**, 551–562 (2017).

[27] Leiserson, M. D. *et al.* Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nature Genetics* **47**, 106–114 (2015).

[28] Lawrence, M. S. *et al.* Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature* **499**, 214–218 (2013).

[29] Tokheim, C. J., Papadopoulos, N., Kinzler, K. W., Vogelstein, B. & Karchin, R. Evaluating the evaluation of cancer driver genes. *Proceedings of the National Academy of Sciences of the United States of America* **113**, 14330–14335 (2016).

[30] Liu, Y., Sun, J. & Zhao, M. ONGene: A literature-based database for human oncogenes. *Journal of Genetics and Genomics* **44**, 119–121 (2017).