Visualization Learning to show Yap1 overexpression in TCGA-HNSC tumor slides using ResNet.

Results

Using gene expression data (RNAseq) on 757 TCGA Head and Neck Squamous Cell Carcinoma patient tumor samples from the National Cancer Institute, the ResNet deep learning model was able to detect Yap1 overexpression with a positive predictive value of 87.8% and a .9383 AUC ROC score. The model’s prediction AUC ROC is plotted below in figure 1. Yap1 overexpression positive values were determined as described in the materials and methods section.

A graph of a function

Description automatically generatedMaterials and Methods

Model Training and Evaluation

The training data was labeled 0 or 1 according to the threshold determined using descriptive statistics for the gene expression values in the dataset. The cutoff for YAP1 overexpression was determined after outliers were identified within the TCGA HNSC and normal control datasets by analyzing the distribution of YAP1 expression levels across all samples. Outliers were removed from the dataset using the Tukey method from the datasets. Following outlier removal, the cutoff for YAP1 overexpression was determined using the mean plus one standard deviation to establish a baseline cutoff value. Additionally, the biological context and significance of YAP1 expression cutoff was considered when determining this signal since potential benefit from YAP targeting therapies may occur on the lower end of the YAP-dependent cancer threshold. These values created the ground truth referenced for labeling tile images in ResNet. The metadata associated with the cases used can be found in the Supplemental Table.

The models were constructed and trained using the Python library PyTorch. We utilized a pre-trained ResNet101 as the base model, adapted by appending fully connected layers configured with 512, 256, and 1 output units respectively, incorporating ReLU activations for non-linearity and dropout layers between each fully connected layer to prevent overfitting. Specifically, we used a dropout rate of 0.5 to deactivate neuron connections randomly during training phases.

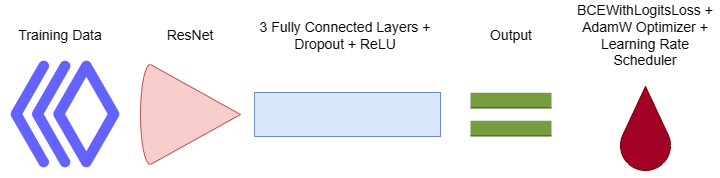
The final output layer uses a sigmoid activation function to yield the probability of binary classes. The loss was computed using Binary Cross-Entropy Loss (BCEWithLogitsLoss). For optimization, we used the AdamW optimizer with an initial learning rate of 0.001. To adjust the learning rate efficiently, ReduceLROnPlateau was used with a factor of 0.4 when the validation loss plateaus, enhancing the ability of convergence.

Training involved shuffling the data with random transformations including horizontal flip, vertical flip, color jitter, and rotation up to 20 degrees.

The model was set to train for up to 70 epochs, but with an early stopping mechanism if the validation loss did not improve for 35 consecutive epochs.

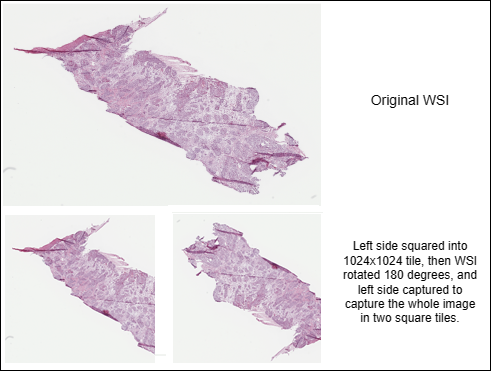
The dataset comprised 757 samples which were converted into 1514 1024x1024 tiles. These tiles were then stratified into a training set of 1060 tiles (70%), an evaluation set of 151 tiles (10%) and a testing cohort of 303 tiles (20%), based on predefined criteria. After training, the model achieved an Area Under the Receiver Operating Characteristic (AUROC) of 0.9383, showcasing high accuracy in distinguishing between the classes. The model demonstrated a Precision (Positive Predictive Value) of 0.8780 and Recall (Sensitivity) of 0.8727, with a Specificity of 0.8925 and an F1 Score of 0.8754. The Negative Predictive Value (NPV) stood at 0.9125.

The trained model was leveraged to predict outcomes for the testing cohort, providing insights into the prognostic value of the model in practical scenarios.



Data Preparation

An instance of the image preparation is shown below. Out of 791 samples available in the NCI’s database for TCGA-HNSC, 757 of these were tumor samples that also contained gene expression profiles. Using these 757 samples, the WSIs were processed into tiles using OpenSlide, and resized to 1024 x 1024 pixels.



Each tile was then labeled as either a “1” if Yap1 is overexpressed, or “0” if Yap1 is under the threshold using the ground truth created in the Materials and Methods section.

Supplementary Information

\*PROVIDE Supplemental Table with tiles, yap (1/0),

[Editing HNSC\_expressionvis/README.md at main · Paulisure/HNSC\_expressionvis (github.com)](https://github.com/Paulisure/HNSC_expressionvis/edit/main/README.md)

Acknowledgements

The results from this paper are in whole or in part based on data gathered from the National Cancer Institute’s GDC Data Portal: <https://portal.gdc.cancer.gov>.

R and Bioconductor: For initial data processing and statistical analysis, we utilized R and several packages from the Bioconductor project, which provided robust tools for analyzing genomic data.

OpenSlide: We used OpenSlide, an open-source C library that provides a simple interface to read whole-slide images of digital pathology slides. This tool was essential for processing and extracting tiles from whole-slide images, facilitating the generation of the input data for our deep learning models.

PyTorch and ResNet: Our models were constructed and trained using PyTorch, a leading deep learning framework that offers both flexibility and a powerful array of tools for building and training neural networks. Specifically, we employed a pre-trained ResNet101 model as the backbone for our neural networks, leveraging its powerful feature extraction capabilities which are well-documented for achieving high performance in image classification tasks.

AdamW and ReduceLROnPlateau from PyTorch's torch.optim library: These components of PyTorch were used for optimizing our neural network, with AdamW handling the weight updates and ReduceLROnPlateau managing the learning rate adjustments based on the validation performance.

Author Contributions

Paul Loupe proposed the visual deep learning method, collected, organized, prepared the data, customized the model and prepared the manuscript. Dr. Sophia Shalhout initiated the concept and participated in writing, critical review, commentary, and revision of the manuscript.

Code Availability

A link to the Github page for the code used to generate these results can be found in the Supplementary Information section.

**Not included in paper -**

**Best model results**:

Evaluating the best model:

Confusion Matrix:

[[107  20]

 [ 20 157]]

AUROC: 0.8648

Precision (PPV): 0.8870

Recall (Sensitivity): 0.8870

Specificity: 0.8425

F1 Score: 0.8870

Positive Predictive Value (PPV): 0.8870

Negative Predictive Value (NPV): 0.8425

Best results (so far) achieved with the following settings:

Learning Rate: 0.0001

Batch Size: 16

Epochs: 60

Dropout 0.7

AdamW 0.001

Patience 30 epochs

Optimizer Step 10 | Gamma 0.8