University of Cambridge

MPhil in Data Intensive Science

DiS MPhil Project 19

Project Report for 19 Pathomic Fusion: an interpretable attention-based framework that integrates genomics and imaging to predict cancer outcomes

Shizhe Xu
Email: sx263@cam.ac.uk
Submission Deadline: 30 June 2024
Total Word Count: 6901

Contents

1 Introduction

Word Count: 560

The mechanisms behind cancer growth are still not fully understood because the tumour microenvironment is a complex network of various interactions between immune cells and non-cellular components [5]. Similar findings in one modality can have diverse interpretations in combination with other modalities. For example, the importance of genomic features might change when conditioning morphological features [20]. However, most existing histology analysis strategies are not automatically aligned with genomic modality and manual adjustments are always time-consuming [11]. These strategies do not explicitly incorporate information from cell graphs, which better capture the proliferation and community structure of tumour cells [31]. To improve the efficiency and accuracy of cancer prediction, there is a pressing demand to develop a new model that extracts and combines information across different modalities in an integrative manner. Using both phenotypic and genotypic information [9], multimodal networks can identify features predictive of patient survival, which could lead to new discoveries in biomarkers.

- **Networks and Fusion Mechanism**: We explore each unimodal network (CNN, GCN, SNN) in detail and conduct a comprehensive review of how to generate feature representations, which are used as inputs for multimodal fusion model. We also explain the mechanism behind the Kronecker product and gating-based attention algorithms, which capture interactions across unimodal feature representations and and mitigate the data heterogeneity gap problem.

- **Environment Setup**: We briefly mention the packages used in our new conda environment and the choice of CUDA and cuDNN. It also includes short examples of updating the original code to make it compatible with the latest PyTorch packages.

- **Data Acquisition**: We describe how to obtain the histology images and genomic data from glioma and clear cell renal cell carcinoma datasets. The old cell graph .pt files are also mentioned due to node index problems and version issues. The issues of missing data and alignment are discussed in this section.

- **Model Implementation and Training**: We emphasise the importance of making data splits for our 15-fold cross validation. We discuss the training process of multimodal models with our own data splits and original splits from the paper. We also introduce loss functions, baseline models and training settings for GBMLGG and KIRC.

- **Validation and Testing**: We check and compare the results from the test splits of the 15-fold cross-validation. In addition, we have successfully reproduced results very similar to those in the original paper. This section also includes evaluation plots for survival analysis and grade classification of GBMLGG and KIRC.

- **Data Visualisation**: We develop our own visualisation code by using Grad-CAM and integrated gradients, and analyse feature importance and localisation across each modality. Moreover, we demonstrate the global explanation of genomic features for each histomolecular subtype across the entire patient cohort in both GBMLGG and KIRC.

- **Further Improvements and Discussion**: We discuss the potential improvements for the pathomic fusion model. The feature encoder for histology images can be replaced by vgg19 from 'torchvi-

sion.models'. The CONCH image encoder can also be applied to the histology images, allowing new feature representations to be fused with those from CNN, GCN, and SNN. We find the multilinear/quadrilinear fusion (CNN+GCN+SNN+CONCH) outperforms the original fusion (CNN+GCN+SNN). The nuclei segmentation tool can be replaced by CellViT, and alternative algorithms for Grad-CAM are also introduced. The novel network KAN is applied to genomic features for comparison purposes. Note that MLPs are based on the universal approximation theorem, while KANs are based on the Kolmogorov-Arnold representation theorem.

2 Networks and Fusion Mechanism

Word Count: 1527

The standard method for most fusion models [26] is to extract features from histology images and then integrate these image features with genomic features. However, we want to develop a multimodal fusion model that integrates histology image, cell graph, and genomic features simultaneously. After obtaining unimodal representations, we use the Kronecker product to fuse them into a multimodal tensor to capture all possible interactions across different modalities. During the fusion process, the gating-based attention mechanism is implemented to control the expressiveness of each feature and regularise unimportant features. Note that patch averaging of predicted hazards is not performed on GCNs since cell graphs are reconstructed from the whole slide.

2.1 Self-Normalising Networks for Genomic Features from Molecular Profiles

The modern sequencing technologies such as single cell sequencing and next generation sequencing (NGS) allow oncologists to extract genomic information from tumour tissues and perform analysis with spatial transcriptomics and multiplexed immunofluorescence [1], [29]. Genomic information could be CNV, mutation status, RNA-Seq expression, etc.

There are only hundreds of training samples/patients, but each of them contains thousands of genomic features. It is inappropriate to implement feedforward networks that are susceptible to overfitting [28]. Feedforward networks are also very sensitive to training instabilities [4]. These instabilities are mainly caused by dropout and gradient descent. Given the small sample size with high-dimensional sample data, we modify feedforward networks and call the new network "MaxNet" by implementing the normalisation layers from Self-Normalising Networks (SNNs) in [16]. With the normalisation layers, the training loss has been dramatically reduced in Figure 1.

Definition 1. *(self-normalising layers) A neural network is self-normalising if it possesses a mapping* $g : \Omega \mapsto \Omega$ *for each activation* y *that maps the mean and variance from one layer to the next and has a stable and attracting fixed point depending on* (ω, τ) *in* Ω*. In other words, the mean and variance remain in the same domain* ω *(*$g(\Omega) \subseteq \Omega$*), where* $\Omega = \{(\mu, \nu) \mid \mu \in [\mu_{\min}, \mu_{\max}], \nu \in [\nu_{\min}, \nu_{\max}]\}$.

We modify the layer in the original feedforward networks to ensure each point within the same domain converges to a fixed point. We also change the rectified linear unit (ReLU) activation into the scaled exponential linear unit (SeLU) activation, which can push the mean and variance output of every layer toward the fixed point 0. In our case, we take the special case of SeLU by setting λ to 1, making the activation function ELU in Definition 2.

Definition 2. $SeLU(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha\,(e^x - 1) & \text{if } x \leq 0 \end{cases}$; $\quad ELU(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(e^x - 1) & \text{if } x \leq 0 \end{cases}$.

Our final SNN architecture consists of four sequentially connected encoder layers, each containing a linear transformation, ELU activation, and Alpha Dropout. Alpha Dropout maintains the mean and variance to their original values, ensuring the self-normalsing property still holds. The last fully-connected layer/classifier learns a genomic representation $\mathbf{h}_n \in \mathbf{R}^{32\times 1}$, which is used as an input for our multimodal fusion model.

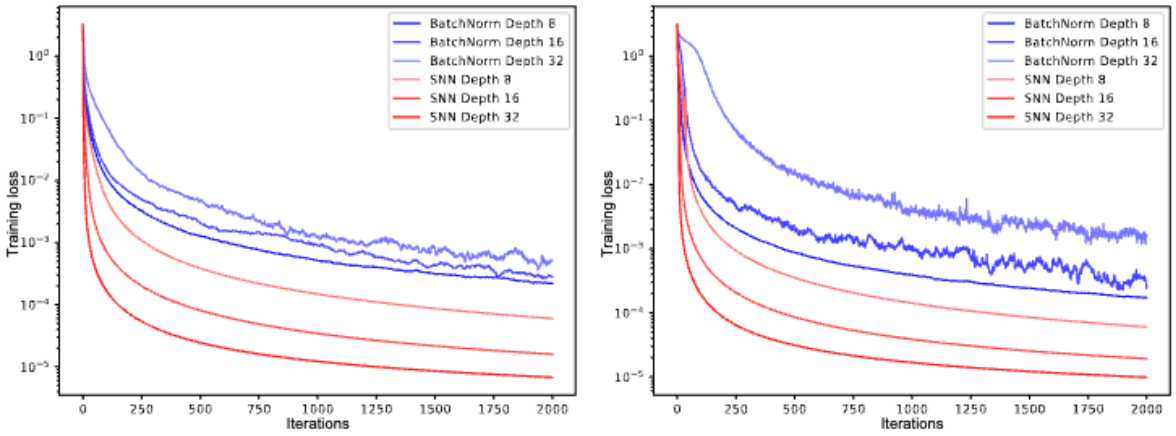


Figure 1: The left panel and the right panel show the training error (y-axis) for feed-forward neural networks (FNNs) with batch normalization (BatchNorm) and self-normalizing networks (SNN) across update steps (x-axis) on the MNIST dataset the CIFAR10 dataset, respectively. We tested networks with 8, 16, and 32 layers and learning rate 1e-5. FNNs with batch normalization exhibit high variance due to perturbations. In contrast, SNNs do not suffer from high variance as they are more robust to perturbations and learn faster.

Figure 1: Comparison between batch normalisation (BatchNorm) and self-normalising networks (SNN) on the MNIST dataset and the CIFAR10 dataset.

2.2 Convolutional Neural Networks for Histology Features from H&E Tissues

Histology images have been extensively used in cancer diagnosis and prognosis to help oncologists understand the characteristics of the tumour microenvironment [19]. The introduction of CNN has aided in the examination of tissue architecture and cellular morphology, playing a critical role in the identification of different histological patterns lung cancer diagnosis and prognosis [10]. Furthermore, quantification of image features such as high cellularity and microvascular proliferation has been extensively linked to tumor suppressor deficiency genes.

To extract image features from preprocessed H&E images, the original paper explicitly built the layers of vgg19 architecture [15] with batch normalisation. We can optimise this by implementing the vgg19bn architecture directly from 'torchvision.models', so there is no need for us to apply batch normalisation again. Note that the pre-existing weights come from 'ImageNet'. The vgg19 CNN is trained on 1024x1024 ROIs from histology slides, then tested on 9 overlapping 512x512 patches from these ROIs. Our Histology CNN PathNet consists of an image encoder and a classifier, and the last hidden extracts a representation $h_i \in \mathbf{R}^{32\times 1}$, which is used as an input for our multimodal fusion model.

2.3 Graph Convolutional Networks for Cell Features from H&E Tissues

According to [24], the spatial heterogeneity of cells in histopathology images can provide valuable insights into cancer invasion and progression, and cell graphs are a more explicit feature representation that directly model cell-to-cell interactions and cellular organisation. The GCN is the most complicated unimodal network in the original paper, and the preparatory work before training the GCN is crucial.

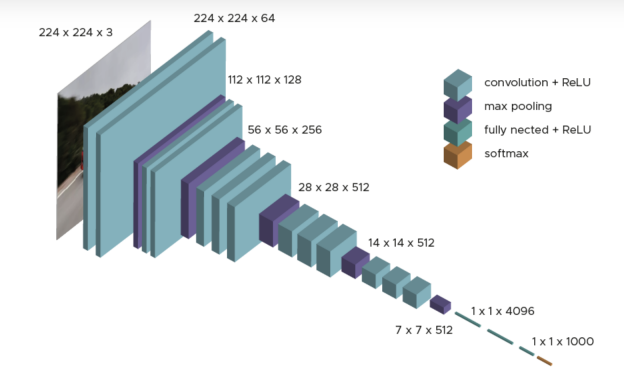


Figure 2: Demonstration of VGG architecture.

The main steps are,

- Nuclei Segmentation
- Cell Graph Construction
- Manual Cell Feature Extraction
- Contrastive Predictive Coding for Cell Feature Extraction
- Graph Convolution Network

2.3.1 Nuclei Segmentation

The traditional nuclei segmentation method has the issue of segmenting multiple nuclei as one, which makes the cell structure and nuclei shape become inaccurate. The novel method used in the original paper is cyclic GAN and conditional GAN from [23]. Given the limited availability of labeled nuclei segmentation data, this method first generates a dataset of perfectly annotated synthetic histology images via random polygon and cyclic GAN (cycle-consistent adversarial training). After adding more realism to the random polygons, we get a collection of ground-truth images and synthetic images with their corresponding masks. The conditional GAN initially generates a segmentation, which can then be refined by incorporating the buffered outputs and ground truth data into the discriminator in Figure 3.

For the cycle-consistent adversarial training, G (random polygon mask to pathology image generator), S (pathology image to polygon mask generator), D_N (discriminator for G), and D_M (discriminator for S). The loss equations are,

$$\mathcal{L}_{\text{GAN}}(G, D_N) = \mathbb{E}_{n\sim p_{\text{data}}(n)}\left[\log D_N(n)\right] + \mathbb{E}_{m\sim p_{\text{data}}(m)}\left[\log\left(1 - D_N(G(m))\right)\right]; \tag{1}$$

$$\mathcal{L}_{\text{GAN}}\left(S, D_M\right) = \mathbb{E}_{m\sim p_{\text{data}}\ (m)}\left[\log D_M(m)\right]\mathbb{E}_{n\sim p_{\text{data}}\ (n)}\left[\log\left(1 - D_M(S(n))\right)\right]; \tag{2}$$

$$\mathcal{L}_{cyc}(G, S) = \lambda_n\mathbb{E}_{n\sim p_{\text{data}}\ (n)}\left[\|G(S(n)) - n\|_1\right] + \lambda_m\mathbb{E}_{m\sim p_{\text{data}}\ (m)}\left[\|S(G(m)) - m\|_1\right]. \tag{3}$$

In equations (1) and (2), the logarithm of the discriminator output is for real samples, and the logarithm of one minus the discriminator outpu is for synthetic samples. Equation (3) ensures that the mappings between

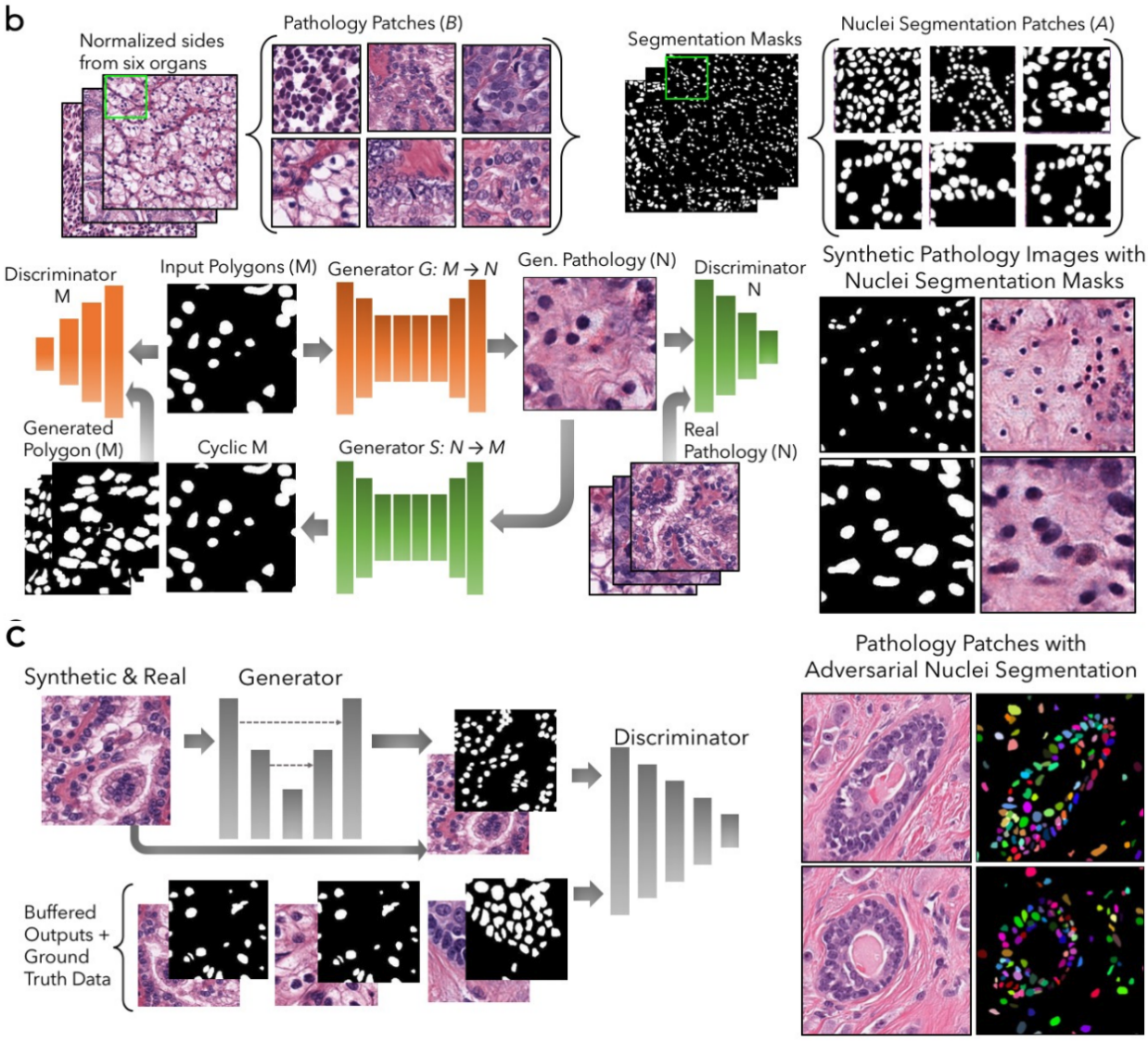


Figure 3: Illustration of cyclic GAN training and conditional GAN.

two domains are consistent.
For the conditional GAN, we have

$$\mathcal{L}_{\mathrm{GAN}}\left(S, D_M\right)=\mathbb{E}_{m,n\sim p_{\mathrm{data}}\ (m,n)}\left[\log D_M(m,n)\right]+\mathbb{E}_{n\sim p_{\mathrm{data}}\ (n)}\left[\log\left(1-D_M(m,S(n))\right)\right]. \tag{4}$$

2.3.2 Cell Graph Construction

It is straightforward if we use the K-Nearest Neighbors (KNN) algorithm from the Fast Library for Approximate Nearest Neighbours (FLANN) library to construct the edges between the adjacent nodes. The adjacency matrix to five nearest neighbours is,

$$A_{ij}=\begin{cases}1 & \text{if } j\in \mathrm{KNN}(i) \text{ and } D(i,j)<d\\ 0 & \text{otherwise}\end{cases} \qquad \text{where } i,j\leq 5. \tag{5}$$

2.3.3 Manual Cell Feature Extraction

By using segmentation from the previous subsection, we can easily find the contours. Then it becomes straightforward to extract eight contour features (major axis length, minor axis length, angular orientation, eccentricity, roundness, area, and solidity), as well as four texture features from gray-level co-occurrence matrices (GLCM) (dissimilarity, homogeneity, angular second moment, and energy).

2.3.4 Contrastive Predictive Coding (CPC) model for Cell Feature Extraction

CPC is designed to capture high-level representations shared among different portions of the complete signal [27]. We first apply the pixel CNN model to perform aggregation,

```
self.MaskConv0 = maskConv0(n_channel, h, k_size=7, stride=1, pad=3)
```

where 'MaskConv0' is a 2D masked convolution operation that aggregates information from the local neighborhood of each pixel. Then we also apply resnet 50 as an image encoder to compute latent representation for each patch. After computing each patch's context, linear predictions can be made, and our goal is to justify whether the network can correctly match predicted features with ground truth features among negative targets.

```
accuracy[i] = torch.sum(torch.eq(torch.argmax(self.softmax(total), dim=1),
                torch.arange(0, bs * cols).to(device))).item()
```

The model training uses the customised contrastive loss as follows,

$$\mathcal{L}_{\mathrm{N}}=-\mathbb{E}_X\left[\log\frac{f_k\left(x_{t+k},c_t\right)}{\sum_{x_j\in X} f_k\left(x_j,c_t\right)}\right]. \tag{6}$$

Its optimisation leads to maximising the mutual information between the available context c_t computed from $\{x_i\}, i\leq t$ and future observations $x_{t+k}, k>0$.

2.3.5 Graph Convolutional Network

Given a cell graph with nodes and edges, GCNs need to aggregate the feature vectors from the previous layer $(k-1)$ of all its neighboring nodes $u\in\mathcal{N}(v)$, then combine its previous feature vector $h_v^{(k-1)}$ with the

current aggregation $a_v^{(k)}$. Therefore, the process can be summarised as,

$$\begin{aligned} a_v^{(k)} &= \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) \\ h_v^{(k)} &= \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right). \end{aligned} \tag{7}$$

By adapting definitions from GraphSAGE in [13], equation (7) can be rewritten as,

$$\begin{aligned} a_v^{(k)} &= \text{MAX}\left(\left\{\text{ReLU}\left(W \cdot h_u^{(k-1)}\right), \forall u \in \mathcal{N}(v)\right\}\right) \\ h_v^{(k)} &= W \cdot \left[h_v^{(k-1)}, a_v^{(k)}\right]. \end{aligned} \tag{8}$$

After normalising the feature vectors and edges, equation (8) can be achieved via 'torch_geometric.nn.SAGEConv' operation with the code.

```
self.conv2 = torch_geometric.nn.SAGEConv(nhid, nhid)
```

To encode the hierarchical structure of cell graphs, the self-attention pooling strategy is also implemented, SAGPooling in [18], which can be achieved by 'torch_geometric.nn.GraphConv'.

```
torch_geometric.nn.SAGPooling(nhid, ratio=pooling_ratio, GNN=torch_geometric.nn.GraphConv)
```

With the node feature X and adjacency matrix A, the attention score $Z = \sigma\left(\text{SAGEConv}\left(X, A + A^2\right)\right)$ is computed to decide the contribution of each node embedding in the pooling receptive field to the next network layer. All the node features are pooled into a $h_g \in \mathbf{R}^{32\times 1}$, which is used as an input for our multimodal fusion model.

2.4 Fusion Strategy

The breakthrough occurs when we notice that feature interactions across unimodal feature representations can be captured by the matrix outer product. Our fusion strategy can be divided into two steps (1) **the Kronecker product** and (2) **the gating-attention mechanism**.

2.4.1 Kronecker Product

With the multimodal representation $(\mathbf{h}_i, \mathbf{h}_g, \mathbf{h}_n)$, the Kronecker product of fusing heterogeneous modalities can be computed as,

$$\mathbf{h}_{\text{fusion}} = \begin{bmatrix} \mathbf{h_i} \\ \text{output size} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{h_g} \\ \text{output size} \end{bmatrix} \otimes \begin{bmatrix} \mathbf{h_n} \\ \text{output size} \end{bmatrix} \tag{9}$$

where the default output size is 1 and the outer product $\otimes$ is achieved by batch matrix multiplication, 'torch.bmm'. We append one to each unimodal feature representation to preserve the unimodal feature interactions when computing the bimodal and trimodal interactions. After appending this one, the feature representations we have are $[33\times 1]$, $[33\times 1]$ and $[33\times 1]$, assuming the image, graph, and genomic dimensions are all 32. The trilinear fusion can be understood as applying batch matrix multiplication twice as in algorithm 1. The bilinear fusion tensor has dimension of $[33 \times 33]$.The trilinear fusion tensor has dimension of $[33\times 33\times 33]$. The multimodal network is initialized with pretrained weights from the unimodal networks, and fine-tuning of the Histology GCN and Genomic SNN begins after the first few epochs.

Algorithm 1 Kronecker Product

1. o1 = torch.cat((o1, torch.ones(o1.shape[0], 1, device=device)), 1) ▷ Append one to use the unperturbed unimodal features
2. o12 = torch.bmm(o1.unsqueeze(2), o2.unsqueeze(1)).flatten(start_dim=1) ▷ Apply batch matrix multiplication to obtain the Kronecker product of two modalities
3. o123 = torch.bmm(o12.unsqueeze(2), o3.unsqueeze(1)).flatten(start_dim=1) ▷ Apply batch matrix multiplication again to obtain the Kronecker product of all three modalities

2.4.2 Gating-attention Mechanism

The gating-based attention mechanism [2] controls the expressiveness of features of each modality. In other words, it assigns the relative importance score to each modality before taking the above Kronecker product, and reduces the size of the feature space. It can be summarised as,

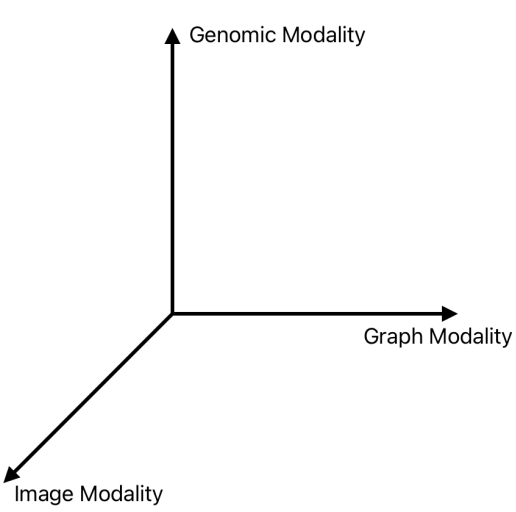
$$\begin{aligned} \mathbf{h}_{m,\text{ gated}} &= \mathbf{z}_m * \mathbf{h}_m, \forall m \in \{i, g, n\} \\ \text{with, } \mathbf{h}_m &= \text{ReLU}\left(W_m \cdot \mathbf{h}_m\right) \\ \mathbf{z}_m &= \sigma\left(W_{ign \to m} \cdot [\mathbf{h}_i, \mathbf{h}_g, \mathbf{h}_n]\right). \end{aligned} \tag{10}$$

Where z_m is an attention weight vector, and W_m and $W_{ign \to m}$ are weight matrix parameters. We convert equation (10) into the following code.

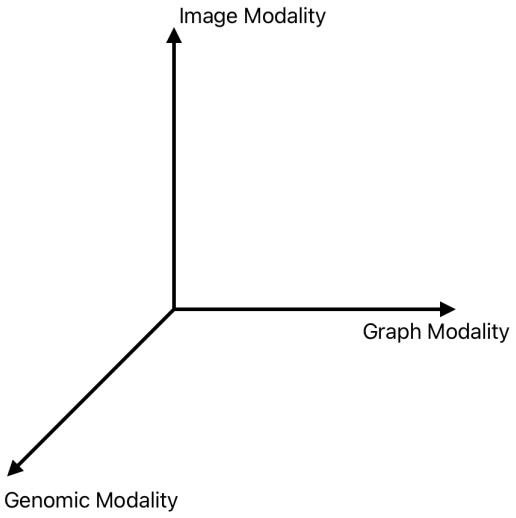
Algorithm 2 Gating-based Attention

1. self.linear_h1 = nn.Sequential(nn.Linear(dim1_og, dim1), nn.ReLU()) ▷ Let the unimodal feature representations be linearly transformed by the weight matrix
2. self.linear_z1 = (nn.Bilinear(dim1_og, dim3_og, dim1) ▷ Define the bilinear transformation between image and genomic unimodal representations (can be changed to others)
3. self.linear_o1 = nn.Sequential(nn.Linear(dim1, dim1), nn.ReLU(), nn.Dropout) ▷ Define element-wise product of the attention weight vector and the unimodal feature representation
4. h1 = self.linear_h1(vec1) ▷ Correspond to the second line of 10
5. z1 = (self.linear_z1(vec1, vec3) ▷ Correspond to the third line of 10
6. o1 = self.linear_o1(nn.Sigmoid()(z1) * h1) ▷ Correspond to the first line of 10

When fusing multimodal representations, not all gates are open; one gate remains open to allow other modalities to undergo a bilinear transformation with it in Figure 4. For example, the genomic modality gates over the image and graph modalities in survival outcome prediction, and the image modality gates over the graph and genomic modalities in grade classification.



(a) Survival Outcome Analysis



(b) Grade Classification

Figure 4: Demonstration of trilinear fusion; left: image and graph modalities are gated over by the genomic modalities for survival outcome prediction; right: genomic and graph modalities are gated over by the image modality for grade classification.

3 Environment Setup

Word Count: 246

We have managed to establish a new conda environment to train the models, and our 'environment.yml' contains up-to-date packages rather than the older versions used in the original paper. Since the original code is outdated, we have implemented several optimisations to enhance the speed of the training and testing process.

3.1 Package Version

Our computational environment is built with these key packages, including Python 3.10.8, PyTorch 2.1.2, PyTorch Geometric 2.5.3, PyTorch Scatter 2.1.2, TorchVision 0.16.2, FLANN 1.6.14 and Lifelines 0.28.0. More details can be found in our GitLab repository.

3.2 CUDA and cuDNN Setup

We initially attempted to run the code locally with MacOS, but the Mac GPU does not support cell graphs and fusion operations. Therefore, most of our work has been conducted on HPC with the NVIDIA GPU. PyTorch 2.1.2 is linked to CUDA 12.1, and cuDNN version is 9.2.0. Installing cuDNN is essential for speeding up parallel computing with multiple GPUs as follows,

```
if len(gpu_ids) > 0:
    assert(torch.cuda.is_available())
    net.to(gpu_ids[0])
    net = torch.nn.DataParallel(net, gpu_ids)
```

3.3 Update Outdated Code

With newer packages in our environment, several modifications have been made to the original code. We give short examples here. 'torch_geometric.nn.SAGPooling' returns more parameters compared to the original version, including node features, edge indices, edge features, batch vectors, and attention scores. The original code should be changed to,

```
x, edge_index, edge_attr, batch, _, _ = self.pool1(x, edge_index, edge_attr, batch)
```

During the GCN training, we need to choose a graph neural network layer for calculating projection, but the one specified in the original code has been outdated. We rewrite the code as,

```
self.pool2 = torch_geometric.nn.SAGPooling(
    nhid, ratio=pooling_ratio, GNN=torch_geometric.nn.GraphConv)
```

Additionally, the usage of scatter(reduce='max') is not enabled for acceleration purposes in the original code. We have fixed this issue by using PyTorch Scatter.

4 Data Acquisition

Word Count: 410

For TCGA-GBMLGG, the histology slides are not directly downloaded from TCGA. We obtain the 1024x1024 ROIs from diagnostic slides curated by [25] (the original paper also used this dataset). We generate 512x512 patches from these ROIs to test our models. Although nuclei segmentation images for 1024x1024 ROIs are not available, it is feasible to regenerate them using the conditional GAN or CellViT (CellViT will be covered later), and .pt files of the reconstructed cell graphs have been provided from the original paper. There are a total of 1505 images for 769 patients. 320 genomic features including CNV (79), mutation status (1), and mRNASeq expression (240) are obtained from TCGA and cBioPortal [6]. For evaluation purposes, we also propose three categories of GBMLGG, (1) IDH-wildtype astrocytomas (IDHwt ATC), (2)IDH-mutant astrocytomas (IDHmut ATC), and (3) IDH-mutant with 1p/19q-codeleted, oligodendrogliomas (ODG). These categories are classified based on histology and molecular subtype from 'grade_data.csv'.

For TCGA-KIRC, 512x512 ROIs are manually extracted from whole slide images from TCGA. For 417 patients, we collect 3 512×512 40x ROIs per patient, resulting in a total of H&E 1251 images. We pair these images with 357 genomic features from CNV (117) of genes and mRNASeq expression(240). There is no missing data issue in TCGA-KIRC.

4.1 Missing Data Issue

There exists missing data issues in TCGA-GBMLGG. In Figure 5, $(769 - 697 = 72)$ patients have missing molecular subtype such as IDHmut with codeletion, $(769 - 736 = 33)$ patients have missing histological subtype such as astrocytoma and oligodendroglioma, and $(769 - 508 = 256)$ patients have missing mRNASeq. To overcome missingness, we create different train-test splits based on (1) survival prediction or grade classification (2) combination of modalities of histology and genomic. For example, the missing histological subtype will be ignored for grade classification, and the missing molecular subtype will be ignored if genomic modality is involved.

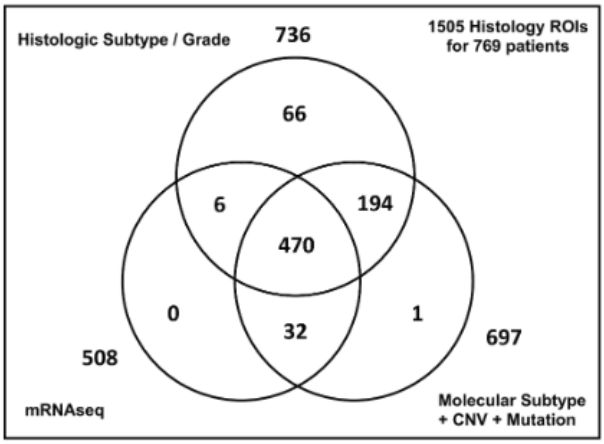


Figure 5: Demonstration of missing data in TCGA-GBMLGG.

4.2 Cell Graph Construction

Segmentation is essential for cell graph .pt files, so we explicitly implement the previously mentioned conditional GAN to segment nuclei. The construction is completed by using handcrafted cell features and CPC features. We have not generated cell graphs for all 1505 images due to time constraints, and reproduced examples are in Figure 6.

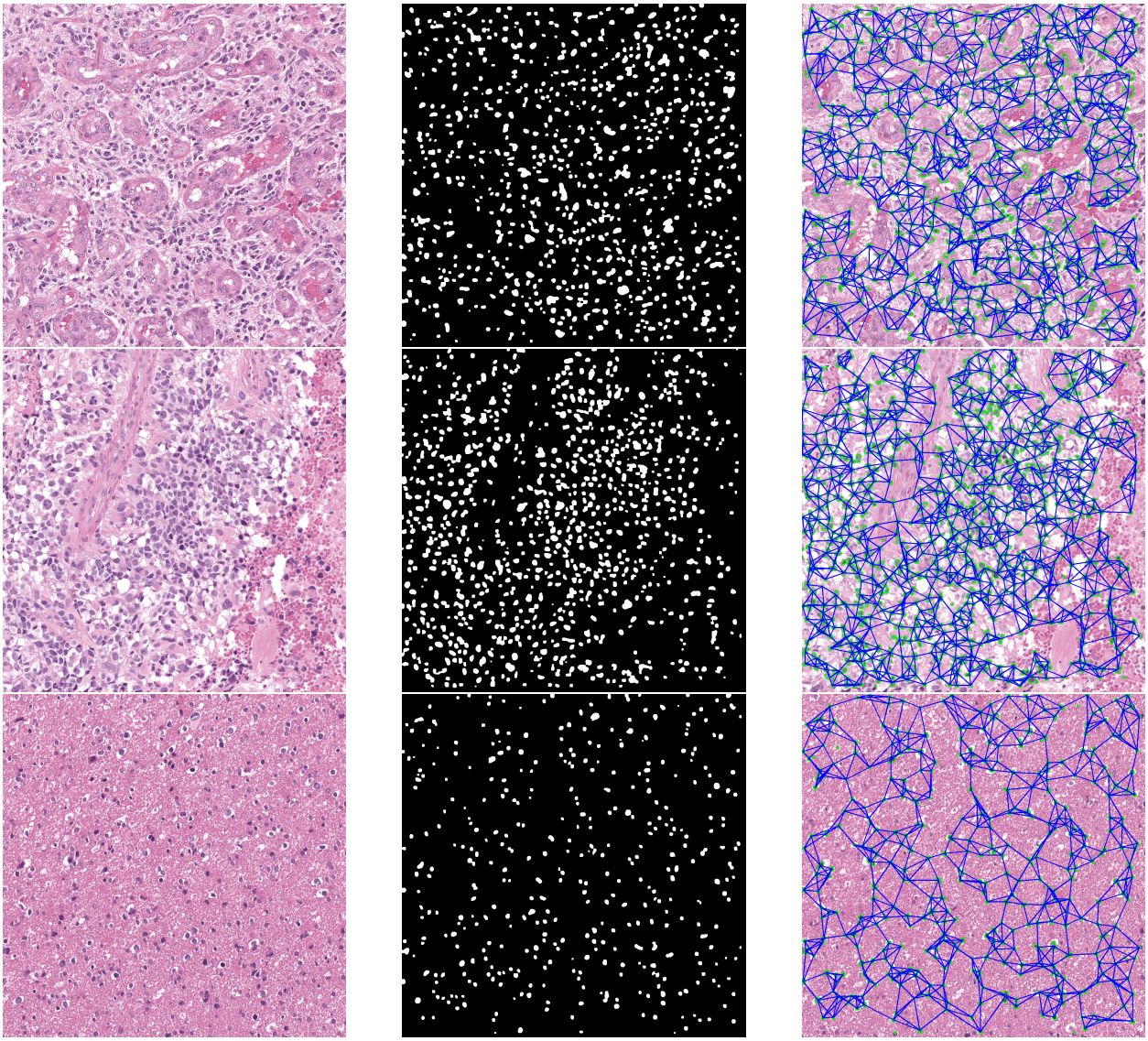


Figure 6: Demonstration of cell graph construction for GCNs; left: 1024x1024 histology image; middle: nuclei segmentation; right: graph cell used for GCNs.

4.3 Update Cell Graph Files with Errors

The provided .pt cell graph files from the original code have two errors (1) the file version is outdated and (2) some edge indices are greater than the number of nodes in Figure 7.

The first one can be solved with the following code,

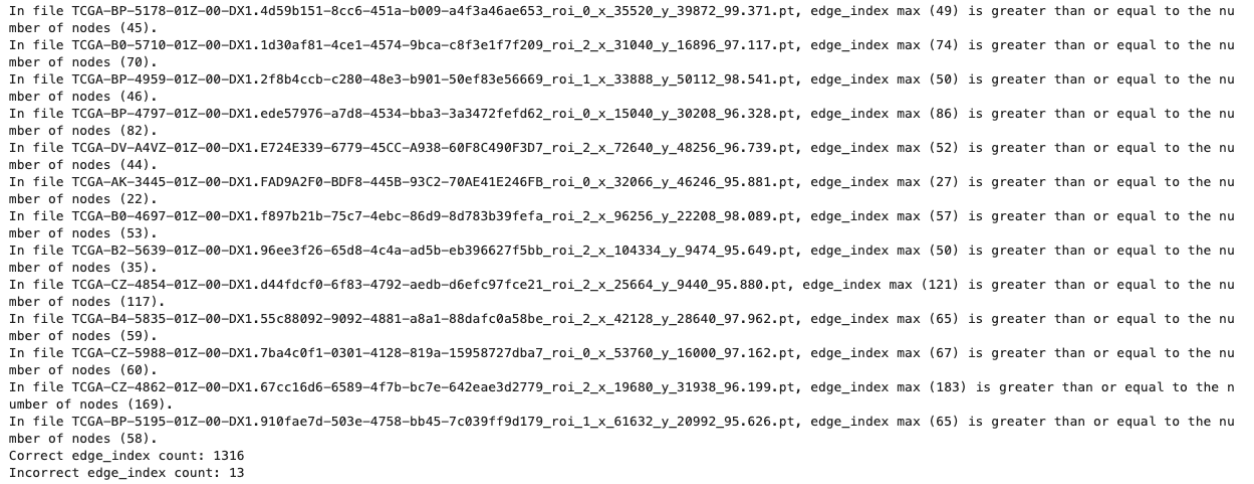


Figure 7: Screenshot of errors from invalid edge indices.

```
torch_geometric.data.data.Data.from_dict(data.__dict__)
```

The second one can be solved by using the mask to filter out invalid edges to ensure edge indices are in the correct range.

```
valid_edges_mask = (edge_index[0] < num_nodes) & (edge_index[1] < num_nodes)
filtered_edge_index = edge_index[:, valid_edges_mask]
new_data.edge_index = filtered_edge_index
```

5 Model Implementation and Training

Word Count: 749

We cover the details of training the proposed unimodal and multimodal networks mentioned above. We not only retrain the models using the original data splits from the paper but also create our own data splits to train from scratch.

5.1 Importance of Making Data Splits

Our 15-fold Monte Carlo cross-validation is inspired by train-test splits from [25] with 80% training and 20% testing. Data splits not only assign train and test datasets but also play a critical role in aligning multimodal data, extracting image features, and storing features based on the specific training task.

5.1.1 Clean and Align Multimodal Data

When creating data splits, it first cleans the entire dataset by converting non-numerical values and imputing missing values. The code below generates values for histological and molecular subtype.

```
hs2int = {"Missing": -1, "astrocytoma": 0, "oligoastrocytoma": 1,
"oligodendroglioma": 2, "glioblastoma": 3}
ms2int = {"Missing": -1, "IDHwt": 0, "IDHmut-non-codel": 1, "IDHmut-codel": 2}
```

The code below imputes missing values.

```
all_dataset["Molecular subtype"] = all_dataset["Molecular subtype"].fillna("Missing")
all_dataset["Grade"] = all_dataset["Grade"].fillna(1)
```

Then the aligning function ensures that image features, graph features, genomic features, censored events, survival months, and grades are all correctly assigned to the corresponding patient by TCGA-ID.

```
x_omic.append(
    np.array(
        all_dataset[all_dataset["TCGA ID"] == pat_name].drop(
            metadata, axis=1)))
e.append(int(all_dataset[all_dataset["TCGA ID"] == pat_name]["censored"].iloc[0]))
```

5.1.2 Extract Image Features

It takes longer to create data splits when the script loads a trained CNN model to extract and store image features, which can be used directly for fusion rather than loading the encoder again. Its 'get_vgg_features' fucntion contains,

```
x_path = torch.unsqueeze(normalize(x_path), dim=0)
features, hazard = model(x_path=x_path.to(device))
```

The size of data splits becomes much larger with image features. Graph features in the splits are relatively small because we only append .pt file paths.

5.1.3 Data Splits for Specific Tasks

Customised data splits are created for different training and testing frameworks. We use 'use_vgg_features' if the task requires image modality such as CNN+SNN, use 'ignore_missing_histype' if the task is grade classification, and use 'ignore_missing_moltype' if the task requires genomic modality such as GCN+SNN.

5.2 Loss Function

We apply two loss functions to train unimodal and multimodal networks.

5.2.1 Cox Partial Likelihood Loss for Survival Analysis

The cox loss function (11) is used when training for survival outcome prediction.

$$l(\beta, X) = -\sum_{i \in U} \left(X_i \beta - \log \sum_{j \subset R_i} e^{X_j \beta} \right) \tag{11}$$

The code we use is adapted from Cox-nnet [8].

5.2.2 Cross Entropy Loss for Grade Classification

The cross entropy loss (12) are used when training for grade classification.

$$L = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log \left(\hat{y}_{ic} \right) \tag{12}$$

5.3 Baseline Model for Survival Analysis

For both TCGA-GBMLGG and TCGA-KIRC, we want to define a baseline against standard statistical approaches / WHO paradigm for survival outcome prediction, hence we train Cox Proportion Hazard Models using age, gender, grade and molecular subtype. The key step for baseline training is,

```
cph = CoxPHFitter(penalizer=penalizer)
cph.fit(train[feats], duration_col='Survival months',
event_col='censored', show_progress=False)
```

The example training is in Figure 8.

```
ckpt_name = './checkpoints/TCGA_GBMLGG/surv_15_paper/'
trainCox_GBMLGG(ckpt_name=ckpt_name, model='cox_molgrade', penalizer=0)
```

```
# Missing Molecular Subtype: 72
# Missing IDH Mutation: 72
# Missing 1p19q Codeletion: 72
# Missing Histological Subtype: 33
# Missing Grade: 33
C-Indices across Splits [0.7467735627688697, 0.7828759604829857, 0.7398725066831174, 0.7757697456492637, 0.7537693561532193, 0.7736149437486733, 0.82
19083366258396, 0.7909964692036092, 0.8236830648672181, 0.7919983670136762, 0.7793599132085707, 0.7514867337602927, 0.7719534050179212, 0.78288169200
26438, 0.7699017445357931]
Average C-Index: 0.7771 ± 0.013
```

Figure 8: Screenshot of Cox Proportion Hazard Model training.

5.4 Training Details for TCGA-GBMLGG

For both survival analysis and grade classification, we first train three unimodal networks mentioned earlier: Histology CNN, Histology GCN, and Genomic SNN. We then use the fusion strategy (the Kronecker product and the gating-based attention) to train multimodal networks: CNN+SNN, GCN+SNN, and GCN+CNN+SNN. We also train ensembling models that input same modality twice into the fusion model.

For survival analysis, all networks are activated using the sigmoid function, and output hazard score is scaled to be between -3 and +3. For grade classification, all networks are activated using the log softmax to compute scores for WHO grades of II, III, IV.

We use pretrained weights from ImageNet to initialise CNN, and the tuning is carried out with a learning rate of 0.0005 and a batch size of 8. We use the self-normalising weights from [16] to initialise GCN and SNN, and the tuning is carried out with a learning rate of 0.002 and a batch size of 32 and 64, respectively. We do not apply regularisation to CNN and GCN, but a L1 regularisation is applied to SNN as follows,

```
for W in model.parameters():
if l1_reg is None:
    l1_reg = torch.abs(W).sum()
```

For multimodal network training, we recall Figure 4, genomic modality gates over the other two modalities in survival anlaysis, while in grade classification, the image modality gates over. Dimension reduction is applied to the gated modality to reduce the output size of the Kronecker product in the trimodal network. For example, linear layers in the genomic modality have 32 hideen units, but linear layers in the image and graph modalities only have 16 hiddent units. After appending one to preserve interactions, feature maps have size of [33x17x17] in CNN+GCN+SNN. GCN and SNN are initially frozen, but we unfreeze them at epoch 5 and continue training for an additional 25 epochs with a learning rate of 0.0001.

The gate scaling in the original command is misleading, we need to ensure path_scale = grph_scale = 2.

5.5 Training Details for TCGA-KIRC

The training process is very similar to the one described above for TCGA-GBMLGG. We use the Fuhrman grading system rather than the WHO grading system for CCRCC, with severity ranging from G1 to G4. The Fuhrman grade is used as a comparative baseline in survival analysis. Since CCRCC does not have multiple molecular subtypes as GBMLGG, we conduct analysis on survival durations rather than subtyping groups. Grade classification is not performed for CCRCC.

6 Validation and Testing

Word Count: 1168

We validate the trained model's performance on the test dataset of TCGA-GBMLGG and TCGA-KIRC. We present our reproduced results to compare with those from the original paper, and conclude pathomic fusion model outperforms in terms of fine-grained patient stratification and interpretability.

6.1 Survival Analysis for TCGA-GBMLGG

We evaluate our models using the Concordance Index, which represents the fraction of all pairs of samples whose predicted survival times are correctly ordered among all uncensored samples. The output hazard scores are divided into 33-66-100 to match grades of II, III and IV. To better observe the significance of the patient stratification, we introduce p-values, which can be obtained by the log rank test function.

Histology CNN testing is incorrect in the original code because it uses image encoder to extract image features twice, once during data split creation and once in implementing vgg19 networks. Therefore, we create specific testing data splits for CNN, and more details can be found in our repository.

6.1.1 Using Original Data Splits for Survival Analysis

In Figure 9 with original data splits, pathomic fusion (CNN+GCN+SNN) achieves the largest c-index of **0.827** compared to baselines, unimodal, and bimodal networks. It also demonstrates that fusing more unimodal feature representations results in a higher c-index. We observe that trimodal fusion has relatively small p-value of $8.96e^{-2}$ for testing the difference in [0,33] vs. (33,66] percentiles compared to bimodal fusion. It means that the difference between low and intermediate risk patient can be statistically significant under CNN+GCN+SNN. We also test network ensembling using CNN+CNN, GCN+GCN, SNN+SNN, and there is no significant improvement in c-index because inputting the same modality twice leads to overfitting.

By comparing Figures 9 and 10, all c-index results, except for CNN, differ by no more than **0.01** from those presented in the original paper. The difference in the CNN results is due to our own data splits that extract image features once and then return testing results immediately. Correct data splits for CNN testing are not provided in the original code. Overall, we have obtained testing results very similar to those in the original paper using their data splits.

6.1.2 Using New Data Splits for Survival Analysis

To train and test our models from scratch, we create new data splits (including feature extraction steps) to rerun the same code. In Figure 11, pathomic fusion (CNN+GCN+SNN) still achieves the largest c-index of **0.823**, and the pattern in the c-index corroborates the conclusion from the original data splits: trimodal pathomic fusion outperforms other models by effectively capturing multiple modalities.

6.1.3 Using the Data Splits by CONCH for Survival Analysis

We extract new feature representations using CONCH (covered in Further Improvement section), and create corresponding data splits for CONCH and its combination with other modalities. Surprisingly,

	C-Index	P-Value (<50% vs. >50%)	P-Value (<33% vs. 33-66%)	P-Value (33-66% vs. >66%)
Cox (Age+Gender)	0.7316 ± 0.012	1.900892e-92	1.475424e-38	5.927583e-27
Cox (Subtype)	0.7600 ± 0.011	2.064634e-228	2.274412e-26	3.252834e-51
Cox (Grade)	0.7379 ± 0.013	5.998349e-224	9.568192e-23	2.941556e-66
Cox (Grade+Subtype)	0.7771 ± 0.013	5.291938e-215	1.140947e-40	5.016280e-52
Genomic SNN	0.8023 ± 0.017	7.880654e-53	7.693598e-01	1.372293e-83
Genomic (SNN + SNN)	0.7924 ± 0.013	2.688850e-52	NaN	2.454380e-143
Histology GCN	0.7388 ± 0.020	3.194850e-26	5.658022e-01	4.368388e-26
Histology (GCN + GCN)	0.7373 ± 0.025	2.646524e-29	2.554809e-02	8.050384e-21
Histology CNN	0.7710 ± 0.018	1.479287e-37	1.398231e-06	1.275895e-26
Histology (CNN + CNN)	0.7936 ± 0.014	1.331124e-46	1.146139e-05	1.568596e-26
Pathomic F. (CNN+SNN)	0.8198 ± 0.011	1.130308e-55	9.521607e-02	1.914901e-77
Pathomic F. (GCN+SNN)	0.8114 ± 0.012	2.429412e-54	3.130986e-01	3.392474e-76
Pathomic F. (CNN+GCN+SNN)	0.8273 ± 0.013	5.983222e-59	8.965891e-02	2.348627e-73

Figure 9: GBMLGG; the c-index values and p-values from our re-trained model by using the same data splits as the original paper.

	C-Index	P-Value (<50% vs. >50%)	P-Value (<33% vs. 33-66%)	P-Value (33-66% vs. >66%)
Cox (Age+Gender)	0.7316 ± 0.012	1.900892e-92	1.475424e-38	5.927583e-27
Cox (Subtype)	0.7595 ± 0.011	2.064634e-228	4.653368e-26	1.444895e-51
Cox (Grade)	0.7379 ± 0.013	5.998349e-224	9.568192e-23	2.941556e-66
Cox (Grade+Subtype)	0.7767 ± 0.013	5.291938e-215	1.140947e-40	5.016280e-52
Genomic SNN	0.8081 ± 0.014	1.519321e-52	1.531065e-01	2.164183e-81
Genomic (SNN + SNN)	0.7944 ± 0.014	5.090139e-54	NaN	1.641976e-138
Histology GCN	0.7464 ± 0.022	1.618834e-21	2.288528e-03	4.202852e-15
Histology (GCN + GCN)	0.7447 ± 0.021	5.620511e-10	8.043318e-02	9.135586e-10
Histology CNN	0.7921 ± 0.014	5.092422e-40	1.774952e-07	6.604784e-25
Histology (CNN + CNN)	0.7935 ± 0.014	1.370966e-44	1.966565e-06	4.653116e-26
Pathomic F. (CNN+SNN)	0.8202 ± 0.009	5.182205e-57	1.029138e-01	4.562223e-79
Pathomic F. (GCN+SNN)	0.8121 ± 0.010	1.054539e-55	2.089946e-01	3.088825e-80
Pathomic F. (CNN+GCN+SNN)	0.8258 ± 0.009	7.087806e-57	2.681224e-03	5.823266e-74

Figure 10: GBMLGG; the c-index values and p-values copied from the original paper.

	C-Index	P-Value (<50% vs. >50%)	P-Value (<33% vs. 33-66%)	P-Value (33-66% vs. >66%)
Cox (Age+Gender)	0.7316 ± 0.012	1.900892e-92	1.475424e-38	5.927583e-27
Cox (Subtype)	0.7595 ± 0.011	2.064634e-228	4.653368e-26	1.444895e-51
Cox (Grade)	0.7379 ± 0.013	5.998349e-224	9.568192e-23	2.941556e-66
Cox (Grade+Subtype)	0.7767 ± 0.013	5.291938e-215	1.140947e-40	5.016280e-52
Genomic SNN	0.8100 ± 0.015	5.316290e-47	1.010861e-01	7.301739e-84
Genomic (SNN + SNN)	0.8092 ± 0.015	2.749143e-51	1.530080e-01	1.113833e-83
Histology GCN	0.7247 ± 0.022	1.369501e-25	1.023542e-03	1.689381e-16
Histology (GCN + GCN)	0.7287 ± 0.020	2.861252e-30	4.127735e-04	1.918003e-15
Histology CNN	0.7743 ± 0.017	7.938201e-36	2.564271e-10	1.462746e-15
Histology (CNN + CNN)	0.7748 ± 0.017	9.584131e-37	4.815573e-09	2.767427e-16
Pathomic F. (CNN+SNN)	0.8203 ± 0.019	3.444377e-54	3.497569e-01	9.893598e-81
Pathomic F. (GCN+SNN)	0.8121 ± 0.015	5.617681e-55	5.249495e-01	1.196039e-81
Pathomic F. (CNN+GCN+SNN)	0.8233 ± 0.017	1.179099e-51	4.096507e-01	1.451836e-81

Figure 11: GBMLGG; the c-index values and p-valuesfrom our re-trained model by using the new data splits

CONCH+SNN achieves a higher c-index of **0.831** compared to 0.822 from CONCH+GCN+SNN in Figure 12.

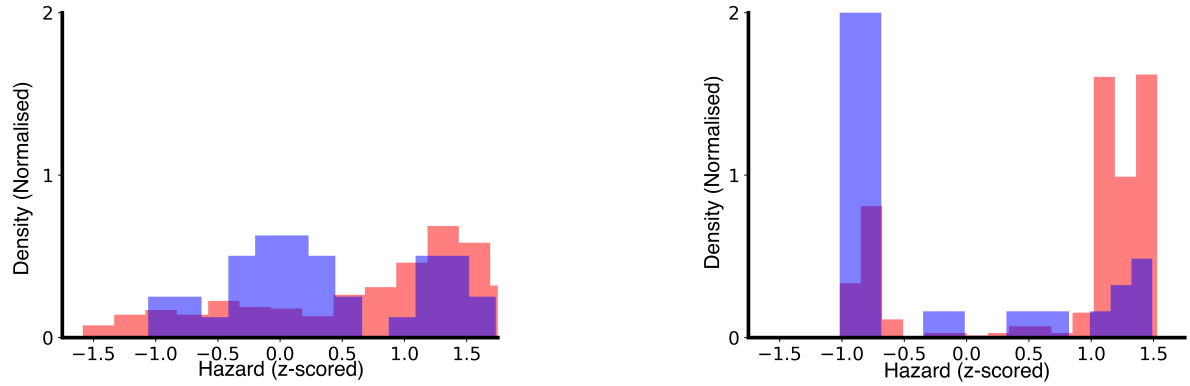
	C-Index	P-Value (<50% vs. >50%)	P-Value (<33% vs. 33-66%)	P-Value (33-66% vs. >66%)
Histology CONCH	0.7742 ± 0.016	4.750643e-41	0.041872	3.418225e-31
Genomic SNN	0.8023 ± 0.017	7.880654e-53	0.769360	1.372293e-83
Histology GCN	0.7388 ± 0.020	3.194850e-26	0.565802	4.368388e-26
Pathomic F. (CONCH+SNN)	0.8309 ± 0.015	7.267651e-61	0.010480	5.082288e-73
Pathomic F. (GCN+SNN)	0.8018 ± 0.016	5.864152e-63	0.093537	1.798589e-77
Pathomic F. (CONCH+GCN+SNN)	0.8218 ± 0.013	6.298128e-58	0.113025	1.775979e-74

Figure 12: GBMLGG; the c-index values from our re-trained model by using the new data splits with CONCH

6.1.4 Evaluation Plots with Original Data Splits

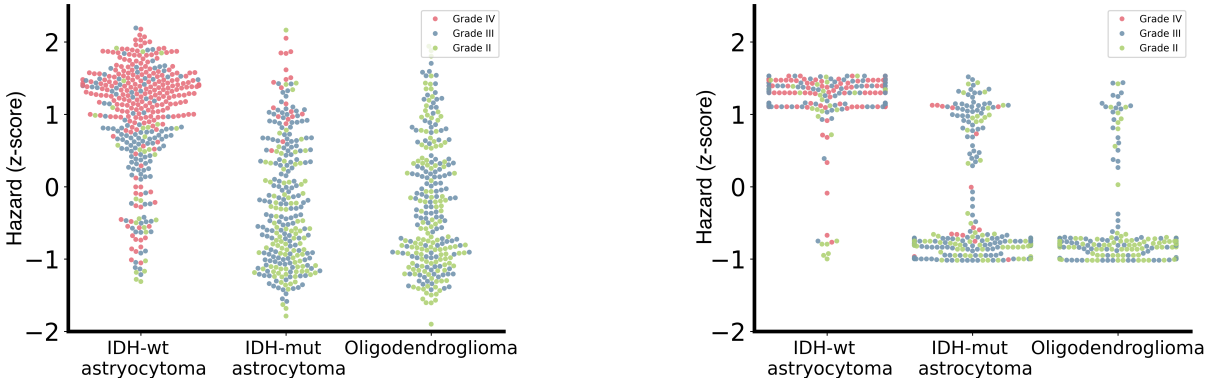
The evaluation plots presented in this section are generated based on testing results from the original data splits.

In Figure 13, there are glioma hazard distributions amongst shorter vs. longer surviving uncensored patients for histology CNN and pathomic fusion (CNN+GCN+SNN). We observe that pathomic fusion assigns risk to patients in three concentrated clusters (corroborate with IDHwt ATC, IDHmut ATC, and ODG). However, CNN assigns low hazard values to patients with intermediate-to-high risk. Additionally, pathomic fusion provides more consistent predictions in patients' survival duration. In Figure 14, there are swarm plots in which GBMLGG are classified into three categories, and CNN displays longer tails and more varied shapes across different molecular subtypes. We also notice that it is much easier to decide the hazard scores for a specific molecular subtype from the pathomic fusion swarm plot.



(a) Histology CNN (b) Pathomic Fusion

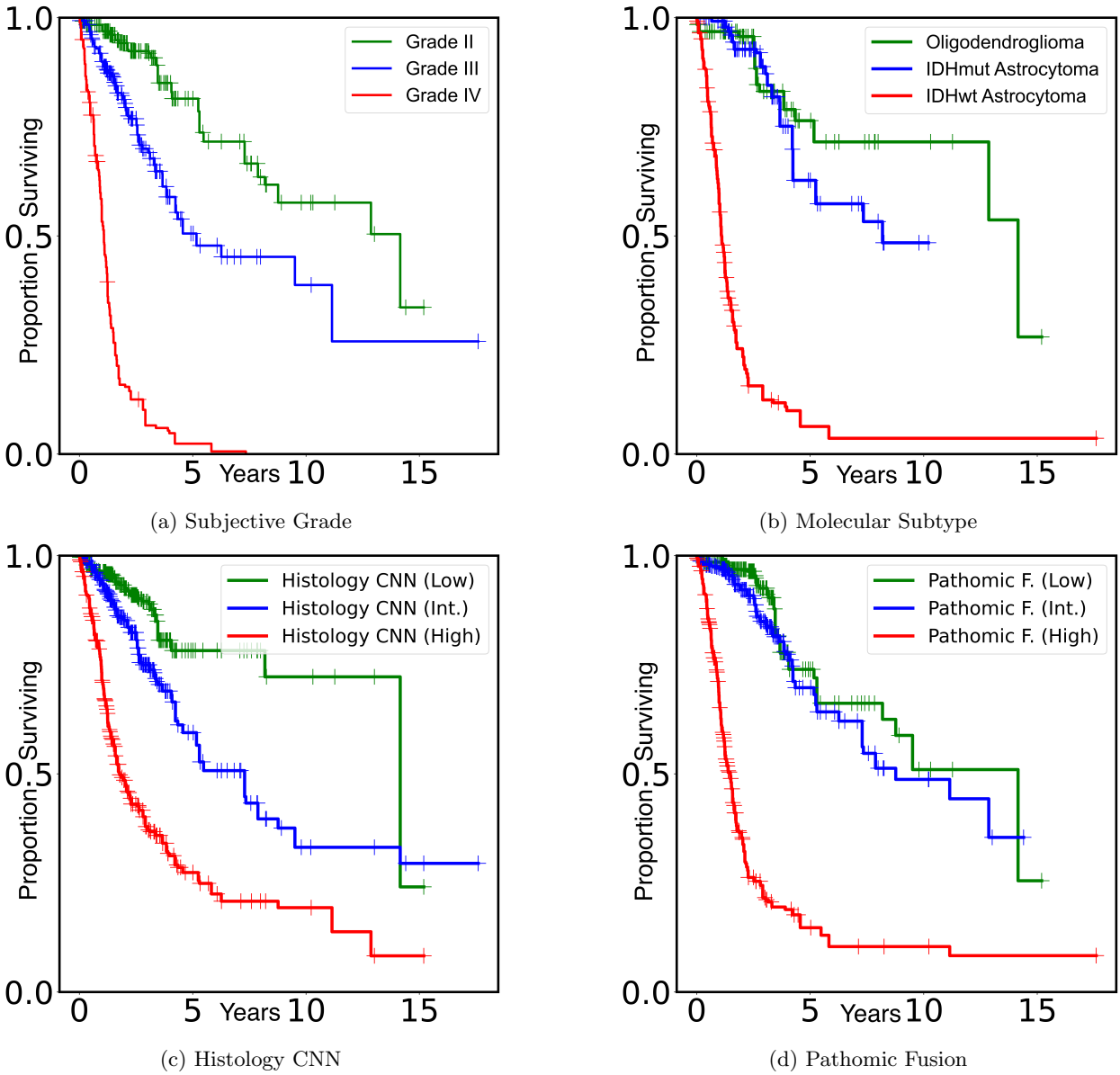
Figure 13: GBMLGG hazard distributions; red for patient survival less then 5 years, blue for patient survival more than 5 years.



(a) Histology CNN (b) Pathomic Fusion

Figure 14: GBMLGG swarm plots for Histology CNN; y-axis is the Hzard value after z-scored

In Figure 15, we pool the predicted hazards from the test splits and plot them against survival time to generate the Kaplan-Meier curves. The Kaplan-Meier analysis is conducted on subjective grade, molecular subtype (WHO paradigm), histology CNN and CNN+GCN+SNN. Hazard predictions from pathomic fusion show better stratification of intermediate-to-high risk patients compared to CNN. The stratification of low-to-intermediate risk patients are very similar to the WHO paradigm (which uses molecular subtyping). Note that low-to-intermediate risk patients are mixed up in the first 5 years. However, differences between low and intermediate hazard survival curves are statistically significant in pathomic fusion according to Figure 9. Overall, pathomic fusion performs better in stratifying patient survival outcomes.



(a) Subjective Grade

(b) Molecular Subtype

(c) Histology CNN

(d) Pathomic Fusion

Figure 15: Kaplan-Meier plots for GBMLGG.

6.2 Grade Classification for TCGA-GBMLGG

We evaluate our models using using Area Under the Curve (AUC), Average Precision (AP), F1-Score (microaveraged across all classes), F1-Score with Grade IV class.

6.2.1 Using Original Data Splits for Grade Classification

In Figure 16, pathomic fusion (CNN+GCN+SNN) achieves the highest F1 of **0.731** and F1 Grade IV of **0.914** compared to other unimodal and bimodal networks. The AUC between CNN+SNN (0.905) and CNN+GCN+SNN (0.904) are very close, which corroborates the results from the original paper in Figure 17. Additionally, the AP is slightly higher in CNN+SNN, mirroring the findings in the paper.

	AUC	AP	F1	F1 Grade IV
omic	0.8522 ± 0.012	0.7287 ± 0.018	0.6503 ± 0.016	0.8541 ± 0.018
omicomic_fusion	0.8521 ± 0.011	0.7293 ± 0.018	0.6519 ± 0.015	0.8577 ± 0.015
graph	0.8458 ± 0.008	0.7584 ± 0.013	0.6482 ± 0.013	0.8100 ± 0.019
graphgraph_fusion	0.8496 ± 0.010	0.7625 ± 0.014	0.6617 ± 0.014	0.8287 ± 0.016
path	0.8922 ± 0.010	0.8122 ± 0.018	0.7141 ± 0.021	0.8836 ± 0.013
pathpath_fusion	0.8882 ± 0.007	0.8067 ± 0.013	0.7192 ± 0.020	0.8819 ± 0.014
pathomic_fusion	0.9052 ± 0.009	0.8341 ± 0.015	0.7254 ± 0.020	0.9066 ± 0.012
graphomic_fusion	0.8960 ± 0.011	0.8146 ± 0.019	0.7169 ± 0.020	0.9026 ± 0.016
pathgraphomic_fusion	0.9041 ± 0.008	0.8262 ± 0.014	0.7311 ± 0.018	0.9142 ± 0.010

Figure 16: AUC, AP, F1 and F1 Grade IV values from our re-trained model by using the same data splits as the original paper.

By comparing Figures 16 and 17, all results differ by around **0.01** from those presented in the original paper. Note that CNN testing differs the most due to the data split issue we mention earlier.

6.2.2 Evaluation Plots with Original Data Splits

In Figure 18, SNN has the worst performance because it only uses CNV, gene mutation and chromosome deletion, whereas the grade classification is usually based on histology images. Pathomic fusion CNN+GCN+SNN shows very strong performance on Grade IV with AUC of **0.979**, which might be attributed to the added genomic features of IDH1 mutation and 1p19q codeletion. The zoom-in plots are in Figure 19. Note that Figure 8 in the original paper is mistakenly swapping the positions of Grade IV and overall.

	AUC	AP	F1	F1 Grade IV
omic	0.8529 ± 0.012	0.7294 ± 0.018	0.6518 ± 0.015	0.8567 ± 0.017
omicomic_fusion	0.8503 ± 0.012	0.7245 ± 0.019	0.6508 ± 0.018	0.8560 ± 0.017
graph	0.8493 ± 0.011	0.7635 ± 0.012	0.6651 ± 0.019	0.8493 ± 0.011
graphgraph_fusion	0.8512 ± 0.015	0.7625 ± 0.021	0.6502 ± 0.023	0.8124 ± 0.022
path	0.8826 ± 0.008	0.7930 ± 0.017	0.7166 ± 0.017	0.8732 ± 0.013
pathpath_fusion	0.8878 ± 0.007	0.8071 ± 0.013	0.7151 ± 0.021	0.8793 ± 0.015
pathomic_fusion	0.9049 ± 0.010	0.8326 ± 0.016	0.7299 ± 0.019	0.9125 ± 0.011
graphomic_fusion	0.8974 ± 0.010	0.8124 ± 0.016	0.7140 ± 0.018	0.9024 ± 0.014
pathgraphomic_fusion	0.9082 ± 0.008	0.8279 ± 0.016	0.7493 ± 0.020	0.9195 ± 0.014

Figure 17: AUC, AP, F1 and F1 Grade IV values copied from the original paper.

6.3 Survival Analysis for TCGA-KIRC

We use the same networks and evaluation methods mentioned for GBMLGG to conduct the survival analysis for CCRCC. We have rewritten scripts for training and testing of CCRCC. Although these scripts are not provided in the original resources, we still achieved very similar results after multiple experiments by adjusting different training hyperparameter and fine-tuning.

The output hazard scores are divided into 25-50-75-100 percentiles to match Fuhrman Grades from G1 to G4. The remaining parts are similar to GBMLGG.

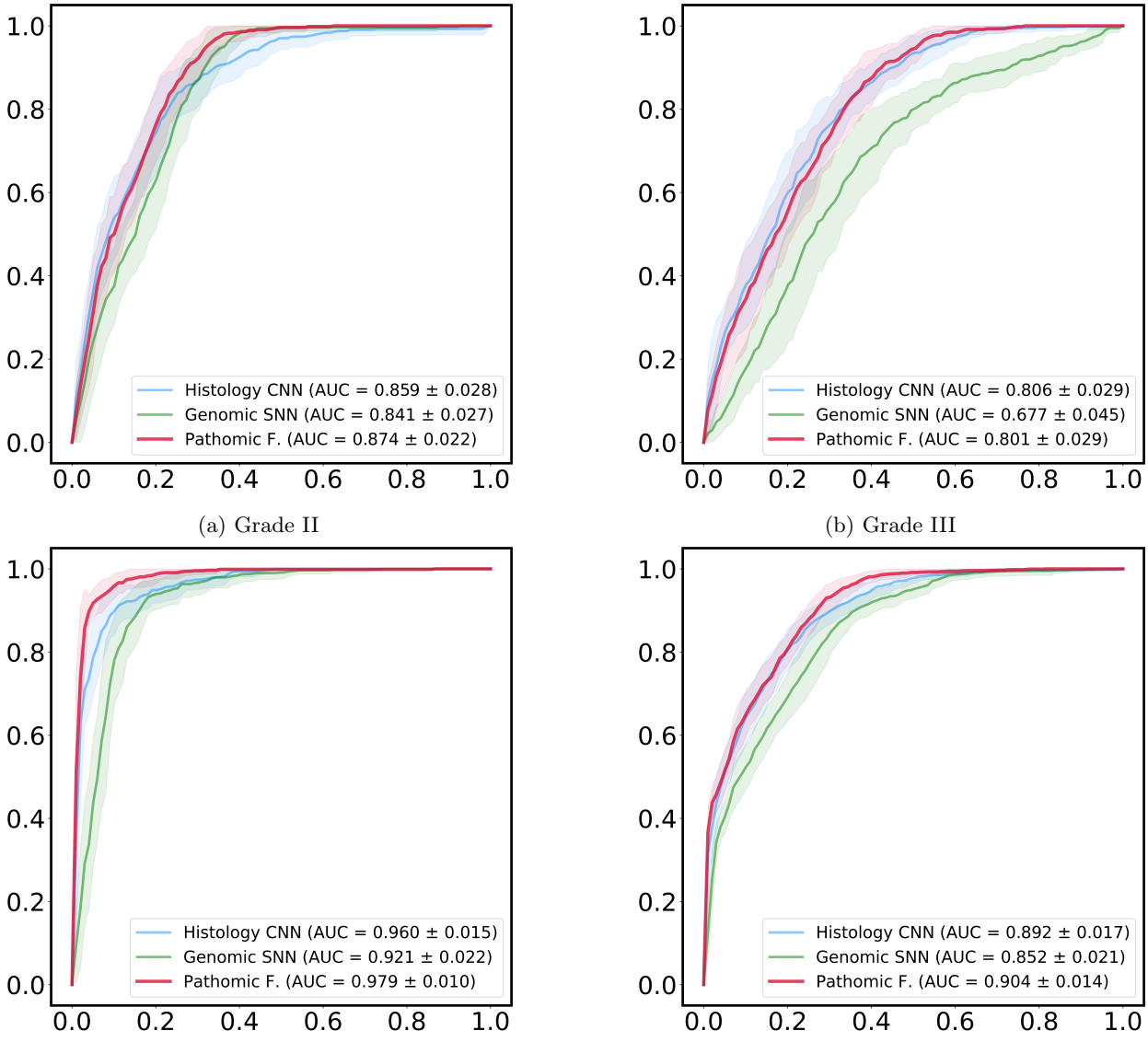
6.3.1 Using Original Data Splits for Survival Analysis

In Figure 20, both CNN+GCN+SNN (c-index of **0.713**) and CNN+SNN perform well with higher c-index values compared to other models, and their c-index values are very similar, differing by only 0.004. This indicates that the fusion of GCN does not add a significant improvement. Moreover, CNN+GCN+SNN has better statistical significance in stratifying patients into mid and high risk. Similarly to GBMLGG, the network ensembling such as CNN+CNN and SNN+SNN does not bring improvement in c-index values and stratification.

By comparing Figures 21 and 20, all results differ by no more than **0.01** from those presented in the original paper. The difference in CNN testing becomes marginal because the testing is conducted on 1024x1024 ROIs during training, rather than being retested on 512x512 patches.

6.3.2 Evaluation Plots with Original Data Splits

In Figure 22, pathomic fusion (CNN+GCN+SNN) performs better in stratifying longer and shorter patients because it exhibits a bimodal distribution in hazard ditribution but histology CNN exhibits a relatively flat



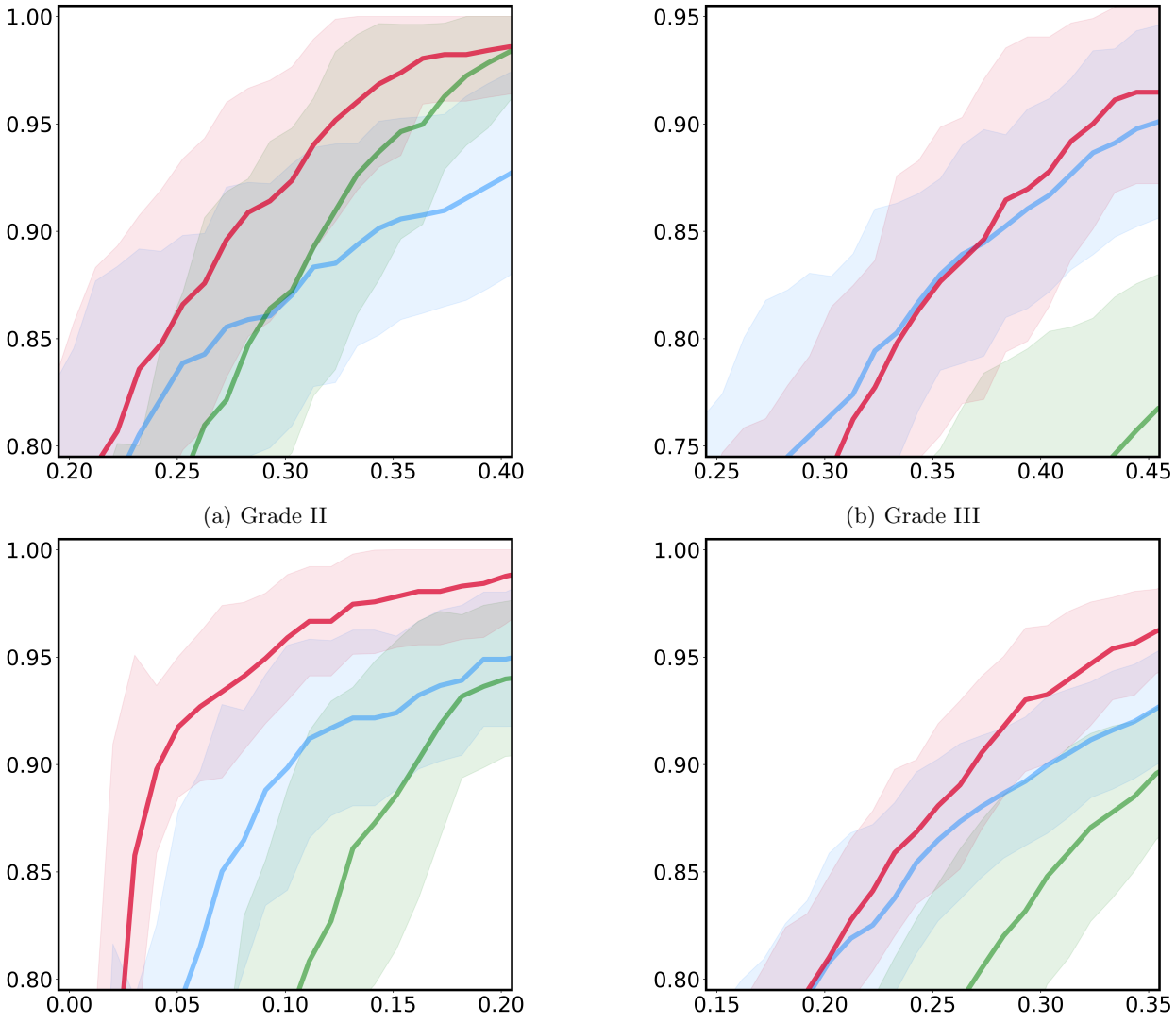
(a) Grade II

(b) Grade III

(c) Grade IV

(d) Overall

Figure 18: AUC curves for GBMLGG; y-axis represents for sensitivity, x-axis represents for specificity.



(a) Grade II

(b) Grade III

(c) Grade IV

(d) Overall

Figure 19: Zoom-in AUC curves for GBMLGG; zoom-in is based on Figure 18.

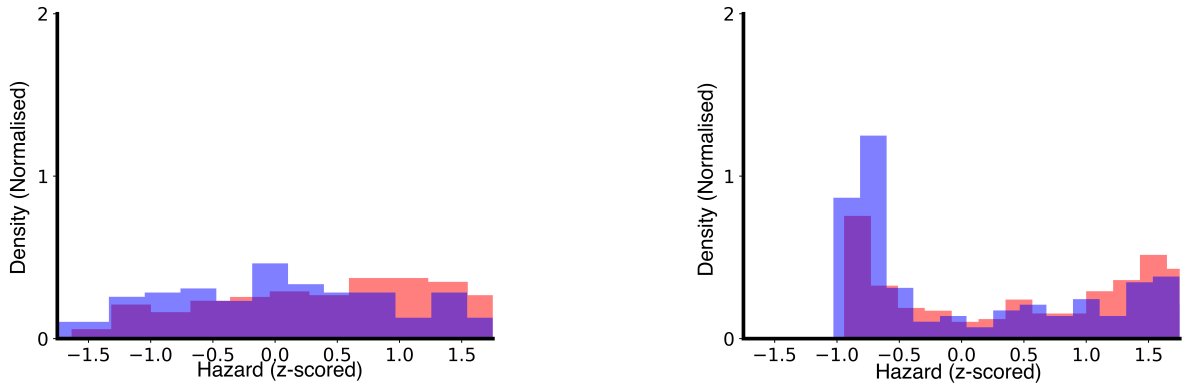
	C-Index	P-Value (<50% vs. >50%)	P-Value (<25% vs. 25-50%)	P-Value (25-50% vs. 50-75%)	P-Value (50-75% vs. >75%)
Cox (Age+Gender)	0.6300 ± 0.024	1.268123e-16	1.081791e-01	0.000012	3.595540e-01
Cox (Grade)	0.6754 ± 0.036	4.418806e-17	1.249698e-07	0.000452	5.130931e-01
Genomic SNN	0.6834 ± 0.027	9.907478e-21	6.770793e-01	0.000226	3.006669e-10
Genomic (SNN + SNN)	0.6782 ± 0.029	1.354063e-17	9.003998e-01	0.012393	2.866140e-12
Histology GCN	0.6400 ± 0.033	6.659951e-07	4.558856e-01	0.007963	4.576400e-01
Histology (GCN + GCN)	0.6386 ± 0.035	3.085023e-06	6.596622e-01	0.107347	5.115435e-03
Histology CNN	0.6627 ± 0.028	1.684364e-13	5.165860e-01	0.015833	8.158234e-07
Histology (CNN + CNN)	0.6725 ± 0.024	1.188256e-13	2.833343e-02	0.069030	3.160776e-06
Pathomic F. (CNN+SNN)	0.7167 ± 0.030	4.865127e-24	2.789467e-01	0.000769	3.084880e-12
Pathomic F. (GCN+SNN)	0.6904 ± 0.028	2.539264e-22	7.375665e-01	0.000243	4.141419e-12
Pathomic F. (CNN+GCN+SNN)	0.7125 ± 0.032	2.117521e-24	2.735330e-01	0.001857	4.325429e-13

Figure 20: KIRC; the c-index values and p-values from our re-trained model by using the same data splits as the original paper.

	C-Index	P-Value (<50% vs. >50%)	P-Value (<25% vs. 25-50%)	P-Value (25-50% vs. 50-75%)	P-Value (50-75% vs. >75%)
Cox (Age+Gender)	0.6300 ± 0.024	1.268123e-16	1.081791e-01	0.000012	3.595540e-01
Cox (Grade)	0.6754 ± 0.036	4.418806e-17	1.249698e-07	0.000452	5.130931e-01
Genomic SNN	0.6843 ± 0.025	2.896933e-18	6.182972e-01	0.015148	3.034575e-15
Genomic (SNN + SNN)	0.6783 ± 0.027	4.605888e-19	4.157303e-01	0.022354	4.342929e-15
Histology GCN	0.6482 ± 0.031	4.227995e-02	1.184032e-02	0.651703	1.439640e-01
Histology (GCN + GCN)	0.6431 ± 0.027	1.454192e-02	8.767505e-02	0.467088	5.067014e-01
Histology CNN	0.6706 ± 0.023	3.870304e-16	4.813033e-01	0.000174	4.194263e-04
Histology (CNN + CNN)	0.6709 ± 0.023	3.449451e-14	2.007879e-01	0.001571	8.699393e-04
Pathomic F. (CNN+SNN)	0.7188 ± 0.031	1.106133e-27	7.720058e-01	0.000007	5.768449e-12
Pathomic F. (GCN+SNN)	0.6853 ± 0.024	8.841046e-19	4.800574e-01	0.012675	7.066381e-16
Pathomic F. (CNN+GCN+SNN)	0.7203 ± 0.028	2.477594e-24	8.655726e-02	0.009373	1.078824e-14

Figure 21: KIRC; the c-index values and p-values copied from the original paper.

distribution. In Figure 23, the stratification between low and high risk is more obvious in pathomic fusion.

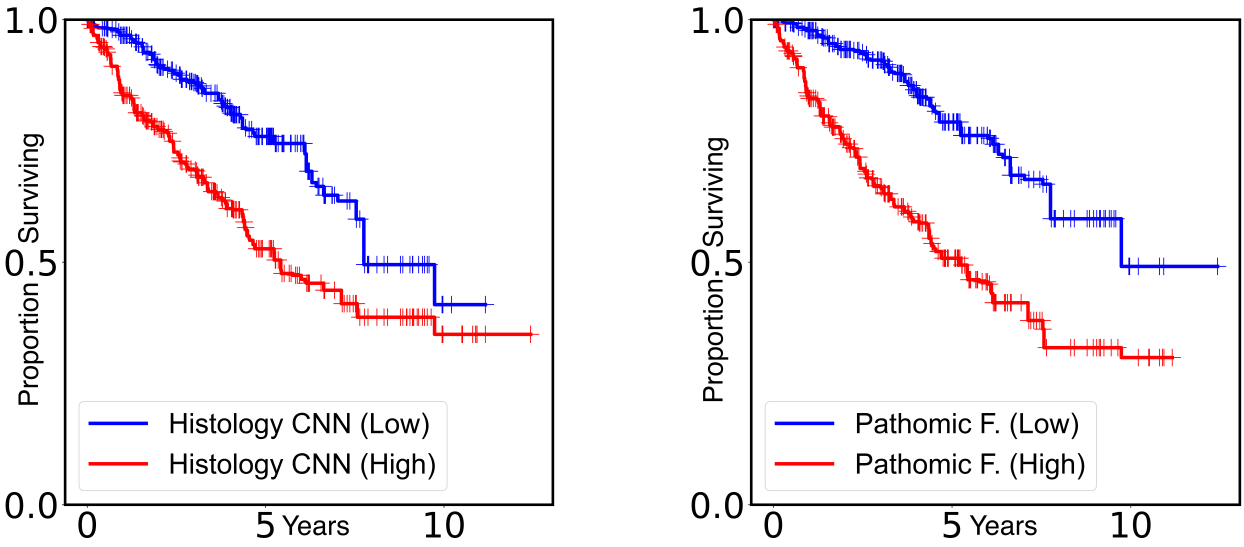


(a) Histology CNN

(b) Pathomic Fusion

Figure 22: KIRC hazard distributions; red for patient survival less then 3.5 years, blue for patient survival more than 3.5 years.

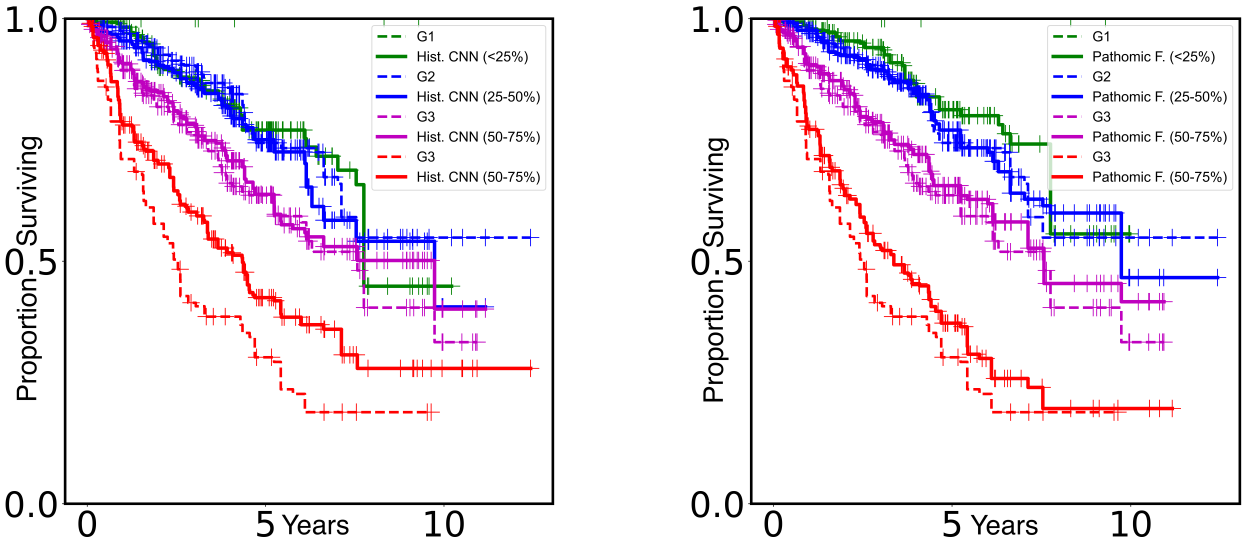
In Figure 24, pathomic fusion performs the best in disentangling each category, and corroborates with the Fuhrham Grading System.



(a) Histology CNN

(b) Pathomic Fusion

Figure 23: Kaplan-Meier curves for KIRC.



(a) Histology CNN

(b) Pathomic Fusion

Figure 24: Kaplan-Meier curves for KIRC, overlayed with grades from G1 to G4.

7 Data Visualisation

Word Count: 1095

The original code does not cover data visualisation and multimodal interpretability, and the inference code has not been released. Therefore, we rewrite the entire visualisation code to analyse feature importance attribution. We provide global explanation of genomic features across all patients for each molecular subtype and offer local explanations of histology images, cell graphs, and genomic modalities for individual patients.

7.1 Local Explanation for Each Modality

7.1.1 Grad-CAM for CNN

We implement Grad-CAM for local explanation of histology image. In [25],

Definition 3. *(Grad-CAM) A gradient-based localisation technique used to produce visual explanations in image classification, in which neurons whose gradients have positive influence on a class of interest are used to produce a coarse heatmap.*

The mathematical computation for Grad-CAM is,

$$\begin{aligned} \alpha_k^c &= \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \quad ; \\ L_{\text{Grad-CAM}}^c &= \text{ReLU}\underbrace{\left(\sum_k \alpha_k^c A^k\right)}_{\text{linear combination}} . \end{aligned} \tag{13}$$

The first equation of (13) computes the importance weights of the feature map k for the class c by averaging the gradients of the class score y^c with respect to the activation A_{ij}^k over all spatial locations. Then (13) computes the Grad-CAM by linearly combining the feature maps using the aforementioned importance weights and applying the ReLU activation.

Before applying the Grad-CAM algorithm 3, we need to register the forward and backward hooks on the target layers we have chosen for backpropogation purposes. Algorithm 3 describes the key steps.

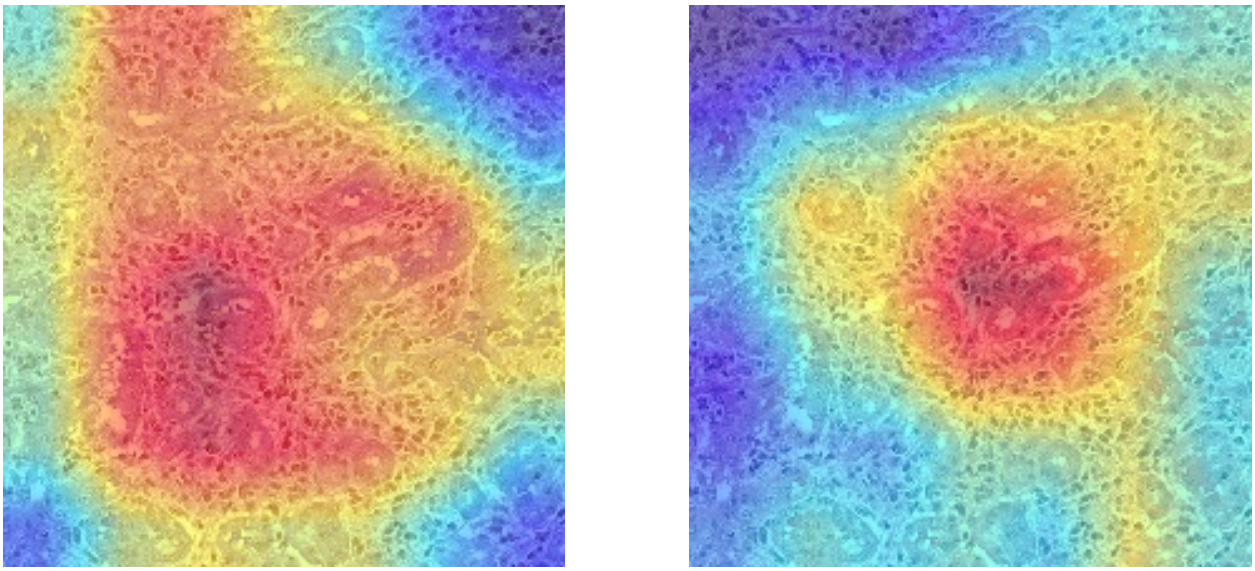
Initially, we let features pass through the classifier, and heatmaps are generated on relatively small patches

Algorithm 3 Grad-CAM Algorithm

1. target_class = model_output.argmax(dim=1).item() ▷ Use the class with the highest hazard score from the model's output
2. model_output.backward(gradient=one_hot_output) ▷ Compute the gradients of the target class score with respect to activation (feature map)
3. weights = torch.mean(self.gradients, dim=(2, 3), keepdim=True) ▷ Average graidents across the spatial dimensions to obtain weights
4. cam = torch.sum(weights * self.activations, dim=1, keepdim=True) ▷ Obtain weighted sum of feature maps for CAM, second line of (13)
5. cam = F.relu(cam) ▷ Add the ReLU activation, second line of (13)

with size of 224x224 in Figure 25. However, the performance is relatively poor due to unclear localisation of tumor cellularity. Therefore, we decide to use the last layer of the feature encoder without passing through the classifier, such as features.36 for VGG19 and layer.4 for ResNet-50. In Figure 26, we obtain better results on patches with size of 1024x1024 without passing through a classifier, and the importance attributions are easily to be observed.

Apart from using the standard Grad-CAM, we also attempt its variations such as Grad-CAM++ and



(a) vgg19 with classifier

(b) resnet50 with classifier

Figure 25: Grad-CAM heatmaps by passing through the classifier, and the patch size is 224x224; patient TCGA-06-0174 with IDHwt ATC.

LayerCAM in the section of Further Improvements.

7.1.2 Integrated Gradients for SNN

For Genomic SNN, we use the Integrated Gradients (IG) method. In [30],

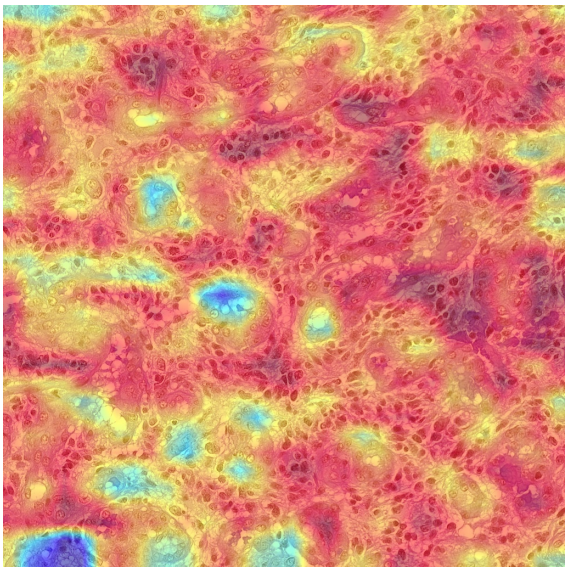
Definition 4. *(Integrated Graidents method) A gradient-based feature attribution method that attributes the prediction of deep networks to their inputs. IG calculates the gradients of the input tensor x across different scales against a baseline x_i, and uses the Gauss-Legendre quadrature to approximate the integral of gradients.*

The baseline x_i is zero scaled in our case, and the mathematical expression is,

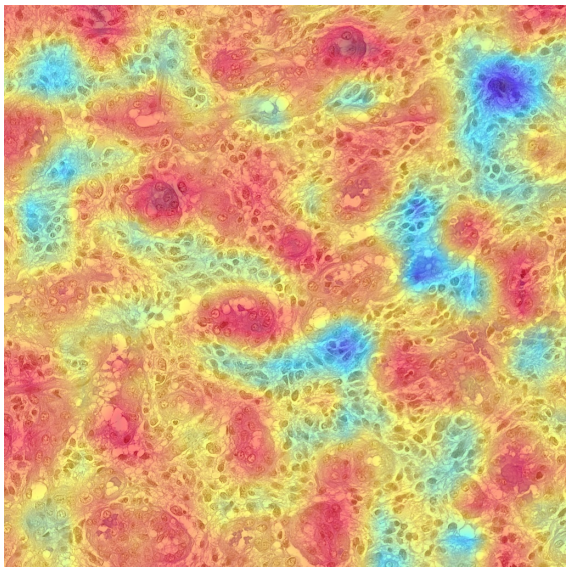
$$\text{IntegratedGrads}_{\,i}(x) ::= (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial F\left(x' + \alpha \times (x - x')\right)}{\partial x_i} d\alpha. \tag{14}$$

Equation 14 is along the i^{th} dimension for an input x and baseline x', and $\frac{\partial F(x)}{\partial x_i}$ is the gradient of $F(x)$ along the i^{th} dimension. In our case, we use trapezoidal rule to approximate the integral. It is also possible to use quadrature integration for better accuracy. Algorithm 4 describes the key steps.

We use the SHapley Additive exPlanations (SHAP) plot to reveal the importance of genomic features



(a) vgg19 without classifier



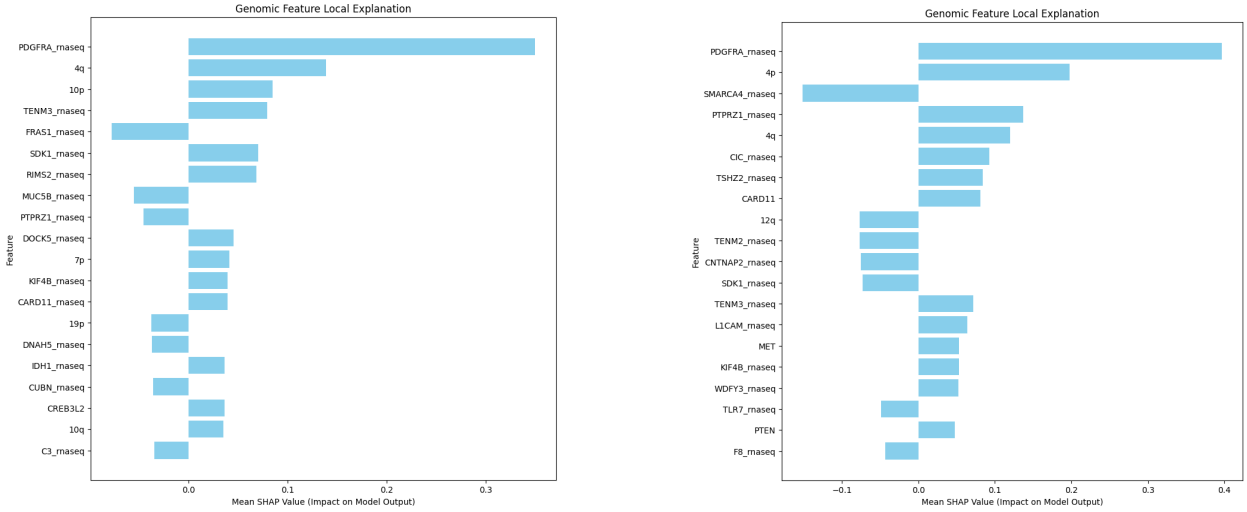
(b) resnet50 without classifier

Figure 26: Grad-CAM heatmaps by using the last layer of feature encoder as the target layer without passing through the classifier, and the patch size is 1024x1024; patient TCGA-06-0174 with IDHwt ATC.

Algorithm 4 Integrated Algorithm

1. scaled_inputs = [baseline + (float(i)/steps) × (input - baseline) for i in range(steps + 1)] ▷ Scale the inputs for IG with a specific number of steps
2. gradients = torch.autograd.grad(outputs=predictions, inputs=scaled_inputs) ▷ Generate gradients for the hazard predictions to get $\frac{\partial F(x)}{\partial x_i}$
3. gradients = (gradients[:-1] + gradients[1:]) / 2.0
4. average_gradients = gradients.mean(dim=0)
5. integrated_gradients = (input - baseline) × average_gradients ▷ Perform trapezoidal rule for integration

for predicting outcomes in individual patients. In Figure 27, we list two examples to show local explanation of genomic features. A positive value on the x-axis indicates that the presence of that genomic subtype can prolong survival time, while a negative value indicates the opposite.



(a) TCGA-ID-06-0174 (b) TCGA-ID-26-1442

Figure 27: Local explanation for genomic data for individual patients.

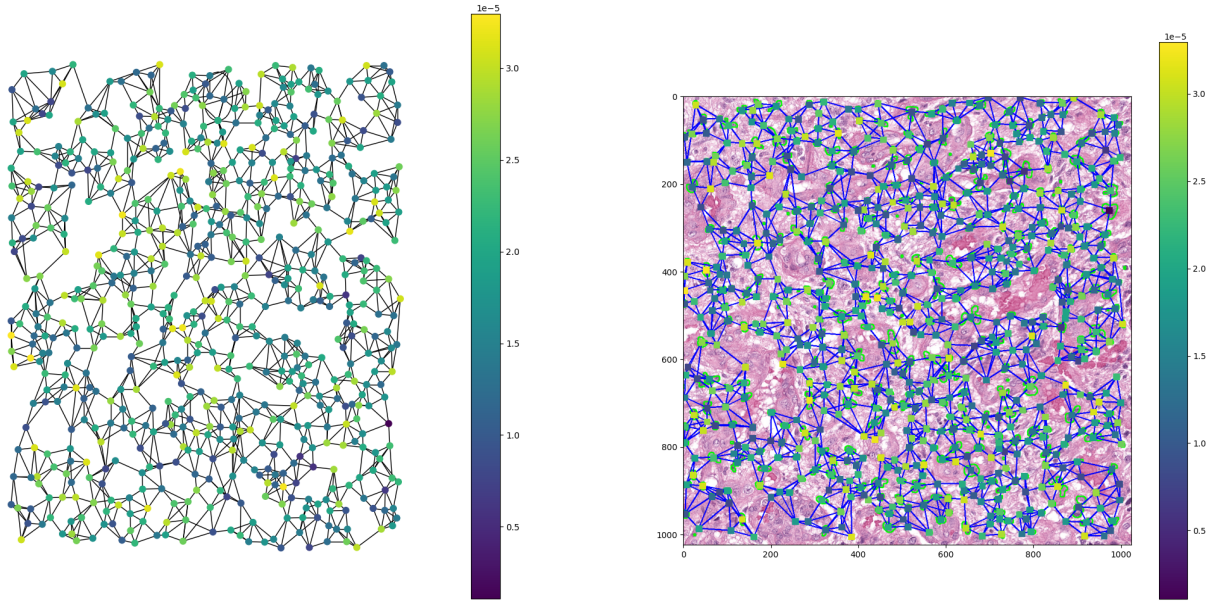
7.1.3 Integrated Gradients for GCN

We use the IG method to generate local explanation for GCN, and modify the code to scale the input graph features accordingly. We extract the features from each graph node (nodes are considered as the batch dimension), and do the same process as before to scale each node by the number of integral approximation steps. When drawing the attribution marks, it is necessary to take the mean of the attributions because there are multiple attributions for each node, rather than a single attribution per node.

```
node_attributions = integrated_grads.mean(axis=1)
```

In Figure 28, we generate a cell graph heatmap and overlay it with the image, noticing that glial cells exist between the microvasculature.

If we do not map the graph nodes to a specific H&E image, the location of centroids in the output will be random, resulting in a complicated edge connecting scenario as shown in Figure 29. The final output is Figure 30. Overall, the cell graph might uncover salient regions that are not recognised in the histology image for risk prediction, as its attributions refer to specific atypical cells rather than pixel regions.



(a) Only with centroids

(b) Overlayed with H&E Image

Figure 28: Local explanation for Cell Graph GCN; attribution colour corresponds to low (purple) vs. high (yellow) feature value; patient TCGA-ID-06-0174.

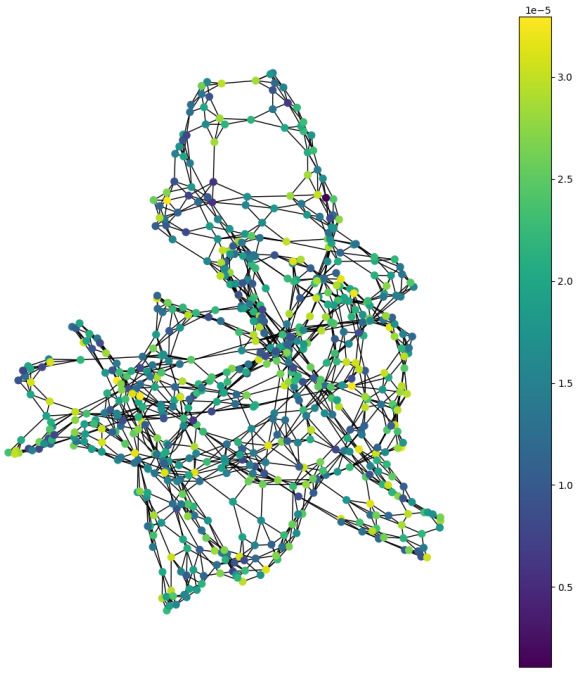


Figure 29: Failure example without fixing nodes with respect to the H&E image; attribution colour corresponds to low (purple) vs. high (yellow) feature value; patient TCGA-ID-06-0174.

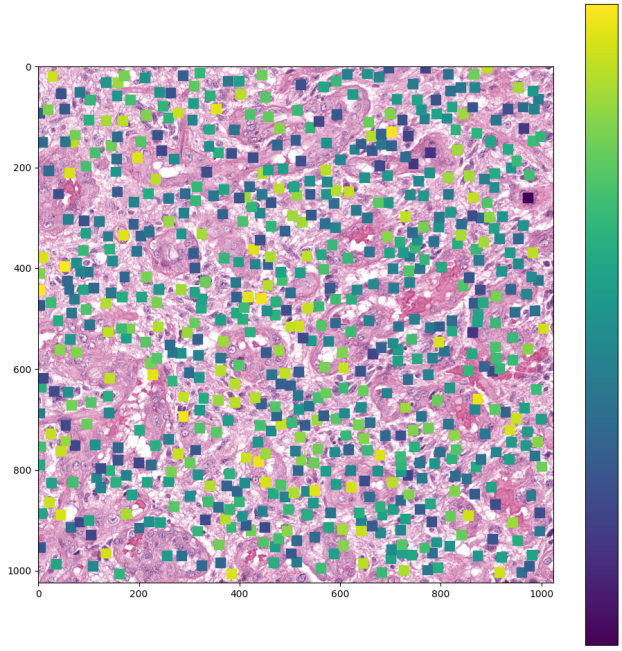


Figure 30: Final output of local explanation for cell graph GCN by removing edges; attribution colour corresponds to low (purple) vs. high (yellow) feature value; patient TCGA-ID-06-0174

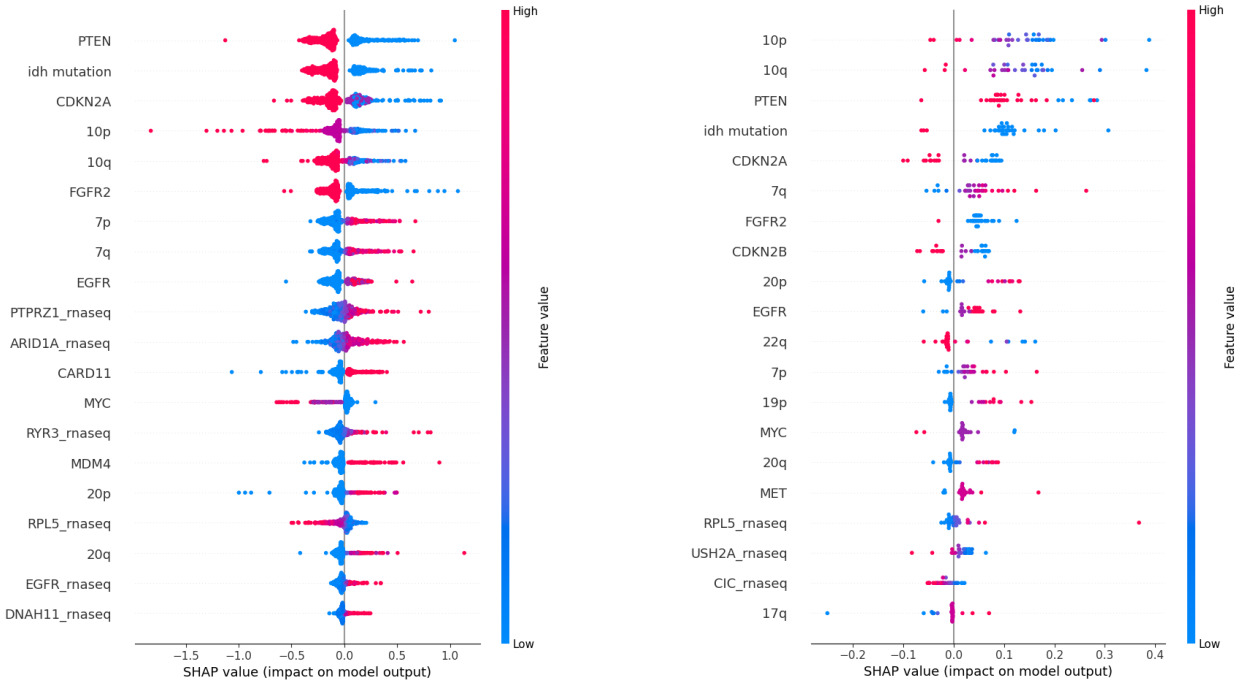
7.2 Global Explanation for Genomic Modality

We explore the global explanation of the top 20 genomic features using the IG method. We want to find out how feature attribution shifts when conditioning on morphological features. The first SHAP plot is generated by using genomic SNN, and the second SHAP plot is generated by using pathomic fusion (CNN+SNN or CNN+GCN+SNN). The experimental details and code are not covered in the original paper. Therefore, we create our own data splits for IG, and adapt the fusion code to resolve dimensionality issues.

```
o1 = o1.view(-1, 32)
o1 = torch.cat((o1, torch.ones(o1.shape[0], 1, device=device)), 1)
```

7.2.1 Global Explanation without Molecular Subtype

We conduct experiments across all the patients without dividing them into specfic molecular subtypes. In Figure 31, we can corroborate important biomarkers, such as IDH mutation, with increased risk. We also observe several signature genes such as PTEN, CDKN2A and EGFR. We also test global explanation again with pathomic fusion (CNN+GCN+SNN). The importance of 10p and 10q increases, while the importance of PTEN decreases. Note that the points are sparser in the fusion model because we use data splits that ignore missing values, as mentioned in the Data Acquisition section. Additionally, we choose to use a smaller sample size due to limited CUDA memory.



(a) Genomic SNN

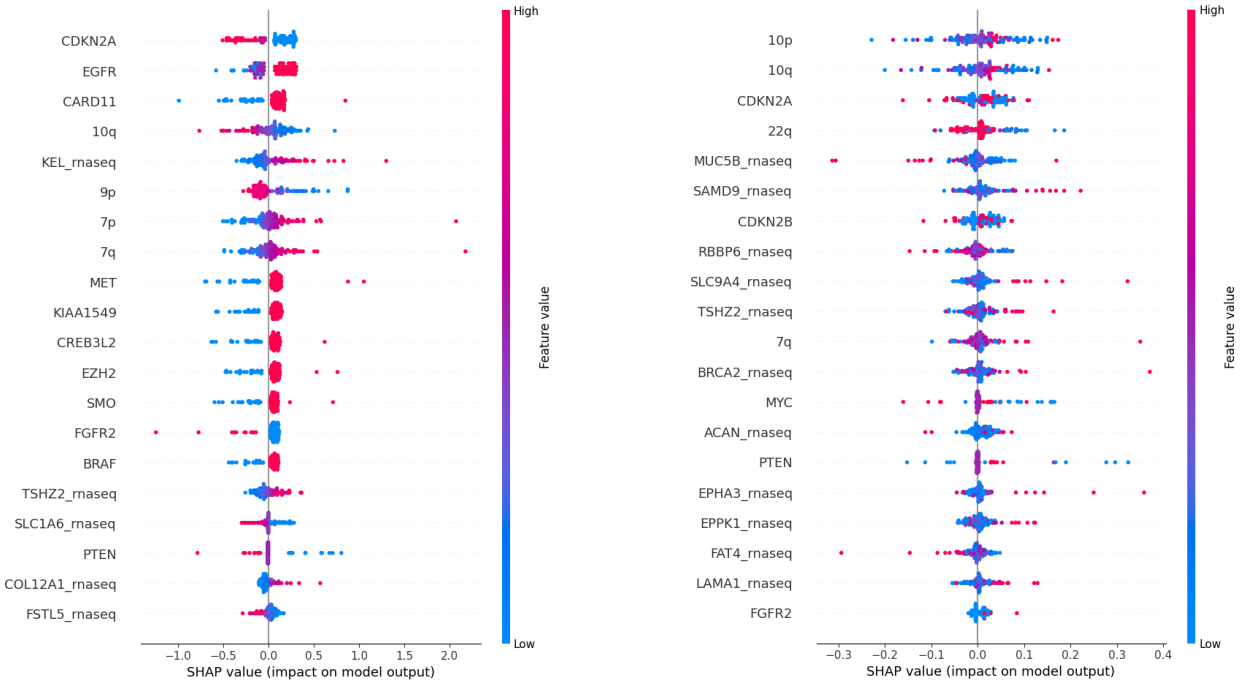
(b) Pathomic Fusion (CNN+GCN+SNN)

Figure 31: Global explanation of genomic features for all patients; comparison between SNN and CNN+GCN+SNN; attribution colour corresponds to low (blue) vs. high (red) feature value, and attribution direction corresponds to how the gene feature value contributes to low risk (left) vs. high risk (right).

7.2.2 Global Explanation with Molecular Subtype

We conduct experiments for each molecular subtype in GBMLGG (IDHwt, IDHmut and ODG). In Figure 32 of IDHwt ATC, EGFR feature importance decreases with pathomic fusion, it might show the fact that EGFR is not a strong therapeutic target in glioblastoma treatment [32]. In the meantime, 10p and 10q importance increases, and they play a crucial role in regulating cell growth.

In Figure 33 of IDHmut ATC, CDKN2A feature importance decreases with pathomic fusion, while TUBA3C



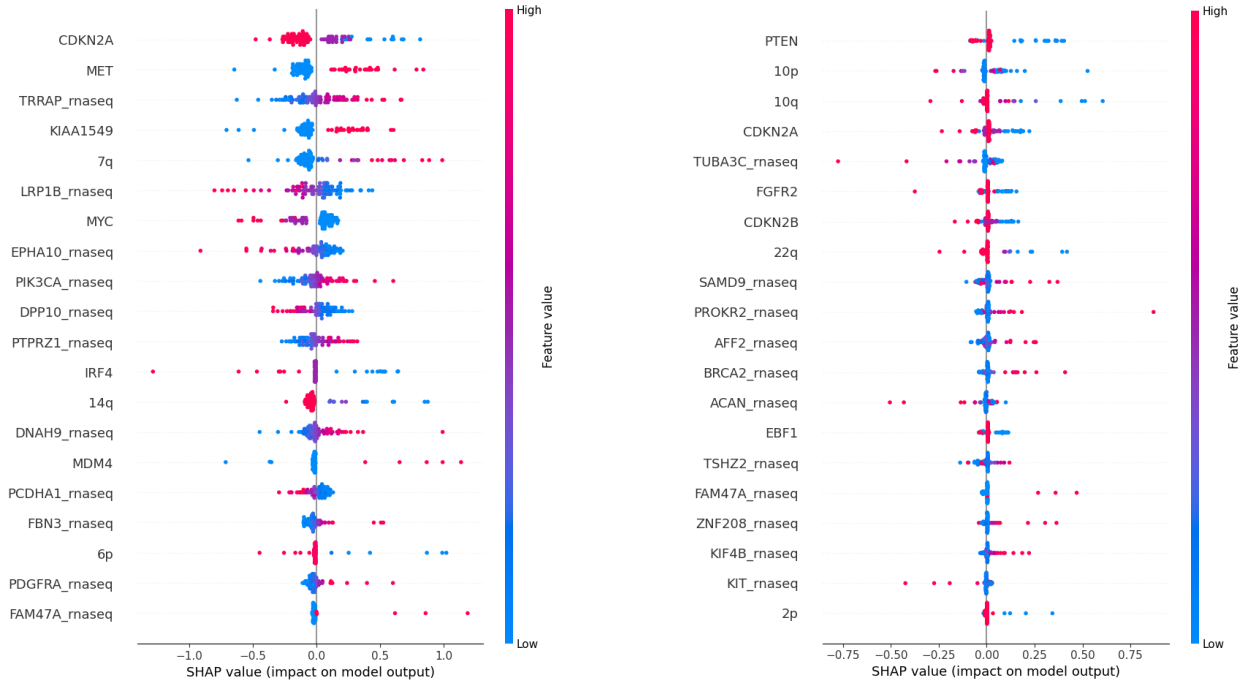
(a) Genomic SNN

(b) Pathomic Fusion (CNN+SNN)

Figure 32: Integrated Graident Attribution: global explanation for top 20 genomic features in IDHwt Astrocytoma; attribution colour corresponds to low (blue) vs. high (red) feature value, and attribution direction corresponds to how the gene feature value contributes to low risk (left) vs. high risk (right).

importance increases, indicting that TUBA3C has strong staining in some gliomas and ovarian cancers [3].

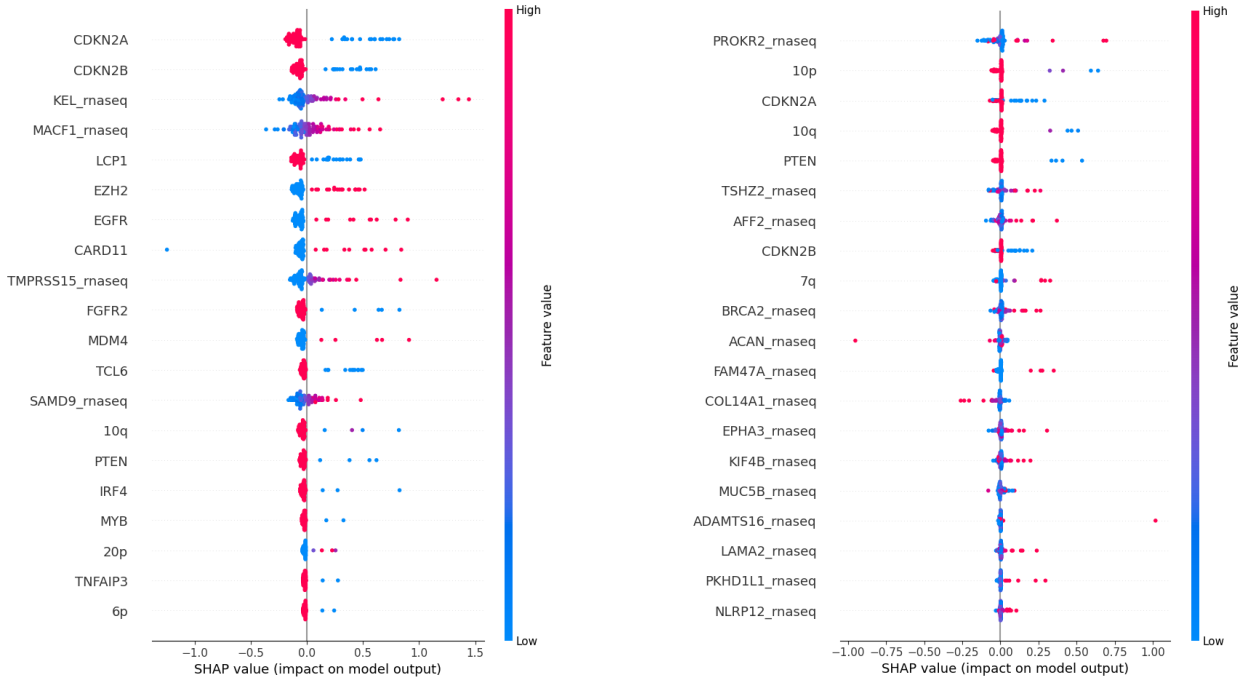
In Figure 34 of ODG, CDKN2B feature importance decreases, while PROKR2 importance increases, suggesting its role as a potential diagnostic marker for gliomas.



(a) Genomic SNN

(b) Pathomic Fusion (CNN+SNN)

Figure 33: Integrated Graident Attribution: global explanation for top 20 genomic features in IDHmut Astrocytoma; attribution colour corresponds to low (blue) vs. high (red) feature value, and attribution direction corresponds to how the gene feature value contributes to low risk (left) vs. high risk (right).



(a) Genomic SNN

(b) Pathomic Fusion (CNN+SNN)

Figure 34: Integrated Graident Attribution: global explanation for top 20 genomic features in Oligodendroglioma; attribution colour corresponds to low (blue) vs. high (red) feature value, and attribution direction corresponds to how the gene feature value contributes to low risk (left) vs. high risk (right).

8 Further Improvements and Discussion

Word Count: 1146

8.1 Import Pretrained Models from Torchvision

The original code explicitly lists the configurations of the vgg layer.

```
for v in cfg:
conv2d = nn.Conv2d(in_channels, v, kernel_size=3, padding=1)
if batch_norm:
    layers += [conv2d, nn.BatchNorm2d(v), nn.ReLU(inplace=True)]
else:
    layers += [conv2d, nn.ReLU(inplace=True)]
in_channels = v
```

To make the PathNet code more structured, the encoder part can be replaced by importing a pretrained vgg19/vgg19bn model directly from 'torchvision.models'. After testing on this, the imported vgg19bn model already includes the batch normalization step, providing a clearer layer architecture. As a result, the time required to run one epoch for the histology CNN has been reduced by around 15 percent. In addition, it is easier to choose the pretrained weights as follows,

```
models.vgg19_bn(weights=models.VGG19_BN_Weights.IMAGENET1K_V1).to(device)
```

Implementing 'torchvision.models' also enables the use of other networks, such as Inception v3 and ResNet 50, without the need to explicitly define layers.

8.2 Nuclei Segmentation by CellViT

In the original paper, the nuclei segmentation used for GCNs and graph cell reconstruction is generated by the conditional GAN, which was trained on a dataset of perfectly-annotated synthetic histopathology images and ground truth data with complete nuclei annotation from [30]. However, this dataset is still not large enough to determine nuclei type and conduct accurate cell classification. The training process for conditional GAN is also time-consuming due to two stages 1) generating synthetic images with masks via the cyclic GAN 2) training a large collection of synthetic and real data. Moreover, the input into conditional GAN needs to be preprocessed manually, and the WSI svs files downloaded from TCGA cannot be directly used for this model.

We have found an alternative model called CellViT for nuclei segmentation, and it is based on a vision transformer architecture. We note that the conditional GAN can solve the issue of segmenting multiple nuclei as a single entity. CellViT achieves the same outcome by using HoVer Net's postprocessing pipeline [12], which computes the gradients of the horizontal and vertical distance maps to capture transitions between nuclei boundaries and the boundary between nuclei and the background. For CellViT mechanism, an input image is transformed into a sequence of tokens, and there are skip connections at multiple encoder depth levels. Then the upsampling decoder network is used to generate precise nuclei instance segmentations. The encoder-decoder structure follows an U-Net architecture in Figure 35. According to [14], CellViT achieves a F1-detection score of 0.83 on the PanNuke dataset. In addition, CellViT incorporates the preprocessing and postprocessing steps all in one. The model completes segmentation and detection of different nucleus types

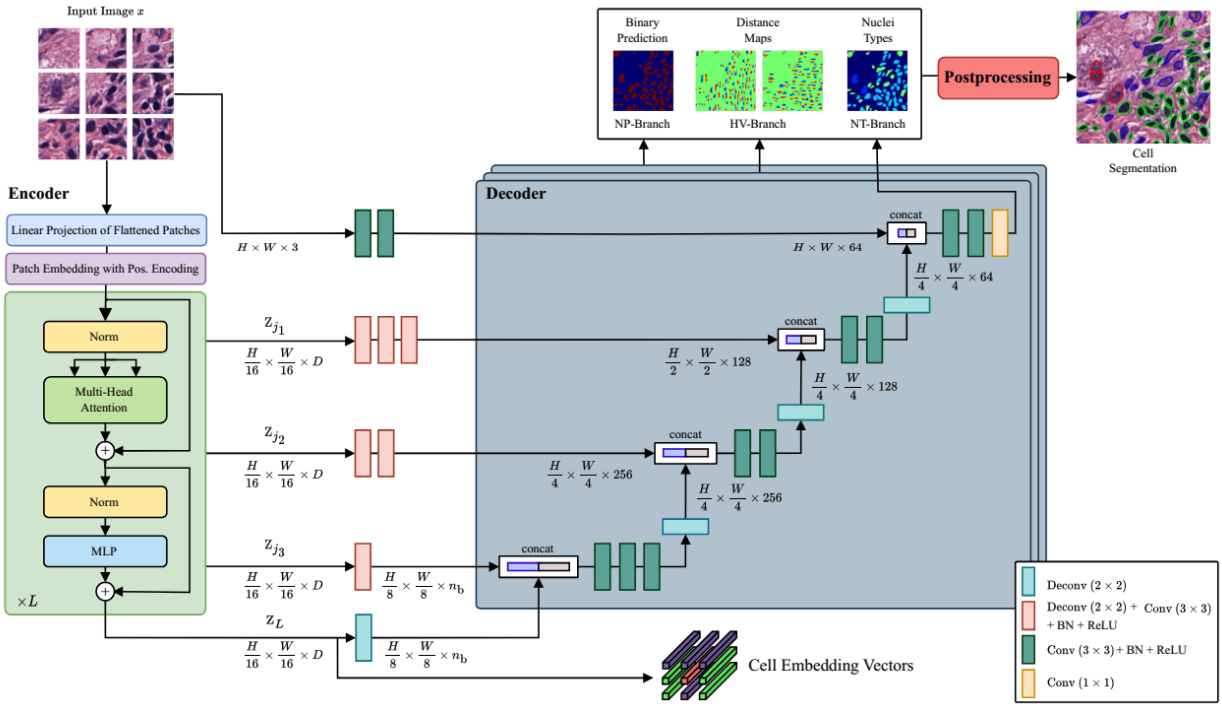


Figure 35: Architecture of CellViT from [14]

within a few minutes. We have run CellViT on several individual patients' slides from TCGA-GBMLGG in Figures 36 and 37. In previous GCNs, we segment all cells, regardless of their types, to construct the cell graph and assign cell features. With CellViT, we can segment only neoplastic cells, which might improve accuracy (higher c-index) by reducing noise from irrelevant cells and simplifying cell graph structure.

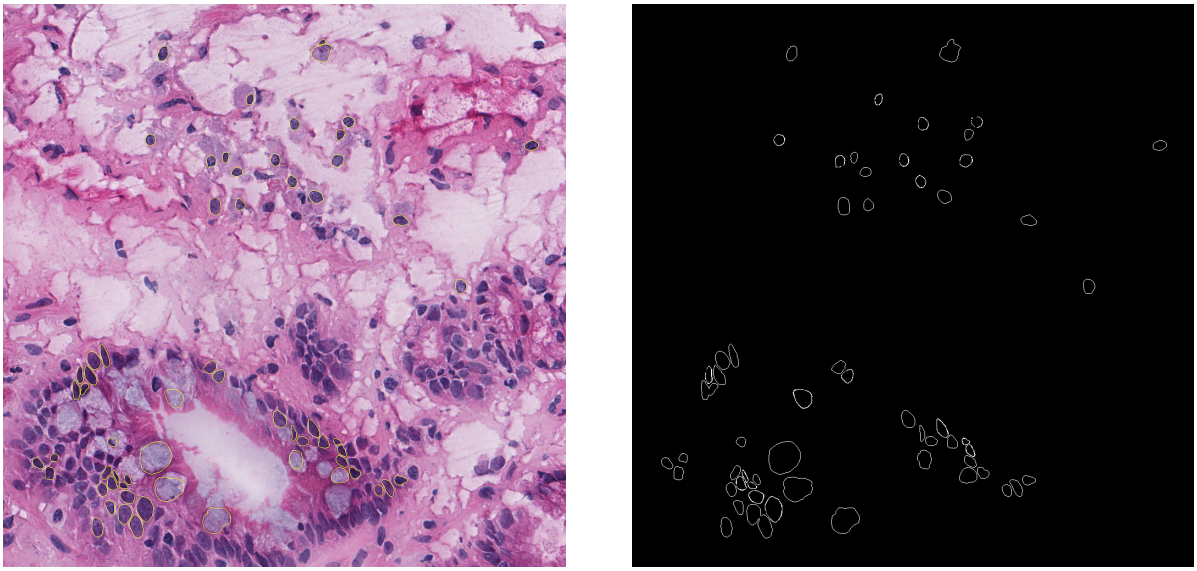


Figure 36: Nuclei Segmentation with CellViT; the highlighted nucleus are neoplastic.

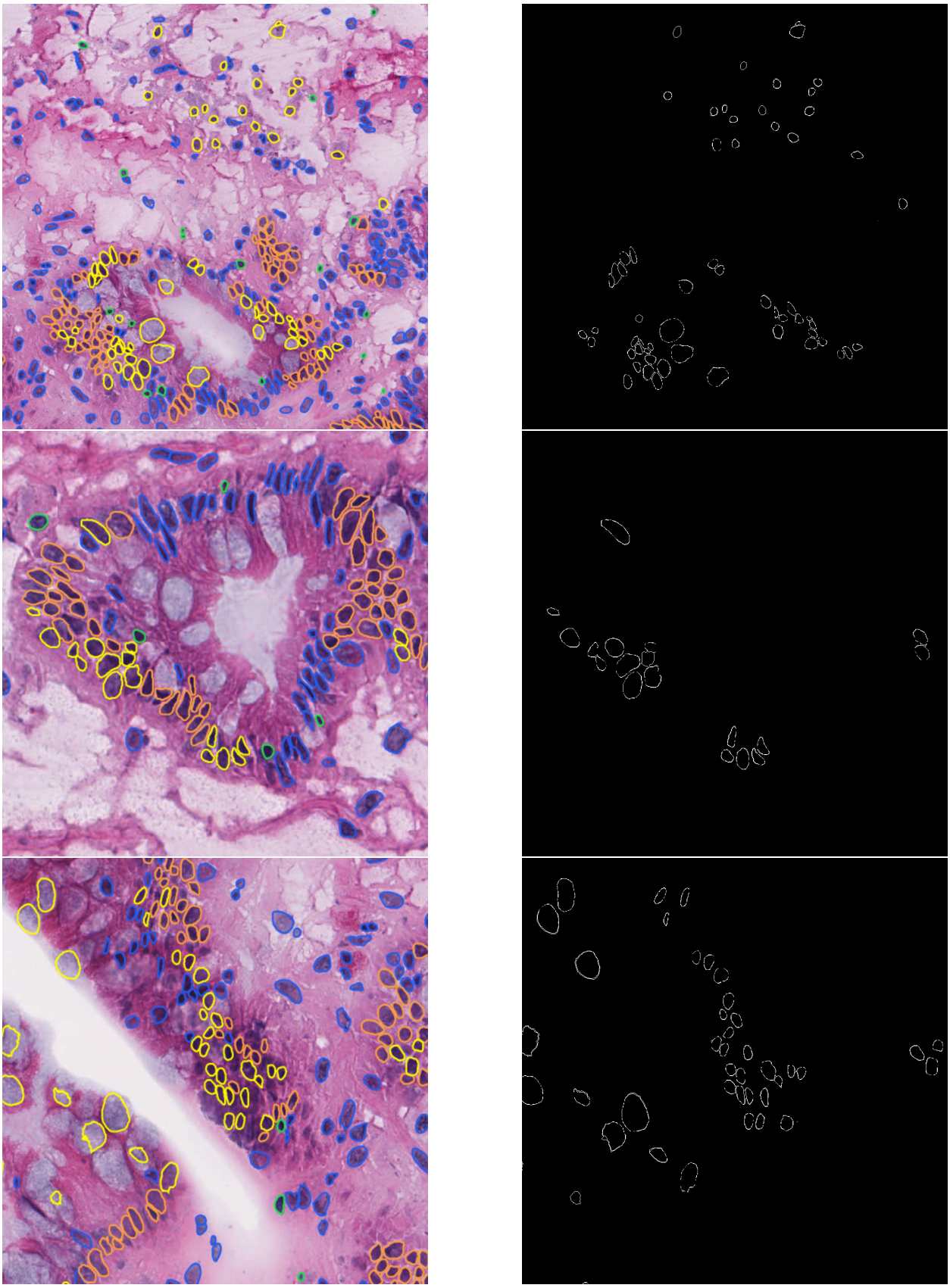


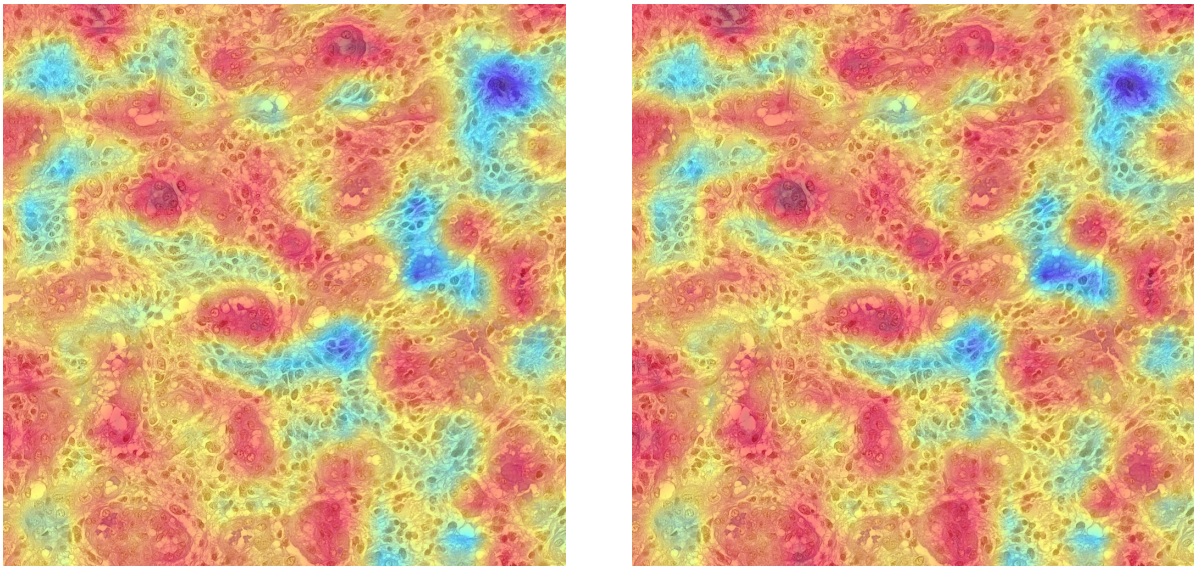
Figure 37: Nuclei Segmentation with CellViT on examples TCGA-V5-A7RE-11A-01-TS1 and TCGA-06-0174-01Z-00-DX3; yellow contours are neoplastic, green contours are epithelial, and purple contours are connective; we only segment neoplastic ones.

8.3 Variations of Grad-CAM

There are different variations of Grad-CAM, and they might help us uncover the missing details in histology images.

8.3.1 Grad-CAM++

Grad-CAM++ is very similar to Grad-CAM, but uses the second-order gradient for the class score. We change the gradient part in (13) from $\frac{\partial y^c}{\partial A_{ij}^k}$ to $\frac{\partial^2 y^c}{\partial \left(A_{ij}^k\right)^2}$. Grad-CAM++ might outperform Grad-CAM in terms of better localisation of objects and explaining occurrences of multiple objects of a class in a single image [7].



(a) Grad-CAM (b) Grad-CAM++

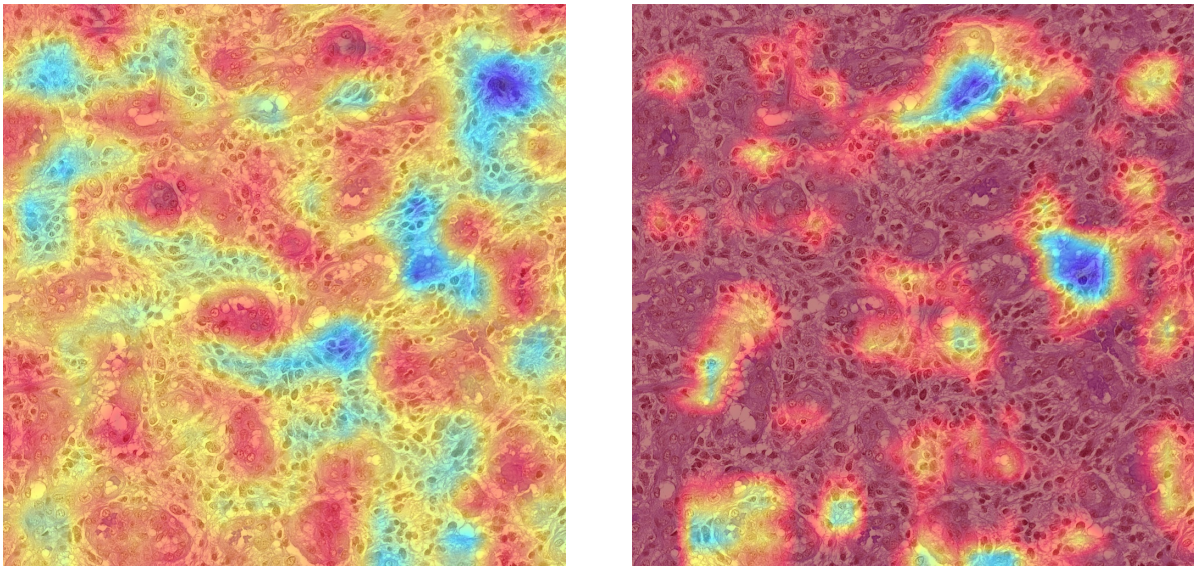
Figure 38: Comparison between Grad-CAM and Grad-CAM++.

8.3.2 EigenCAM

EigenCAM is based on principal component analysis (PCA). It takes the first principle component of the 2D Activations, and there is no class discrimination/target layer, which means it focuses on the dominant patterns in the activations without considering the gradients in Figure 39. It is also a simpler computation without backpropagation.

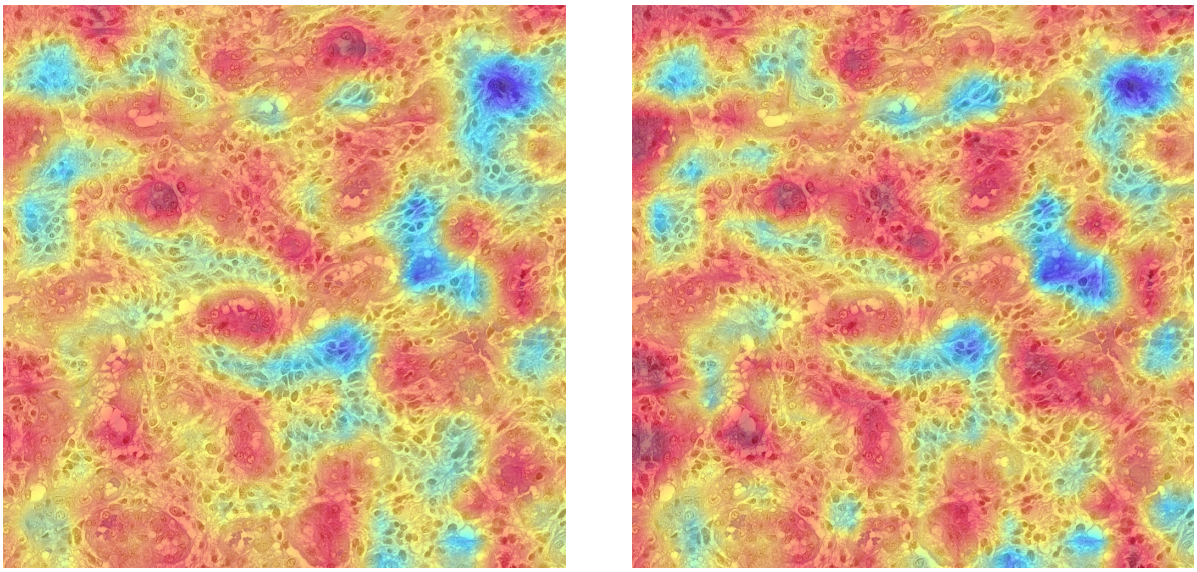
8.3.3 LayerCAM

LayerCAM spatially weights the activations by positive graidents, and it provides detailed visual explanations even from lower layers. We change the gradient part in (13) from $\frac{\partial y^c}{\partial A_{ij}^k}$ to $\max\left(0, \frac{\partial y^c}{\partial A_{ij}^k}\right)$. The comparison can be found in Figure 40.



(a) Grad-CAM (b) EigenCAM

Figure 39: Comparison between Grad-CAM and EigenCAM.



(a) Grad-CAM (b) LayerCAM

Figure 40: Comparison between Grad-CAM and LayerCAM.

8.4 CONCH

We have found a new way to extract feature representations that differs from previous approaches like CNNs, GCNs and SNNs. CONtrastive learning from Captions for Histopathology (CONCH) is a visual language foundation model developed using histopathology images, biomedical text, and over 1.17 million image-caption pairs via task-agnostic pretraining [22]. The training on CONCH involves (1) contrastive alignment objectives that seek to align the image and text modalities in the model's representation space (2) a captioning objective that learns to predict the caption corresponding to an image. In Figure 41, CONCH consists of an image encoder, a text encoder and a multimodal text decoder. We replace the PathNet encoder with CONCH image encoder to obtain the image embeddings/features.

By inspecting the released CONCH code, the image encoder consists of the backbone and two attentional

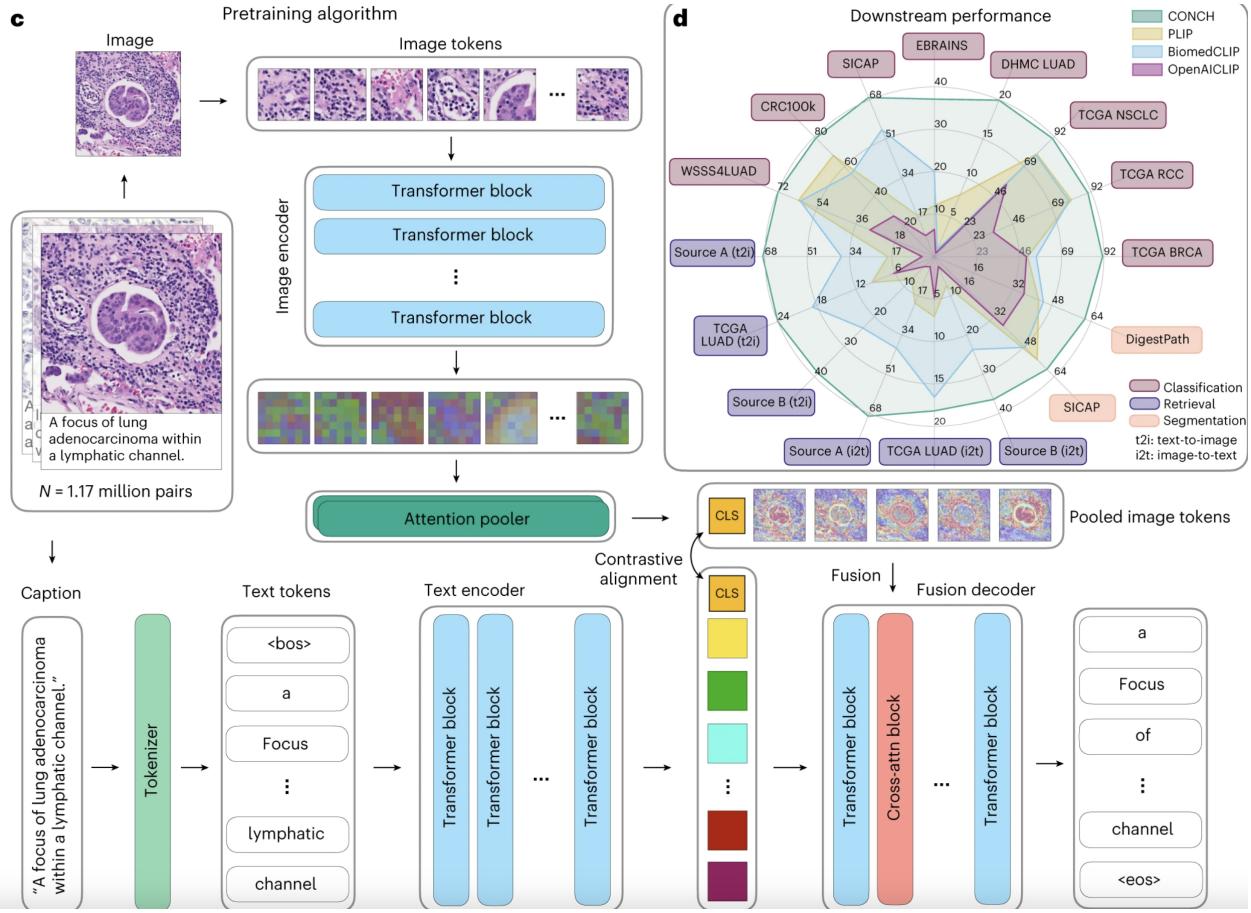


Figure 41: CONCH visual-language pretraining setup

pooler modules. The backbone follows the standard ViT-base architecture [17], which transforms images to dense feature maps in a representation space learned from data. The attention pooler modules use the last layer of the ViT backbone to compute image tokens for the global and local representation of the image. After using CONCH instead of CNN, CONCH+SNN model even generates a higher c-index score **0.831** compared to 0.822 from CONCH+GCN+SNN in Figure 42. The additional CONCH embeddings motivate us to compute the quadrilinear fusion.

	C-Index	P-Value (<50% vs. >50%)	P-Value (<33% vs. 33-66%)	P-Value (33-66% vs. >66%)
Histology CONCH	0.7742 ± 0.016	4.750643e-41	0.041872	3.418225e-31
Genomic SNN	0.8023 ± 0.017	7.880654e-53	0.769360	1.372293e-83
Histology GCN	0.7388 ± 0.020	3.194850e-26	0.565802	4.368388e-26
Pathomic F. (CONCH+SNN)	0.8309 ± 0.015	7.267651e-61	0.010480	5.082288e-73
Pathomic F. (GCN+SNN)	0.8018 ± 0.016	5.864152e-63	0.093537	1.798589e-77
Pathomic F. (CONCH+GCN+SNN)	0.8218 ± 0.013	6.298128e-58	0.113025	1.775979e-74

Figure 42: The c-index values from our re-trained model by using the new data splits with CONCH

8.5 Multilinear Fusion

With image embeddings from CONCH, we modify the original trilinear fusion code, and rewrite the code of fusion strategy and data splits to achieve multilinear (quadrilinear) fusion CNN+GCN+SNN+CONCH. The final result shows a significant improvement, with the c-index increasing from 0.823 to **0.835**.

	C-Index	P-Value (<50% vs. >50%)	P-Value (<33% vs. 33-66%)	P-Value (33-66% vs. >66%)
Cox (Age+Gender)	0.7316 ± 0.012	1.900892e-92	1.475424e-38	5.927583e-27
Cox (Subtype)	0.7595 ± 0.011	2.064634e-228	4.653368e-26	1.444895e-51
Cox (Grade)	0.7379 ± 0.013	5.998349e-224	9.568192e-23	2.941556e-66
Cox (Grade+Subtype)	0.7767 ± 0.013	5.291938e-215	1.140947e-40	5.016280e-52
Genomic SNN	0.8100 ± 0.015	5.316290e-47	1.010861e-01	7.301739e-84
Genomic (SNN + SNN)	0.8092 ± 0.015	2.749143e-51	1.530080e-01	1.113833e-83
Histology GCN	0.7247 ± 0.022	1.369501e-25	1.023542e-03	1.689381e-16
Histology (GCN + GCN)	0.7287 ± 0.020	2.861252e-30	4.127735e-04	1.918003e-15
Histology CNN	0.7743 ± 0.017	7.938201e-36	2.564271e-10	1.462746e-15
Histology (CNN + CNN)	0.7748 ± 0.017	9.584131e-37	4.815573e-09	2.767427e-16
Pathomic F. (CNN+SNN)	0.8203 ± 0.019	3.444377e-54	3.497569e-01	9.893598e-81
Pathomic F. (GCN+SNN)	0.8121 ± 0.015	5.617681e-55	5.249495e-01	1.196039e-81
Pathomic F. (CNN+GCN+SNN)	0.8233 ± 0.017	1.179099e-51	4.096507e-01	1.451836e-81
Pathomic F. (CNN+GCN+SNN+CONCH)	0.8353 ± 0.015	1.067744e-59	2.685057e-02	1.523663e-68

Figure 43: Apply CONCH image embedding to the multimodal fusion model

As previously mentioned, the data splits not only align multimodal data, but also store the image features extracted by vgg19/CNNs, graph cell file paths and genomic features. To incorporate the new image features extracted by CONCH when making splits, we develop two additional functions: one to extract CONCH features independently,

```
x_conch = torch.unsqueeze(normalize(x_path), dim=0)
features, hazard = model(x_path=x_conch.to(device))
```

and another to append these CONCH features in a manner similar to appending vgg features.

```
x_conch.append(get_conch_features(model, device,
    os.path.join(opt.dataroot, opt.roi_dir, img_fname)))
```

We also adjust the dataloader for these splits to properly load CONCH features.

The multilinear fusion still uses the Kronecker product and the gating-based attention mechanism from algorithms 1 and 2. The default setting we implement is that the genomic modality gates over the graph, vgg and CONCH modalities, which means that the omic gate is turned off and other gates are turned on as follows,

```
if self.gate4:
    h4 = self.linear_h4(vec4)
    z4 = self.linear_z4(vec4, vec3) if self.use_bilinear
    o4 = self.linear_o4(nn.Sigmoid()(z4) * h4)
```

where vec4 represents for CONCH features and vec3 represents for genomic features. The fusion process in Figure 44. We still append one to each unimodal feature representation to preserve feature interactions. The Kronecker product step is modified by incorporating an additional tensor, o4 from CONCH, using batch matrix multiplication again.

```
o1234 = torch.bmm(o123.unsqueeze(2), o4.unsqueeze(1)).flatten(start_dim=1)
```

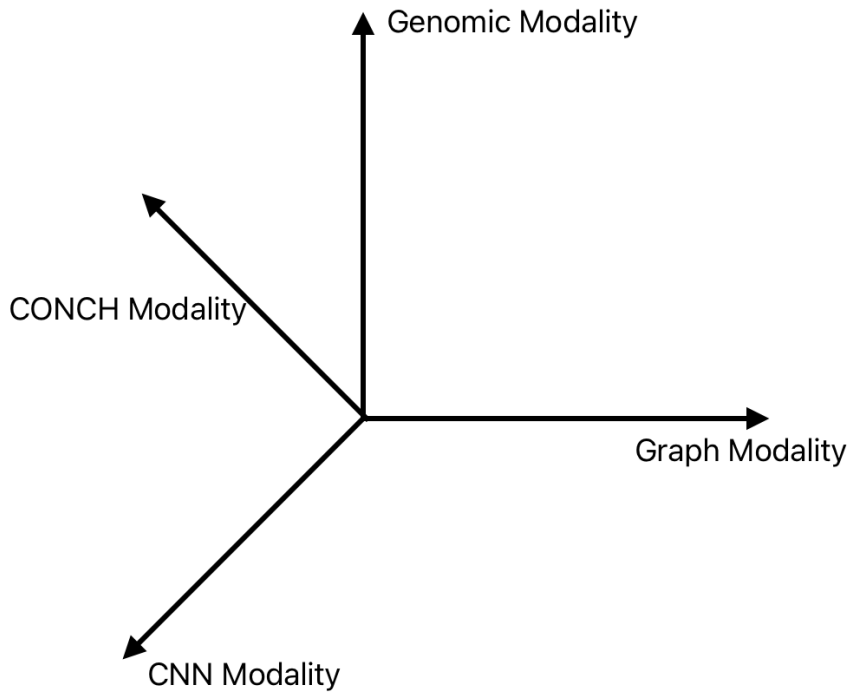


Figure 44: Demonstration of multilinear fusion, where CNN, CONCH and graph modalities are gated over by the genomic modality; CNN, CONCH and graph modalities are on the same plane.

8.6 Use KAN to Replace MLP for Genomic Data

The recent paper [21] finds an alternative way, Kolmogrov-Arnold Network (KAN), to replace the traditional MLP. We have attempted this new network with genomic features. Figure 45 demonstrates how to explicitly

parametrise $f(\mathbf{x}) = f(x_1, \cdots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$. We observe that each 1D function is parametrised as a B-spline curve with learnable coefficients as in Figure 46. The training process have been completely changed because (1) the activation functions become learnable rather than being fixed and (2) the activation functions are placed on edges rather than nodes in KANs. To implement KANs, we first

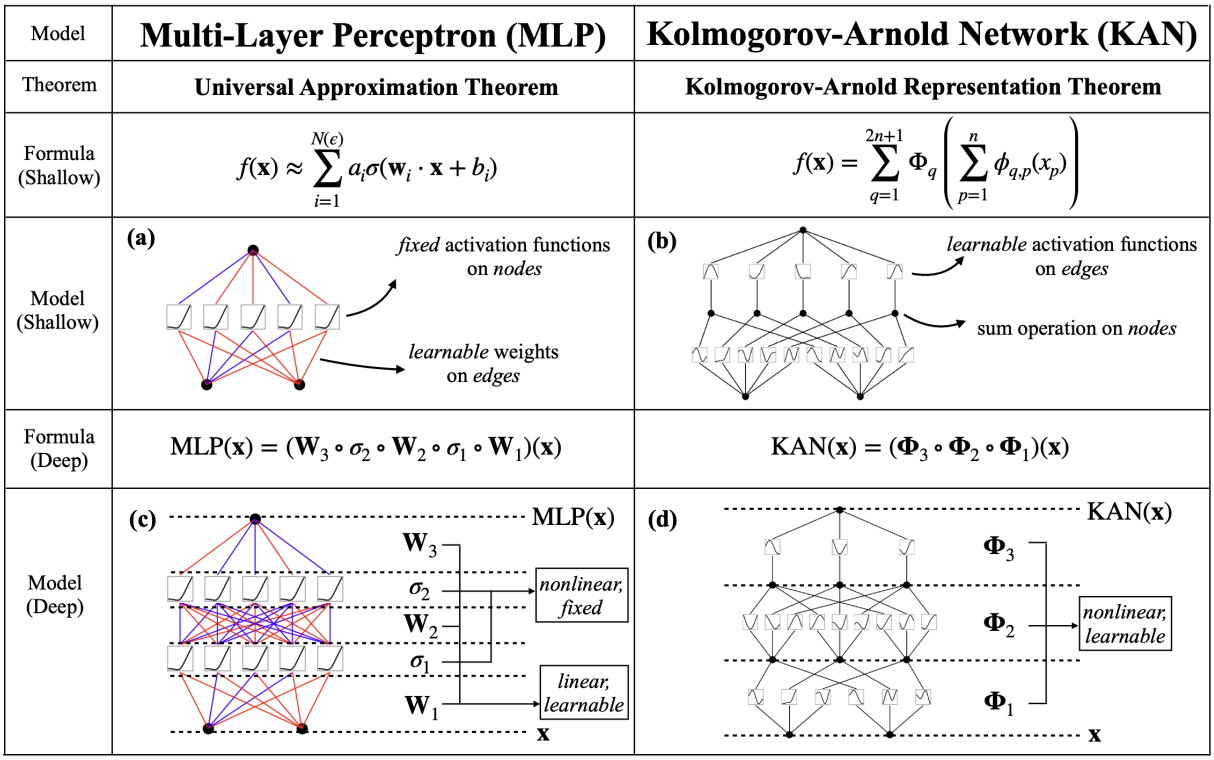


Figure 45: Demonstration of Kolmogrov-Arnold Network (KAN)

define a 'create_dataset' function to label the train and test data, and define the depth and the width by stacking more KAN layers (proof is in [21]).

```
net = KAN(width=[opt.input_size_omic,1,opt.label_dim], grid=200, k=3, seed=0)
```

The loss used during the training is RMSE rather than the customised cox loss, and the output has been adjusted to the hazard score for comparison purposes.

The c-index from genomic data becomes slightly higher after implementing KAN on the same data split from the paper, but we have not adopted KAN to other modalities due to time constraints. If we decrease the size of input genomic features, KANS might significantly outperform MLPs, as KANs offer better accuracy with fewer parameters.

	KAN	**MLP**
c-index value	0.813 ± 0.012	0.808 ± 0.014

Table 1: Comparison of c-index values from genomic data for KAN and MLP; we use the original data split from the paper with omic input size of 320.

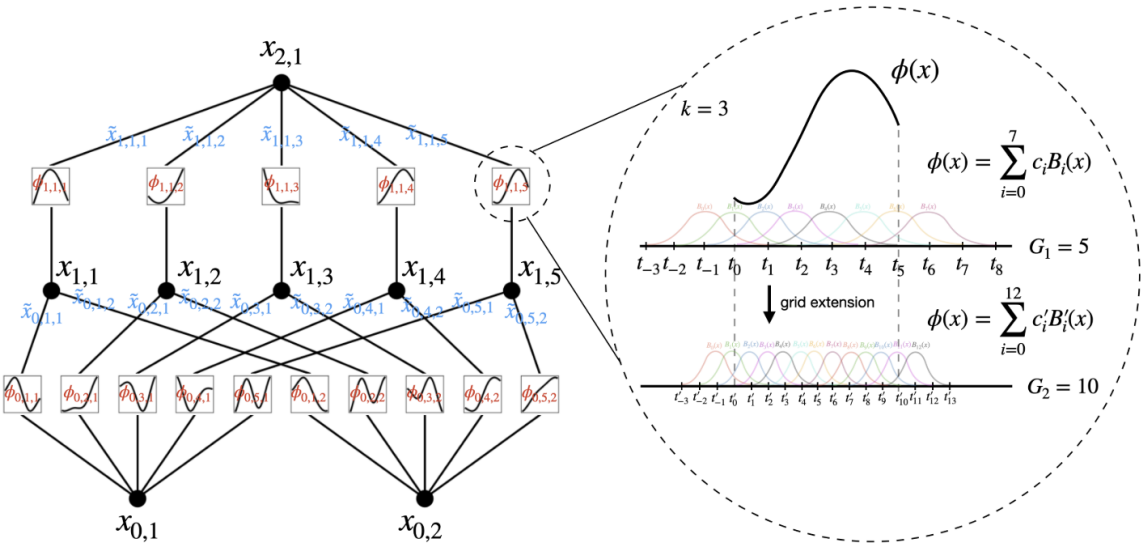


Figure 46: Activation function can be parameterised as a B-spline in KANs.

9 Reference

References

[1] Mohamed Abdelmoula Walid. *Data analysis for mass spectrometry imaging: methods and applications*. PhD thesis, Leiden University, 2017.

[2] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. Gated multimodal units for information fusion. *arXiv preprint arXiv:1702.01992*, 2017.

[3] Human Protein Atlas. Expression of tuba3c in colorectal cancer. https://www.proteinatlas.org/ENSG00000167057-TUBA3C/pathology. Accessed: 2024-06-28.

[4] Christopher M Bishop. Regularization and complexity control in feed-forward networks. 1995.

[5] Aleksandra Bożyk, Kamila Wojas-Krawczyk, Paweł Krawczyk, and Janusz Milanowski. Tumor microenvironment—a short review of cellular and interaction diversity. *Biology*, 11(6):929, 2022.

[6] Ethan Cerami, Jianjiong Gao, Ugur Dogrusoz, Benjamin E Gross, Selcuk Onur Sumer, Bülent Arman Aksoy, Anders Jacobsen, Caitlin J Byrne, Michael L Heuer, Erik Larsson, et al. The cbio cancer genomics portal: an open platform for exploring multidimensional cancer genomics data. *Cancer discovery*, 2(5):401–404, 2012.

[7] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[8] Travers Ching, Xun Zhu, and Lana X Garmire. Cox-nnet: an artificial neural network method for prognosis prediction of high-throughput omics data. *PLoS computational biology*, 14(4):e1006076, 2018.

[9] Pierre Courtiol, Charles Maussion, Matahi Moarii, Elodie Pronier, Samuel Pilcer, Meriem Sefta, Pierre Manceron, Sylvain Toldo, Mikhail Zaslavskiy, Nolwenn Le Stang, et al. Deep learning-based classification of mesothelioma improves prediction of patient outcome. *Nature medicine*, 25(10):1519–1525, 2019.

[10] Athena Davri, Effrosyni Birbas, Theofilos Kanavos, Georgios Ntritsos, Nikolaos Giannakeas, Alexandros T Tzallas, and Anna Batistatou. Deep learning for lung cancer diagnosis, prognosis and prediction using histological and cytological images: a systematic review. *Cancers*, 15(15):3981, 2023.

[11] O Gallego. Nonsurgical treatment of recurrent glioblastoma. *Current oncology*, 22(4):273–281, 2015.

[12] Simon Graham, Quoc Dang Vu, Shan E Ahmed Raza, Ayesha Azam, Yee Wah Tsang, Jin Tae Kwak, and Nasir Rajpoot. Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical image analysis*, 58:101563, 2019.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] Fabian Hörst, Moritz Rempe, Lukas Heine, Constantin Seibold, Julius Keyl, Giulia Baldini, Selma Ugurel, Jens Siveke, Barbara Grünwald, Jan Egger, et al. Cellvit: Vision transformers for precise cell segmentation and classification. *Medical Image Analysis*, 94:103143, 2024.

[15] Simonyan Karen. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv: 1409.1556*, 2014.

[16] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[17] A Kolesnikov, A Dosovitskiy, D Weissenborn, G Heigold, J Uszkoreit, L Beyer, M Minderer, M Dehghani, N Houlsby, S Gelly, et al. An image is worth 16× 16 words: Transformers for image recognition at scale. arxiv 2021. *arXiv preprint arXiv:2010.11929*.

[18] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743. PMLR, 2019.

[19] Kyubum Lee, John H Lockhart, Mengyu Xie, Ritu Chaudhary, Robbert JC Slebos, Elsa R Flores, Christine H Chung, and Aik Choon Tan. Deep learning of histopathology images at the single cell level. *Frontiers in artificial intelligence*, 4:754641, 2021.

[20] Jana Lipkova, Richard J Chen, Bowen Chen, Ming Y Lu, Matteo Barbieri, Daniel Shao, Anurag J Vaidya, Chengkuan Chen, Luoting Zhuang, Drew FK Williamson, et al. Artificial intelligence for multimodal data integration in oncology. *Cancer cell*, 40(10):1095–1110, 2022.

[21] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[22] Ming Y Lu, Bowen Chen, Drew FK Williamson, Richard J Chen, Ivy Liang, Tong Ding, Guillaume Jaume, Igor Odintsov, Andrew Zhang, Long Phi Le, et al. Towards a visual-language foundation model for computational pathology. *arXiv preprint arXiv:2307.12914*, 2023.

[23] Faisal Mahmood, Daniel Borders, Richard J Chen, Gregory N McKay, Kevan J Salimian, Alexander Baras, and Nicholas J Durr. Deep adversarial training for multi-organ nuclei segmentation in histopathology images. *IEEE transactions on medical imaging*, 39(11):3257–3267, 2019.

[24] Andriy Marusyk, Vanessa Almendro, and Kornelia Polyak. Intra-tumour heterogeneity: a looking glass for cancer? *Nature reviews cancer*, 12(5):323–334, 2012.

[25] Pooya Mobadersany, Safoora Yousefi, Mohamed Amgad, David A Gutman, Jill S Barnholtz-Sloan, José E Velázquez Vega, Daniel J Brat, and Lee AD Cooper. Predicting cancer outcomes from histology and genomics using convolutional networks. *Proceedings of the National Academy of Sciences*, 115(13):E2970–E2979, 2018.

[26] Adriana Olar and Kenneth D Aldape. Using the molecular classification of glioblastoma to inform personalized treatment. *The Journal of pathology*, 232(2):165–177, 2014.

[27] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[28] Shaeke Salman and Xiuwen Liu. Overfitting mechanism and avoidance in deep neural networks. *arXiv preprint arXiv:1901.06566*, 2019.

[29] Denis Schapiro, Hartland W Jackson, Swetha Raghuraman, Jana R Fischer, Vito RT Zanotelli, Daniel Schulz, Charlotte Giesen, Raúl Catena, Zsuzsanna Varga, and Bernd Bodenmiller. histocat: analysis of cell phenotypes and interactions in multiplex image cytometry data. *Nature methods*, 14(9):873–876, 2017.

[30] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[31] Jingwen Wang, Richard J Chen, Ming Y Lu, Alexander Baras, and Faisal Mahmood. Weakly supervised prostate tma classification via graph convolutional networks. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, pages 239–243. IEEE, 2020.

[32] Manfred Westphal, Cecile L Maire, and Katrin Lamszus. Egfr as a target for glioblastoma treatment: an unfulfilled promise. *CNS drugs*, 31:723–735, 2017.

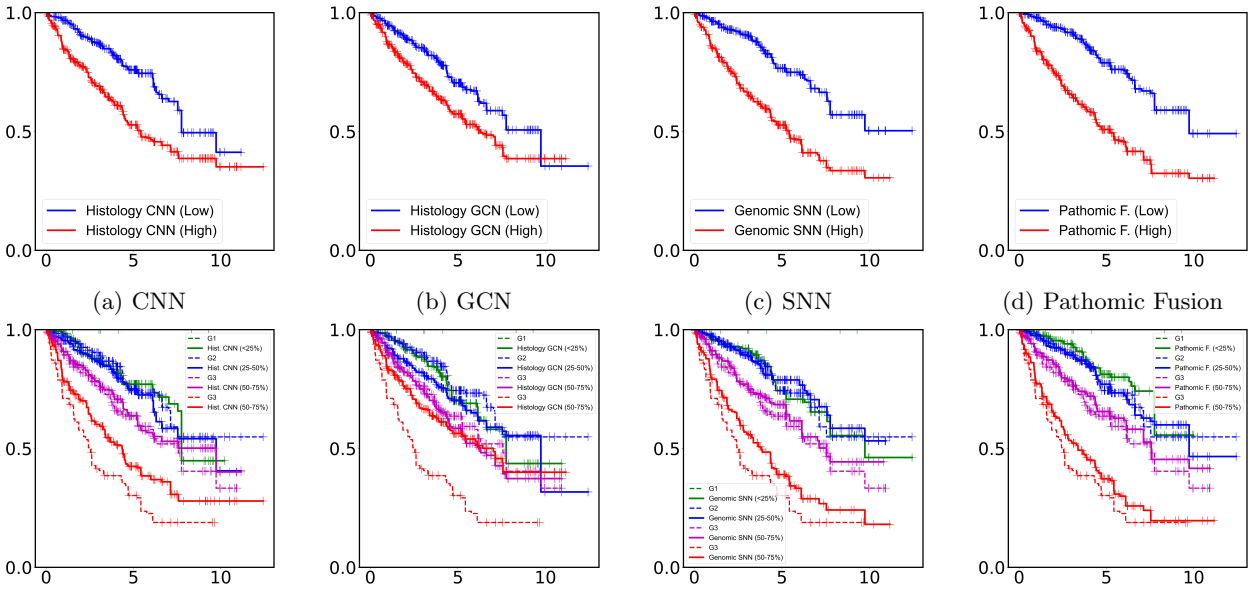
10 Appendix

10.1 Declaration of Auto-generation Tools

The autocompletion feature of GitHub Copilot was passively used throughout the code development process. It is pervasive to all the code without the use of any specific prompts.

The writing of this report was completed manually without the use of any auto-generation tools.

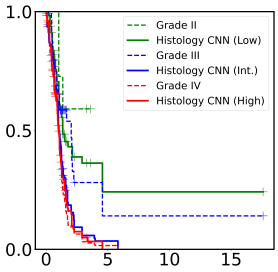
10.2 Additional Evaluation Plots



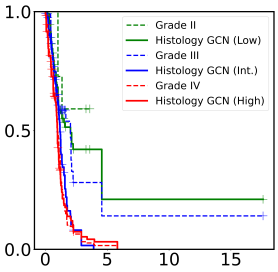
(a) CNN (b) GCN (c) SNN (d) Pathomic Fusion

(e) CNN (f) GCN (g) SNN (h) Pathomic Fusion

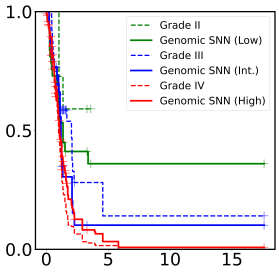
Figure 47: CCRCC Kaplan-Meier comparative analysis of Histology CNN, Histology GCN, Genomic SNN, and Pathomic Fusion with respect to low vs. high survival using the 50-100 percentile of hazard predictions, and Fuhrman Grades I, II, III, and IV using the 25-50-75-100 percentile of hazard predictions.



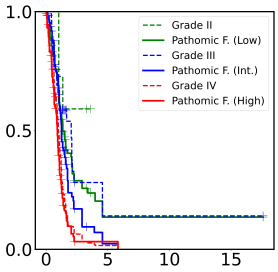
(a) IDHwt - CNN



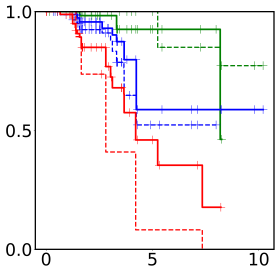
(b) IDHwt - GCN



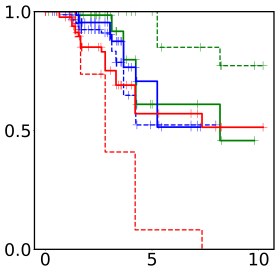
(c) IDHwt - SNN



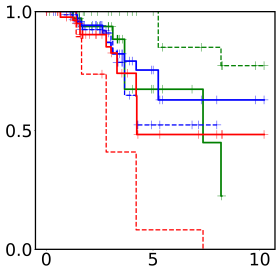
(d) IDHwt - Pathomic Fusion



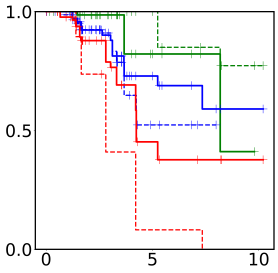
(e) IDHmut - CNN



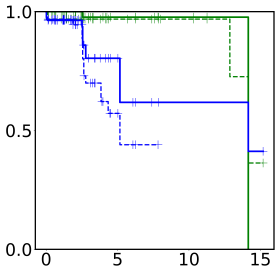
(f) IDHmut - GCN



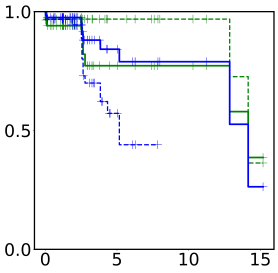
(g) IDHmut - SNN



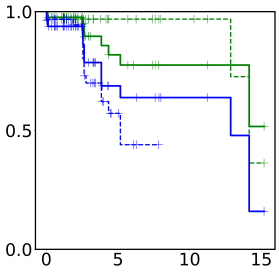
(h) IDHmut - Pathomic Fusion



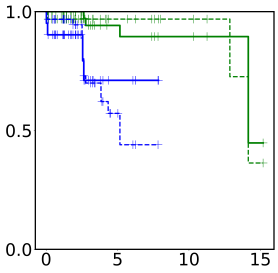
(i) ODG - CNN



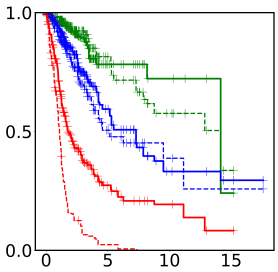
(j) ODG - GCN



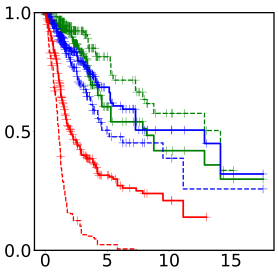
(k) ODG - SNN



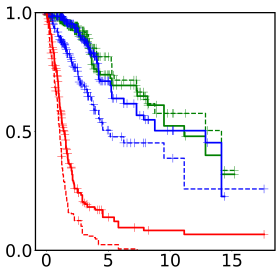
(l) ODG - Pathomic Fusion



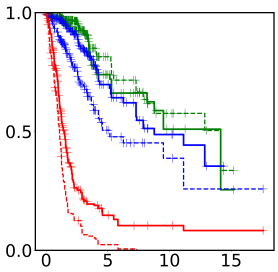
(m) Overall - CNN



(n) Overall - GCN

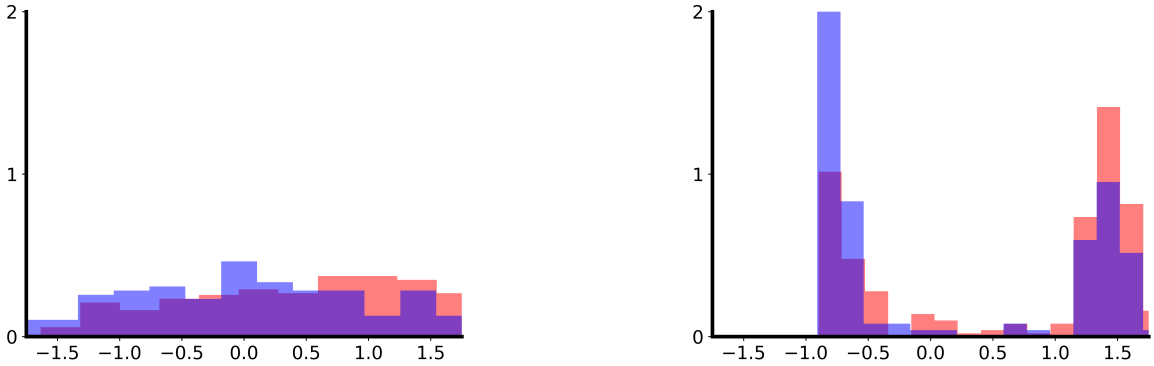


(o) Overall - SNN



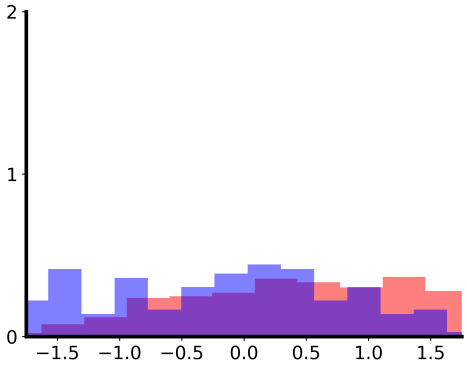
(p) Overall - Pathomic Fusion

Figure 48: GBMLGG Kaplan-Meier comparative analysis of Histology CNN, Histology GCN, Genomic SNN, and Pathomic Fusion with respect to IDHwt ATCs, IDHmut ATCs, ODGs, and all molecular subtypes in stratifying WHO Grades II, III, and IV using the 33-66-100 percentile of hazard predictions.

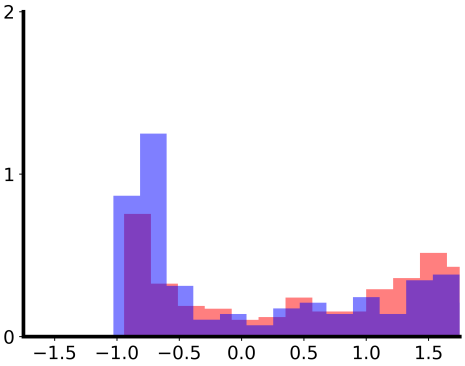


(a) CNN

(c) SNN

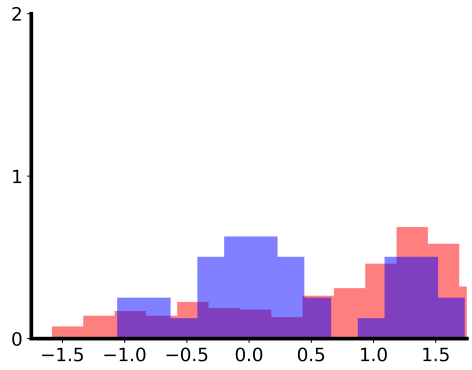


(b) GCN

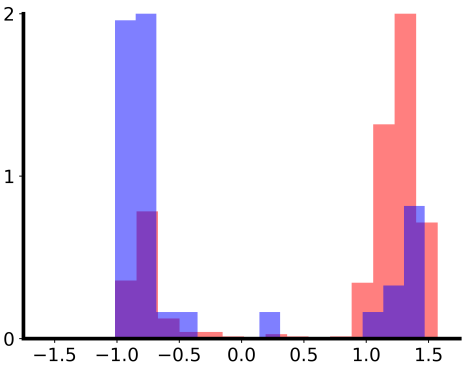


(d) Pathomic Fusion

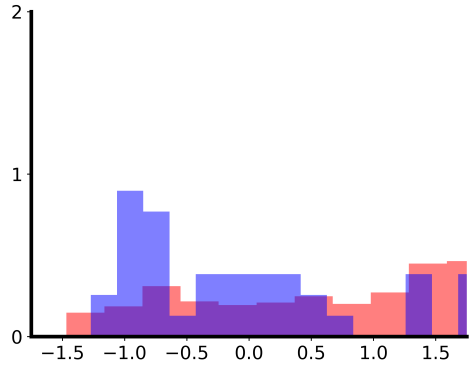
Figure 49: CCRCC Distribution of hazard predictions for Histology CNN, Histology GCN, Genomic SNN, and Pathomic Fusion.



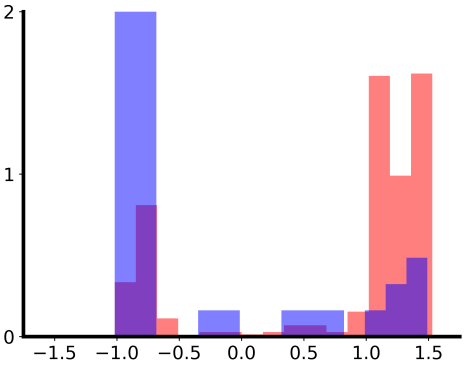
(a) CNN



(c) SNN



(b) GCN



(d) Pathomic Fusion

Figure 50: GBMLGG Distribution of hazard predictions for Histology CNN, Histology GCN, Genomic SNN, and Pathomic Fusion.