Final Project Report for CS 184A/284A, Fall 2024

Project Title: Malignant and Benign Breast Cancer Detection Using KNN and CNN
Project Number: 23

Student Name(s)
Ayaan Dhir, 73844062, ayaand@uci.edu
Rohit De, 30271270, rde1@uci.edu

1. Introduction and Problem Statement

Breast cancer is estimated to reach 310,720 cases in 2024, yet only 66% of diagnoses are found in early stages (American Cancer Society). Is this simply due to a gap in knowledge or a lack of data leverage? In today's world, where data is everywhere, how can we better understand breast cancer with the help of machine learning? With the usage of simple machine learning models, it became evident that breast cancer diagnoses can become more efficient and ultimately more accurate.

There exist large datasets filled with images of breast cancer that have been classified as either having breast cancer or not having it. Our goal is to decipher whether a kNN or a CNN model is more effective at diagnosing the type of breast cancer. The input into the model will be a set of features derived from a breast cancer analysis, and the output will be a prediction of whether the breast cancer is malignant or benign. We plan to classify utilizing a kNN model as displayed in this article and a CNN model as seen in our previous assignments. With a classification model, doctors could easily cross-verify their diagnosis in the attempt to have a data-driven approach. Our goal is to find the best one to utilize in breast cancer diagnoses.

2. Related Work

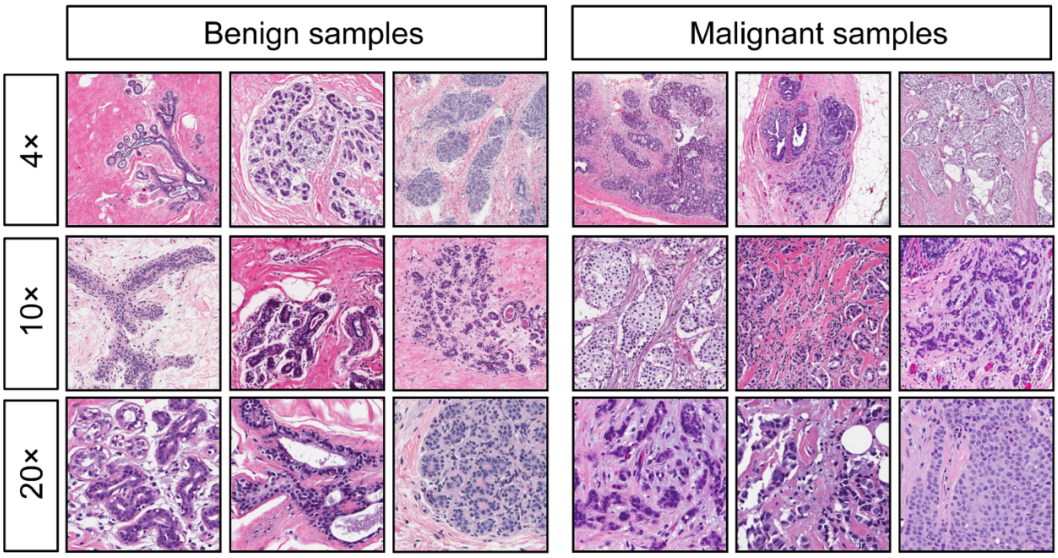
Machine learning has been used extensively in the medical field in recent years, especially with better understanding of data that is readily available. A research report from April 11, 2024 shows the utilization of machine learning models such as decision tree, random forest, logistic regression, naive bayes, and XGBoost being used to help predict breast cancer from a dataset taken from the Dhaka Medical College Hospital with a split of 254 noncancerous cases and 246 cancerous. We wanted to try out kNN and CNN models to see how they would operate given this research report chose other machine learning techniques.

The Breast Cancer Research Foundation recently released research that explored the impact that AI can have on breast cancer detection. Two of their researchers developed a deep learning model called MIRAI that was able to analyze multiple mammogram images over time, and provide risk factor information regarding a diagnosis. Their findings were that AI in large, can help to make radiologists more effective and accurate when screening for breast cancer. Our approach in the scope of the project was to help mimic some of these findings with a simple machine learning model such as a kNN or CNN model.

3. Data Sets

In our project we utilized the Breast Cancer Wisconsin (Diagnostic) dataset. This dataset was curated by Dr. William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian. It contains 569 instances with 30

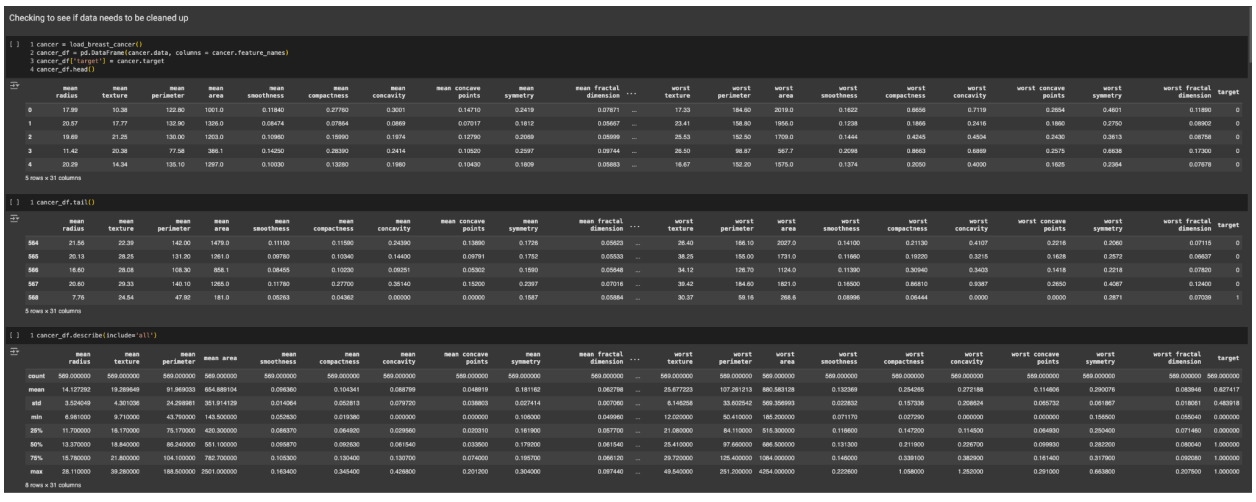
different features to analyze. The distribution of data is 212 malignant, 357 benign. These features include metrics denoting both statistical and geometrical descriptors of the tumor cells. Attributes like the mean radius, texture, and various second-order statistics like compactness and concavity errors provide a diversified data foundation to enhance diagnostic accuracy.



Shown above is a comparison between the benign and malignant samples in our dataset in a visual sense. Our data was a compilation of numerical values for 30 features rather than an image.

We checked for missing values and split the data into training and test sets using train_test_split() to facilitate robust validity.

By checking the extent of missing data through counts and percentages, we can make informed decisions for handling these missing entries, such as dropping features with incomplete cases or assessing data collection methods.



```python
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state =
RANDOM_STATE)
# spliting data into training and testing data subsets
```

4. Description of Technical Approach

KNN Model:

The kNN model involves selecting an appropriate k value, which can dramatically influence the model's accuracy. The KNeighborsClassifier() provides the architecture necessary to fit a kNN model, with experiments conducted over various k values to ascertain optimal neighbors for classification tasks. The kNN model learns by example, where each prediction relies on the majority class of k nearest points. Its efficacy lies in its ease of understanding and implementation, making it a great choice for baseline comparisons against more complex models such as CNNs. Predictive accuracy is measured, with optimal results indicating a best-case accuracy of approximately 90.21% when k=1. Visualization through a confusion matrix elucidates true vs. false classifications, providing clarity into which tumor types are being effectively classified.

```python
def get_best_knn_neighbors():
    global X_train, X_test, y_train, y_test  # Access global training and
testing datasets
    best_scores = 0
    best_neighbor = 0
    # Loop through values of k (number of neighbors) from 1 to 100
    for i, k in enumerate(range(1, 101)):
        # Create a KNeighborsClassifier with k neighbors
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train) # Fit the classifier using the training data
        score = knn.score(X_test, y_test) # Evaluate the classifier on the test
data
# If current score is better than the best score, update the best values
        if score > best_scores:
            best_scores = score
            best_neighbor = k

    # Return the number of neighbors that resulted in the best score
    return best_neighbor

best_k = get_best_knn_neighbors()
print('The best number of neighbors is: {}'.format(best_k))
n_neighbors = best_k
preds_default, knn_default = knn_model_fitting(X_train, X_test, y_train,
y_test, n_neighbors)
```

CNN Model:

A sequential 1D CNN model is constructed, featuring various layers that encompass convolutional, batch normalization, and dropout layers, all finely tuned to capture intricate patterns within the 1D inputs. Convolutional layers in a 1D CNN slide filters over the input data, learning and extracting meaningful features from the raw numerical values. This process allows the model to identify patterns and structures within the data that are crucial for distinguishing between different cancer tissue types.

The architecture begins with a Conv1D layer with parameters like a kernel size of 2 and an activation function of ReLU to ensure efficient feature extraction. Batch normalization layers follow, stabilizing learning by normalizing inputs to the subsequent layer. Dropout mechanisms are employed at varying rates to counteract overfitting, a common issue in deep learning models. The model structure includes a Flatten layer that ensures dimensional compatibility, followed by Dense layers. The final layer utilizes the sigmoid activation function, converting outputs to a range suitable for binary classification.

The model undergoes an optimization process using Adam, known for its adaptability and precise gradient calculation, with binary cross-entropy serving as the loss function to model binary classification effectively. The CNN model demonstrates higher accuracy, peaking at 98.95% during training epochs and maintaining robust performance in validation datasets.

Python

```
X_train = X_train.reshape(398,30,1)
X_test = X_test.reshape(171,30,1)
'''
The reshaping to (398, 30, 1) ensures that each sample is treated as a sequence
consisting of 30 features with a single channel, which aligns with the input
requirement of a 1D CNN.
'''
```

Python

```
def create_model():
    model = Sequential()

    # Add the first 1D convolutional layer
    # - filters=16: Number of filters (16 feature detectors)
    # - kernel_size=2: Size of the convolution kernel (2 time steps wide)
    # - activation='relu': ReLU activation function to introduce non-linearity
    # - input_shape=(30, 1): Input shape (30 time steps, 1 feature per step)
    model.add(Conv1D(filters=16, kernel_size=2, activation='relu',
input_shape=(30, 1)))

    # Normalizes the output of the previous layer to stabilize learning
```

```
    model.add(BatchNormalization())

    # Add dropout layer
    # Randomly disables 20% of neurons during training to prevent overfitting
    model.add(Dropout(0.2))

    # Add the second 1D convolutional layer
    model.add(Conv1D(32, 2, activation='relu'))

    # Add batch normalization layer
    model.add(BatchNormalization())

    # Add another dropout layer
    model.add(Dropout(0.2))

    # Converts the 3D tensor (time steps, filters) to a 1D vector to prepare
for the dense layers
    model.add(Flatten())

    # Add a dense (fully connected) layer
    model.add(Dense(32, activation='relu'))

    # Add a dropout layer
    model.add(Dropout(0.2))

    # Outputs a probability for binary classification
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(learning_rate=0.0001),
loss='binary_crossentropy', metrics=['accuracy'])

    return model
```

5. Software

Python Libraries Used:

- Pandas, NumPy, Matplotlib: These libraries facilitated data manipulation, numeric computations, and visualizations vital for model development.
- scikit-learn: Employed for kNN model implementation, tools for model evaluation.
- TensorFlow and Keras: Used to develop and train the CNN model, ensuring efficient handling of extensive datasets and deep learning tasks. Utilizing open-source frameworks, the project built foundational models addressing breast cancer diagnosis challenges.

Data Loading and Train-Test Split

- A DataFrame (cancer_df) is created using the dataset's features and target labels.
- The DataFrame is examined through methods like .head(), .describe(), and .info() to check for missing values, dimensions, and data types.
- The dataset is split into training and testing sets using train_test_split from sklearn.model_selection, with a random seed for reproducibility.

K-Nearest Neighbors (KNN) Model

- An initial KNN classifier is fitted and tested with n_neighbors set to 1, achieving an accuracy of approximately 90.21%.
- We determine the optimal number of neighbors (k) through iterations from 1 to 101.
- The best k value is found to be 3, which improves the model's accuracy to around 93.01%.

Convolutional Neural Network (CNN) Model

- The TensorFlow library is imported, and a sequential CNN model is defined with layers including Conv1D, MaxPooling, BatchNormalization, Dropout, and Dense layers.
- The CNN is compiled with Adam optimizer and binary cross entropy loss function.
- The model is trained over 35 epochs, with the accuracy peaking at approximately 95.91%.

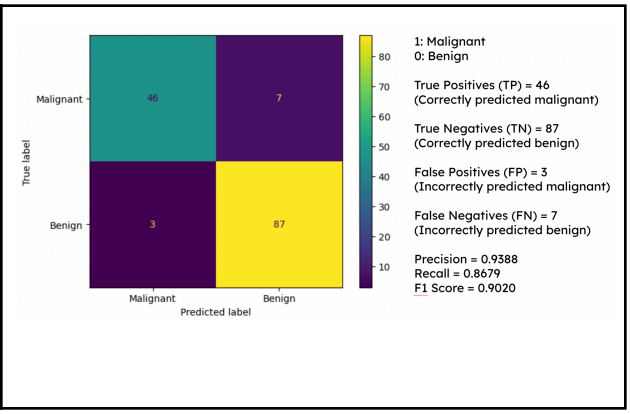
Visuals

- Plots are utilized to visualize the confusion matrices of both the KNN and CNN models for better insights into their capabilities.

6. Experiments and Evaluation

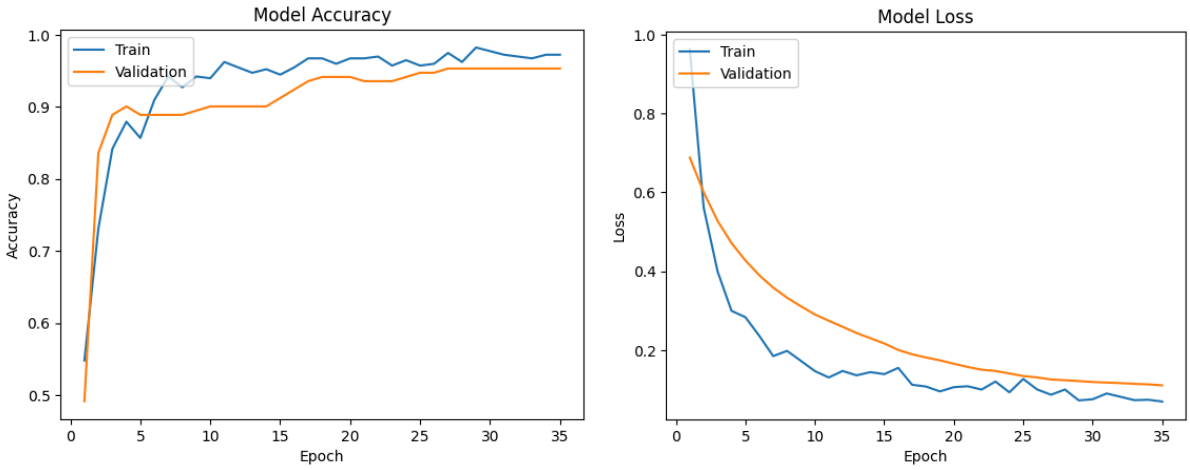
Our first step was training our models on the testing-training split that we set up. We split into training and testing sets using train_test_split from sklearn.model_selection, with a random seed for reproducibility.

Our first model, the kNN began with an initial value of k=1 and iterated till k=101. Shown below is a graph displaying the testing and training accuracy. Our ideal value for k was found to be 3 after conducting our testing. Also shown is our confusion matrix illustrating the overall accuracy of the kNN model.

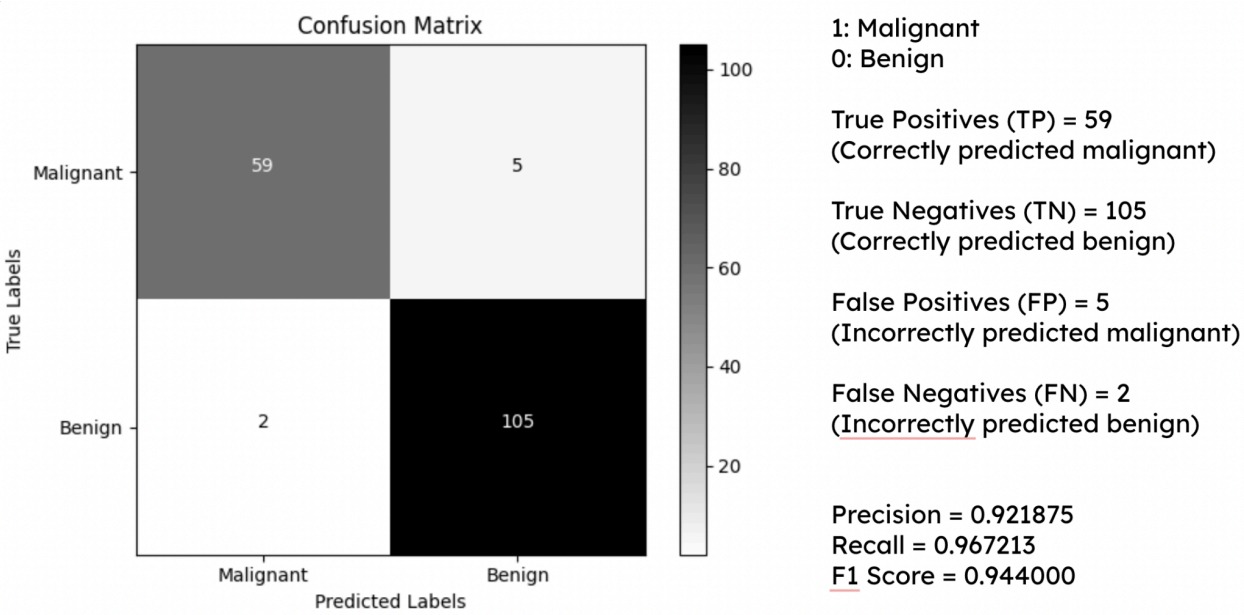


Our next model was the CNN model where we defined layers including Conv1D, MaxPooling, BatchNormalization, Dropout, and Dense layers. The CNN is compiled with Adam optimizer and binary

cross entropy loss function. We then trained the model over 35 epochs. Shown below is the model accuracy over the epochs and the model loss over epochs.



Our confusion matrix shown below makes it clear that the CNN model was more accurate than the kNN model with value k=3. The overarching difference was only 3% but we estimate that with larger datasets and a more fine tuned model the gap would grow.



The end accuracy of the CNN model was 96% compared to the 93% accuracy of the kNN model.

7. Discussion and Conclusion

Insights gained from deploying machine learning models for breast cancer detection illuminated significant advancements in diagnostic accuracy and provided valuable lessons about the strengths and

limitations of different algorithms. In comparing K-Nearest Neighbors (KNN) and Convolutional Neural Networks (CNN), it became clear that CNN outperformed KNN, likely due to its ability to effectively process complex high-dimensional data such as medical datasets. This result aligned with expectations, as CNNs are renowned for their feature extraction capabilities and adaptability to unstructured data. The project highlighted the transformative potential of integrating machine learning into medical diagnostics, paving the way for more precise, data-informed diagnostic strategies. Yet, it also revealed limitations in current approaches. For instance, the reliance on labeled data introduces challenges in scaling models to broader applications, as creating high-quality, annotated datasets is both time-intensive and expensive. Additionally, KNN, while conceptually straightforward and effective for small, well-separated datasets, struggles with scalability and performance on larger datasets, where its computational cost and sensitivity to noise become significant drawbacks. CNNs, despite their superior performance, are resource-intensive, requiring substantial computational power and expertise to train and fine-tune effectively.

If we were in charge of a research lab, our focus would include several key directions to address these challenges. First, we would invest in semi-supervised and unsupervised learning approaches to reduce the dependence on labeled data, exploring techniques such as self-supervised learning or generative models for feature extraction. Next, improving access to computational resources and exploring lightweight models that maintain performance while reducing computational demands could democratize these technologies. By addressing these limitations and exploring innovative solutions, machine learning has the potential to revolutionize breast cancer diagnostics, enabling earlier detection, personalized treatment strategies, and ultimately, better patient outcomes.

8. Individual Contributions

Ayaan: My involvement in the project was focused on the data research, analysis, and the development and training of the kNN model. The inception of our project was heavily influenced by research that we both did online on how imagery and data collected by hospitals are interpreted. Rohit and I found that radiologists regularly analyze and diagnose images that they receive along with data that they have to continuously process. Given how time-consuming that is, we wanted to develop a model that can help support radiologists in the process. We chose breast cancer given the large datasets available along with the 30 features that we could derive information from. I focused on the research of the overall process and the dataset analysis that we coded into our notebook. At the end I focused on developing the presentation and assisting Rohit in transferring our code to GitHub.

Rohit: My contribution to the project was concentrated on CNN model construction, training, and analysis. Our project's concept was inspired by research we both conducted online on the interpretation of hospital data and photos. Radiologists routinely examine and diagnose the images they receive in addition to the data they must continuously process. We wanted to create a model that can assist radiologists in the process because that takes a lot of time. Given the abundance of available datasets and the 30 features from which we might extract information, we decided on breast cancer. I concentrated on the dataset analysis that we coded into our notebook and the investigation of the entire procedure. After finishing the entirety of the code, we uploaded our notebook to github and worked on the presentation that we gave to the entire class.