Biology-guided deep learning predicts prognosis and cancer immunotherapy response.

Adithya Suresh Babu

Department of Computer Science

University of Central Florida

Type of Project – Implementation and Improvement of Chosen Biomedical paper

Background:

Substantial progress has been made in utilizing deep learning for cancer detection through medical imaging, yet predicting treatment responses and outcomes remains a challenge. This study introduces a novel biology-guided deep learning approach, simultaneously predicting tumor microenvironment status, treatment outcomes, and responses to immune checkpoint inhibitors.

The model is validated in a multi-center international study focusing on gastric cancer prognosis and benefits of adjuvant chemotherapy. Additionally, it identifies tumors with mismatch repair deficiencies that don't respond to immunotherapy, offering insights for patient selection in combination treatments.

The study emphasizes the importance of interpretability, advocating for integrating disease biology insights into deep learning model development. The tumor microenvironment's significance is highlighted, emphasizing the role of imaging for noninvasive, repeated tumor analysis.

The novel framework involves training separate models for TME status and treatment outcomes, aiming to enhance precision and applicability, especially in predicting gastric cancer prognosis and guiding chemotherapy and immunotherapy decisions. International validation ensures model generalizability across diverse populations, presenting an innovative approach to personalized cancer treatment through deep learning.

Implementation:

Dataset:

Data for Image Classification:

The GasHisSDB dataset comprises 245,196 images categorized into 97,076 abnormal and 148,120 normal images. It undergoes a two-stage creation process, starting with 600 gastric cancer pathology images sized 2048×2048 from four pathologists. Sub-sized images are then generated by five biomedical researchers using a weakly supervised learning approach. Calibration involves two experienced pathologists. Specific rules govern dataset preparation, including cropping normal sections at various sizes, selecting cancerous regions of interest for abnormal sections, and applying random rotations to minimize correlation. The chosen subset for the model consists of 160 × 160 pixels images. The overall image order in the database is randomized to prevent biases.

Data for TME prediction:

The TME dataset comprises information from 2149 patients, with DFS (disease-free survival) representing the duration between initial cancer treatment and disease

reappearance or secondary cancer emergence. DFS is a critical clinical endpoint for assessing cancer treatment efficacy and overall patient prognosis. "OS" denotes overall survival, measuring the time from disease diagnosis or treatment initiation to patient demise, playing a crucial role in clinical trials and offering insights into treatment efficacy and patient prognosis.

TNM stages categorize the gastric cancer TME into four classes using well-established biomarkers: ISGC (ImmunoScore of Gastric Cancer) and POSTN (periostin protein expression). ISGC incorporates immune cell types, calculated with specific coefficients for CD3, CD8, CD45RO, and CD66b. POSTN, a stromal protein, regulates cancer cell behavior. Both biomarkers indicate the TME's immune and stromal axes. TME classes are defined by dichotomizing ISGC and POSTN into low and high categories based on median values. This categorization results in four TME classes (1: high ISGC/low POSTN, 2: high ISGC/high POSTN, 3: low ISGC/low POSTN, 4: low ISGC/high POSTN), providing nuanced insights into TME heterogeneity in gastric cancer patients.

Data Preprocessing for Gastric CT Images:

I employed the split folders library to partition a dataset into training, testing, and validation subsets with a 70%, 20%, and 10% distribution ratio, respectively. This process, utilizing a seed value of 38, ensures consistent and reproducible random splitting, and the resulting datasets are stored in the output directory.

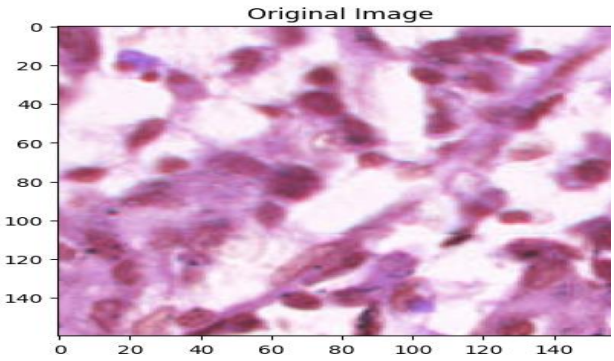
For the convolutional neural network (CNN), I configured data generators using TensorFlow's ImageDataGenerator class. These generators apply various data augmentation techniques, such as rescaling, shearing, zooming, and horizontal flipping, specifically tailored for efficient batch processing during model training (train_set), testing (test_set), and validation (val_set).

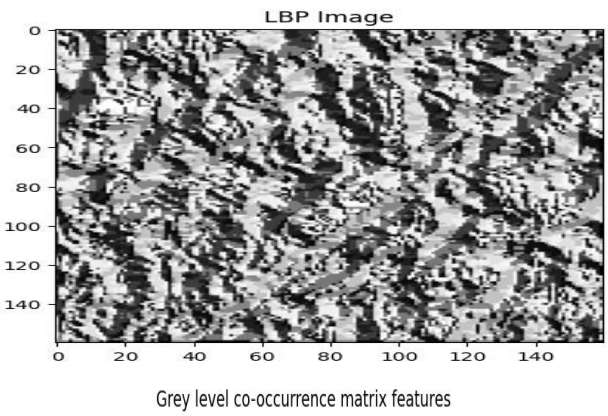
To prepare the database for machine learning classification, two feature extraction methods were applied: Local Binary Patterns (LBP) and Gray-level Co-occurrence Matrix (GLCM). These methods capture texture features reflecting surface homogeneity and contribute to a quantitative analysis of structural organization through statistical calculations on pixel regions.

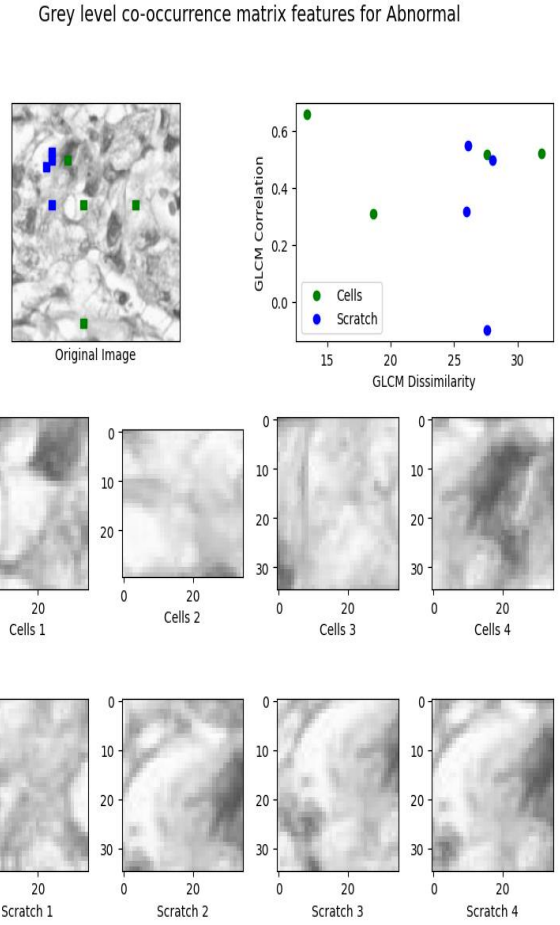
LBP, recognized for its texture description efficiency, ensures grey level and rotational invariance by assigning binary values based on pixel comparisons within an eight-neighbor configuration.

GLCM, characterized by joint probability density of pixels, portrays brightness distribution and spatial relationships. It serves as the foundation for extracting features like Contrast, Correlation, Energy, and Homogeneity.

Contrast: Reflects image sharpness and depth of texture grooves.
Correlation: Measures similarity in grayscale matrix elements.
Energy: Represents the sum of squares, indicating uniformity and thickness of texture.
Homogeneity: Measures the compactness of the distribution of diagonal elements.







Machine Learning Methods:

Image Classification:

The Inception architecture, introduced by Google, revolutionizes image classification. Its key innovation lies in inception modules, utilizing parallel convolutional operations with varying filter sizes, enabling hierarchical feature extraction at multiple scales. To enhance computational efficiency, 1x1 convolutions serve as bottleneck layers. Auxiliary classifiers at intermediate stages combat the vanishing gradient problem during training. Global average pooling replaces fully connected layers, aiding in reducing overfitting. Inception's versatility in capturing diverse features at different scales makes it influential in deep learning. Its innovative design and efficiency

principles have inspired subsequent neural network architectures.

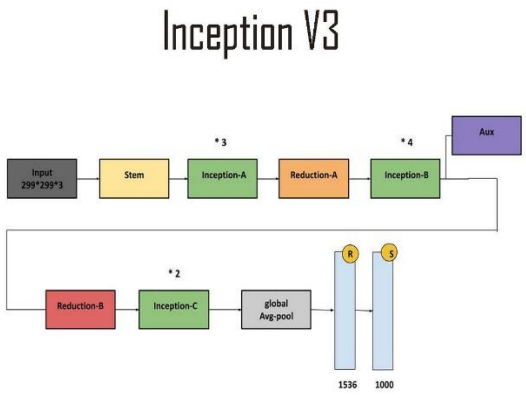
My Machine Learning Model establishes a neural network model utilizing the Inception architecture, a popular design for image classification. Let's break down the key components:

Inception Module Definition:
The inception module function defines a building block of the Inception architecture. It incorporates multiple convolutional operations with distinct filter sizes to capture features at various scales.
The module takes an input tensor x and applies convolutional layers with 1x1, 3x3, and 5x5 filter sizes independently. Additionally, it includes a max-pooling operation.
The outputs of these operations are concatenated along the depth axis to create a richer feature representation.

Model Architecture:
The model starts with standard convolution and pooling layers.
Inception modules (inception_3a, inception_3b, etc.) are used to extract hierarchical features from the input image.
Auxiliary classifier layers (auxiliary_output_1 and auxiliary_output_2) are strategically inserted to assist in training and enhance gradient flow.
The architecture concludes with more Inception modules, global average pooling, dropout, and fully connected layers.
The model has three output branches, each with its own loss function and associated weight.

Learning Rate Schedule:
A learning rate schedule is implemented to dynamically adjust the learning rate during training. The decay function defines a stepwise decay strategy, gradually reducing the learning rate after specific epochs.

Model Compilation:
The model is compiled using stochastic gradient descent (SGD) as the optimizer with momentum.
Three output branches exist, and each is assigned a distinct loss function. The contributions of these branches during training are weighed accordingly.
Categorical cross-entropy is chosen as the loss function, and model performance is evaluated using accuracy. Finally, I trained the neural network model using TensorFlow's Keras API. It utilizes the model.fit function to train on a dataset (train_set) with validation on another dataset (val_set). The training process spans 25 epochs, with batches of 256 samples processed in each iteration. Additionally, a learning rate scheduler callback (lr_sc) is employed to dynamically adjust the learning rate during training, contributing to more effective optimization.



TME Prediction:

This is a classification with multi-class classification problem since there are four TME classes to predict. First, I started checking with supervised learning algorithms like AdaBoost, K-nearest Neighbors and Gradient Boosting Classifier.

I took these algorithms because they can handle multi class classification problems.

First, I split the data into training and testing using train_test_split from sklearn.model_selection. The AdaBoost Classifier, employing 1000 weak learners, is trained on a labeled dataset (X_train, y_train). Subsequently, it predicts labels for a distinct test dataset (X_test). The accuracy of these predictions is then assessed using the accuracy score function, with the obtained accuracy being printed as the result.

The K-Nearest Neighbors (KNN) Classifier, configured with 100 neighbors, is trained on a dataset (X_train, y_train). It then predicts labels for a separate test dataset (X_test). The accuracy of these predictions is evaluated using scikit-learn's accuracy score and printed as the result.

The Gradient Boosting Classifier utilizes the Gradient Boosting algorithm with 59 weak learners, a learning rate of 0.8, and a maximum depth of 1. It is trained on a specified dataset (X_train, y_train) and predicts labels for an independent test dataset (X_test). The model's accuracy is assessed using scikit-learn's accuracy_score, and the result is printed, offering insights into its performance on the test dataset.

Finally, I want to use Fully Connected Neural network for TME prediction, so I created a neural network model using TensorFlow's Keras API. It consists of two dense layers, the first with 256 units and a sigmoid activation function, and the second with 1 unit and a sigmoid activation function. The model is compiled using the Adam optimizer, binary cross entropy loss, and accuracy as the evaluation metric. Training occurs on the provided dataset for 20 epochs. The model's architecture and parameters are summarized, and its performance is evaluated on a separate test dataset, showcasing loss and accuracy results. Overall, the code defines, compiles, trains, summarizes, and evaluates a binary classification neural network.

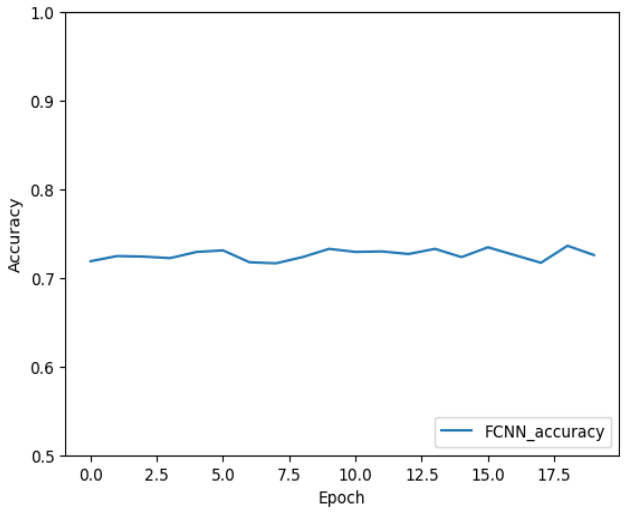
Results:

The initial image classification model gave me accuracy of 70.29% for running the model for five epochs. So, I ran the same model for 25 epochs which gave a boost in accuracy of 84% whereas the paper accuracy was 81.0%.
For TME prediction, I got accuracies for the supervised and fully connected network are below in the table.

Model	My Model	Paper Model
AdaBoost	73.0%	-
K-Nearest Neighbors	73.2%	-
Gradient Boosting Classifier	73.9%	-
Fully Connected Neural Network	74.8%	83.2%

Model	Precision	Recall	Paper Precision	Paper Recall
AdaBoost	62.2%	60.8%	-	-
K-Nearest Neighbors	59.2%	58.3%	-	-
Gradient Boosting Classifier	60.1%	59.8%	-	-
Fully Connected Neural Network	63.9%	62.95%	79.8%	70.36%

I got low accuracy for TME classification lack of data resources because I had only 2149 patient records which were made available for public access.



In paper, there used Resnet18 for Image classification and TME prediction, but I used InceptionV3 and fully connected Network for Image Classification and TME prediction. I used InceptionV3 because original the data size was small, they have opted for ResNet18 because it is suitable for limited resources, but the dataset has been changed, I opted for InceptionV3 which is more suitable for larger dataset because of its deeper network with layers than ResNet18. Comparatively the accuracy of Resnet18 after running for 20000 epochs is 81.0% whereas my model's accuracy was 83.2% for 25 epochs.

Conclusion:

In Conclusion, the research "Biology-guided deep learning for predicting cancer prognosis and response to immunotherapy" presents an innovative deep learning model designed to forecast treatment response and outcomes in gastric cancer through the analysis of radiology images. The model, assessed using precision, recall, and accuracy measurements, exhibits remarkable performance in predicting both tumor microenvironment (TME) classes and prognosis.

Insights:

The study pioneers an innovative approach by combining biological insights with advanced machine learning for cancer prognosis and treatment response prediction. Utilizing non-invasive radiology images, it comprehensively evaluates the tumor microenvironment (TME), overcoming limitations of traditional histopathology. The model's precision and recall, validated across diverse populations, emphasizes its accuracy and broad clinical relevance. With prognostic and predictive value, it refines staging, enhances risk stratification, and guides personalized treatment decisions. The model's potential to discern patients benefiting from adjuvant chemotherapy and predict immunotherapy responses underscores its role in personalized therapy.

References:

[1] Dataset: Link

[2] InceptionV3 Image: Image Link

[3] **GasHisSDB: A New Gastric Histopathology Image Dataset for Computer Aided Diagnosis of Gastric Cancer**
Weiming Hua, Chen Lia,,XiaoyanLib, Md Mamunur Rahamana, Jiquan Mac, Yong Zhangb,Haoyuan Chena, Wanli Liua, Changhao Suna, Yudong Yaod, Hongzan Sune and Marcin Grzegorzekf.

[4] **Rethinking the Inception Architecture for Computer Vision**
Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna