A Comparative Analysis for Classifying Breast Cancer Tumors

Patrick Leary* Davidson College

Jack Schwanewede† Davidson College

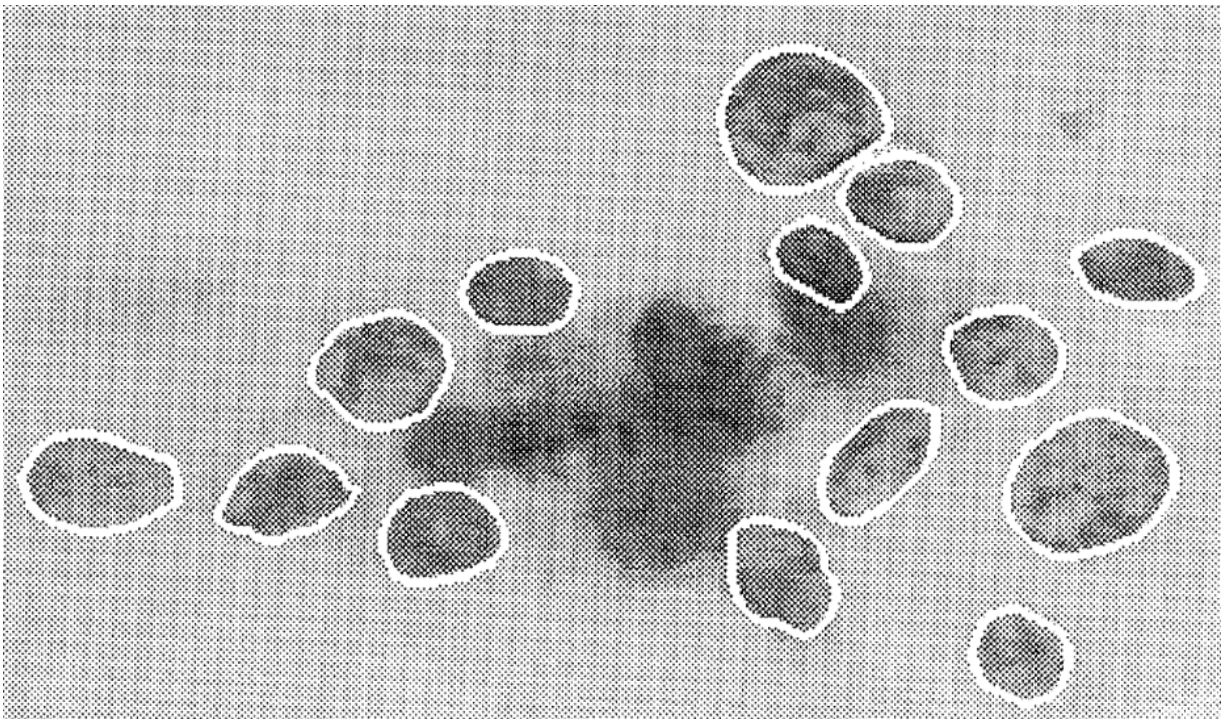

Figure 1: Cell Nuclei Edge Detection [3].

ABSTRACT

In this paper, we explored tumor data from the University of Wisconsin containing features of 569 breast cancer patients to create a neural network binary classification model. The goal of this model is to determine whether a tumor is benign (non-cancerous) or malignant (cancerous). This project produced two models: one with a 93.9% accuracy and the other with a 96.5% accuracy, similar to the 97.3% accuracy seen in Street et al. [3]. We created two neural networks, one using tools from the Python machine learning library PyTorch, and the other built from the ground up using the JAX library. Due to its superior optimizations and ease of use, the PyTorch model performed better than the JAX model, with lower loss and higher precision, recall, and accuracy. Additionally, The PyTorch model was more complex, containing an extra hidden layer with more nodes compared to the JAX model. This project highlights how deep learning can benefit medicine and healthcare by improving diagnosis accuracy in breast cancer patients.

*e-mail: paleary@davidson.edu

†e-mail: jaschwanewede@davidson.edu

Index Terms: Deep Learning, Neural Networks, Binary Classification, Breast Cancer.

1 Introduction

Deep learning has revolutionized medical diagnostics, offering new ways to detect diseases with improved accuracy. Neural networks—mathematical models comprised of input layers, multiple hidden layers, and output layers—have shown unprecedented precision in binary classification tasks. These models can correct errors and help medical professionals locate ailments faster, significantly reducing the time to treatment.

In this paper, we present a comparative analysis of two neural network models designed to classify breast cancer tumors as benign or malignant based on extracted tumor features. Our first approach implements a neural network from the ground up using JAX, and the second model leverages the PyTorch library. We hypothesized that both models would show excellent accuracy in classifying between malignant and benign tumors, though the PyTorch model would outperform the JAX implementation due to the library's optimizations.

2 Data

2.1 Background

Our dataset comes from the paper *Nuclear Feature Extraction for Breast Cancer Diagnosis* published in 1993 [3]. Using convolutions for edge detection, these researchers extracted 30 features of breast cancer cell nuclei from 569 images of tumors. Only 3 of the 30 features were used for the researchers' model, and the goal of this feature extraction was to classify between malignant and benign tumors. Using linear programming techniques, the researchers achieved a ten-fold cross-validation accuracy of 97.3%, a significant improvement upon the past medical literature. At the time of the paper's release, their model was in use at the University of Wisconsin hospitals.

2.2 Exploration

Each row in our dataset contains 32 columns with no missing values. These rows are comprised of the ID, the diagnosis (outcome), and 10 features separated into their mean, standard error, and 'worst' value. These features are:

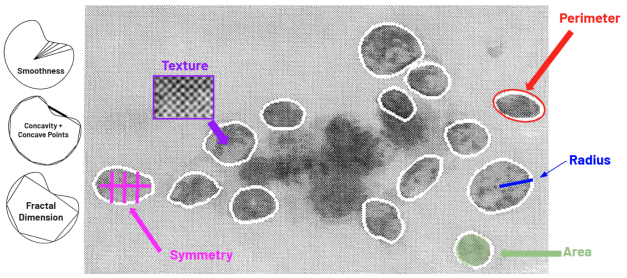

Figure 2: Visualization of features, used in our Data Exploration [3]

- **Radius:** Mean distance from the center of the cell to the perimeter.
- **Perimeter:** Perimeter of the tumor.
- **Area:** Area of the tumor.
- **Compactness:** How dense the tumor is (calculated using Perimeter^2 / area - 1.0).
- **Smoothness:** Local variations in radius lengths.
- **Concavity:** Severity of the concave points.
- **Concave Points:** Number of concave points.
- **Symmetry:** Symmetry of the tumor.
- **Fractal Dimension:** "Coastline approximation" - 1.
- **Texture:** Standard deviation of the tumor's gray-scale values.

These 10 features are represented visually in Figure 2.

212 of the 569 data points are benign and 357 are malignant. With approximately 37% benign and 63% malignant samples, this dataset is moderately unbalanced, but further testing showed that class balancing was not required to achieve a well-performing model.

In our exploration, we also created a feature importance chart to see what was most correlated with different diagnoses as seen in Figure 3. The features most correlated with malignant diagnoses were the worst concave point, the worst perimeter, the mean concave points, and the worst radius worst.

2.3 Preprocessing

Because this dataset comes from a preexisting paper, the dataset was already cleaned and had no

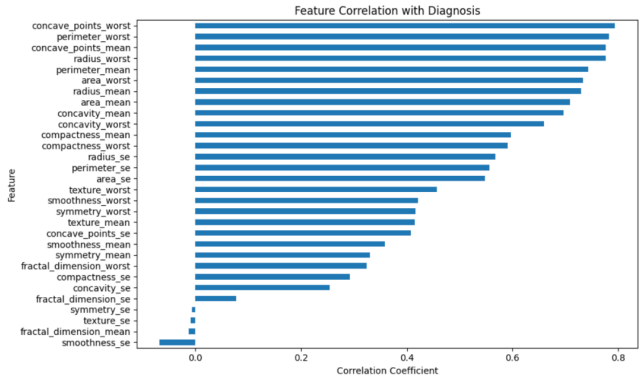

Figure 3: Correlation of Features To Diagnosis

missing values. We added column names to distinguish between features, and we used the standard scaler to normalize all the values. Finally, we split the data into training, validation, and testing data with a 64-16-20 split.

3 METHODS

In order to properly learn how neural networks function, we created two neural networks: one using JAX and the other implemented with PyTorch. Creating a model from the ground up with JAX allows us to understand the math behind a neural network, and PyTorch lets us learn a popular library used in industry to create a more performant network. While both models are neural networks, their structures were significantly different; this was due to some limitations in our JAX code as well as the ease of use of PyTorch. The end goal of both of these models was to output a single prediction score of 0, which indicated a benign tumor, or 1, which indicated a malignant tumor.

3.1 JAX

For the first model, we built a neural network from the ground up using JAX in Python. All 30 features were included in the input for the neural network, which consisted of the input layer and one hidden layer with 10 nodes. We initially created a more complex model, but the structure of 10 nodes in the hidden layer proves to be effective in distinguishing between the two classes. Both the hidden layer and the output layer use the sigmoid activation function, and we used binary cross entropy as the loss function. Lastly, to train this model, we implemented gradient descent by hand, using a learning rate of 0.1, and it trained 1000 epochs.

3.2 PyTorch

Implementing the neural network model in PyTorch proved to be much simpler than using JAX. We no longer had to define our own functions, which meant we could more quickly test different networks. We initially created a model with four hidden layers with 64 nodes, 256 nodes, 64 nodes, and 32 nodes. Each hidden layer used ReLU as their activation function, and the output used the sigmoid activation function. However, after initial testing, we simplified the model and used only two hidden layers of sizes 256 then 64 nodes, and we changed the activation function to sigmoid. Because developing the neural network in PyTorch was so much simpler, we were able to add additional features to the network. We used the Adam optimizer to improve results, and we introduced L2 regularization with a lambda of 0.01 to help combat overfitting with this larger model. Additionally, we lowered the learning rate to 10^{-4}, and trained the model for 2000 epochs. These enhancements over the JAX model led to improved results for the PyTorch network.

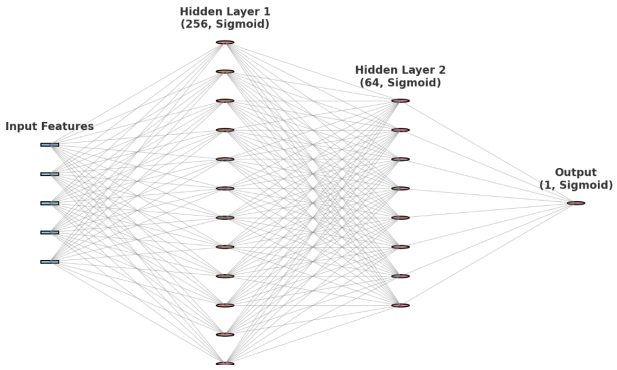

Figure 4: Visualization of our PyTorch Neural Network, created using ChatGPT.

4 RESULTS

Both neural networks provided high-accuracy results, with the JAX implementation achieving 93.9% accuracy and the PyTorch model reaching 96.5%.

4.1 JAX

The JAX model showed high levels of loss, even after thousands of epochs. The validation loss of the model was 0.40. However, the model performed favorably on the testing dataset, obtaining an F1 score of 0.92 and accuracy of 93.9%. In our confusion matrix below (see Figure 6), we ended with seven false predictions, four of which were false negatives and three false positives.

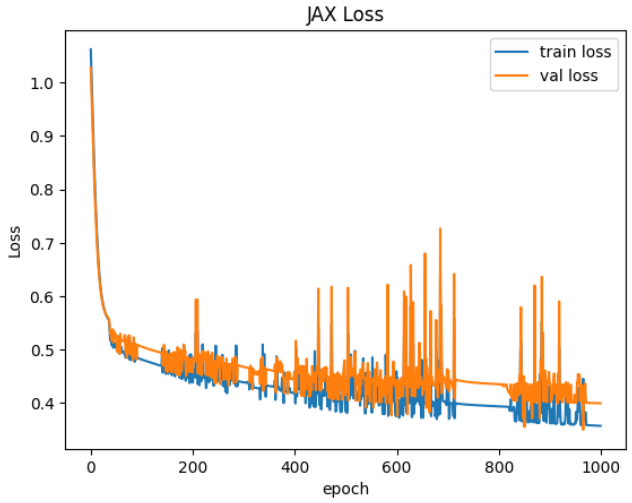

Figure 5: JAX Model Loss

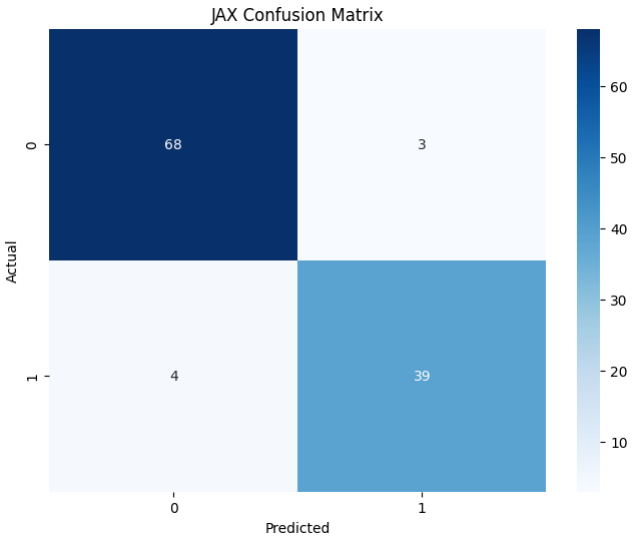

Figure 6: Confusion Matrix for Diagnoses in the JAX Model- 0 benign, 1 malignant

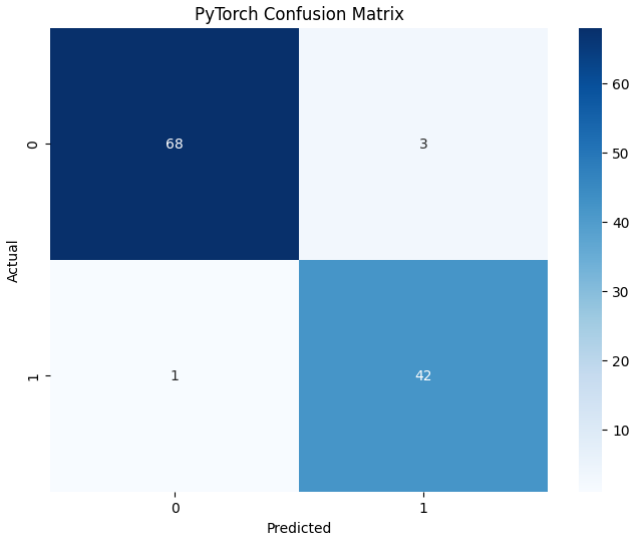

Figure 7: Confusion Matrix for Diagnoses in the PyTorch Model(with regularization) - 0 benign, 1 malignant

4.2 PyTorch

For our PyTorch model, we calculated the loss for models with and without regularization. In the model with regularization, there was a final training loss of 0.144, and a validation loss of 0.167. In comparison, without regularization our loss was 0.09 for training and 0.173, indicating that the model was overfitting. Because regularization mitigated this, we continued using the regularized model for our other measures, such as the confusion matrix. In this confusion matrix seen in Figure 7, we had only four incorrect diagnoses, with one being a false negative and three as false negatives. This resulted with an accuracy of 0.9649, precision of 0.9333, recall of 0.9767, and finally an F1 score of 0.9545.

5 Analysis

5.1 Activation Function

Originally, we planned on using ReLU as our activation functions. However, due to our networks' small number of hidden layers, we opted to use sigmoid instead. This change saw slight improvements over the ReLU model, so we kept the change.

5.2 Diagnosis Threshold

Another hyperparameter we manipulated was the threshold at which the models would decide whether a tumor was benign or malignant. However, manipulating the threshold of the JAX model beyond 0.2 resulted in the same F1 score of 0.9176. On further analysis, we determined that this was because most of the JAX predictions hovered around a value of 0.175. This may be due to the model's simplicity. because the network is only made up of one hidden layer with 10 nodes, it cannot create a large distribution of values. As seen in Figure 8, we see no predicted values between 0.2 and 0.8. This is why we see such a drastic change in model performance when changing the threshold past a certain limit.

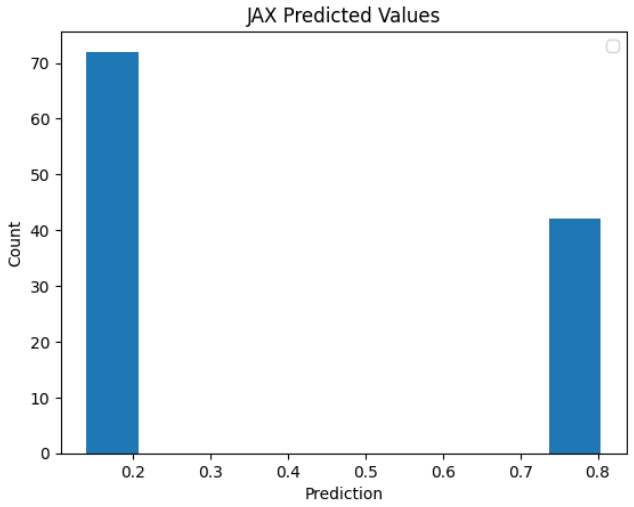

Figure 8: JAX Model Predictions

Model	Threshold	F1 Score
JAX	0.3	0.9176
JAX	0.4	0.9176
JAX	0.5	0.9176
PyTorch	0.3	0.9545
PyTorch	0.4	0.9412
PyTorch	0.5	0.9398

Table 1: Effect of threshold on model performance

5.3 Overfitting & Regularization

Due to its complexity, the PyTorch model began to overfit after about 2000 epochs. To reduce this overfitting, we implemented L2 regularization by adding the weight decay parameter to the optimizer. This regularization penalizes large weights, which helps combat overfitting. As shown in Figure 9, the validation loss begins to increase after epoch 2000, while Figure 10 shows how the weight decay allows for more stable validation loss at higher epochs. This addition to the model allowed it to train for longer, providing lower validation loss values.

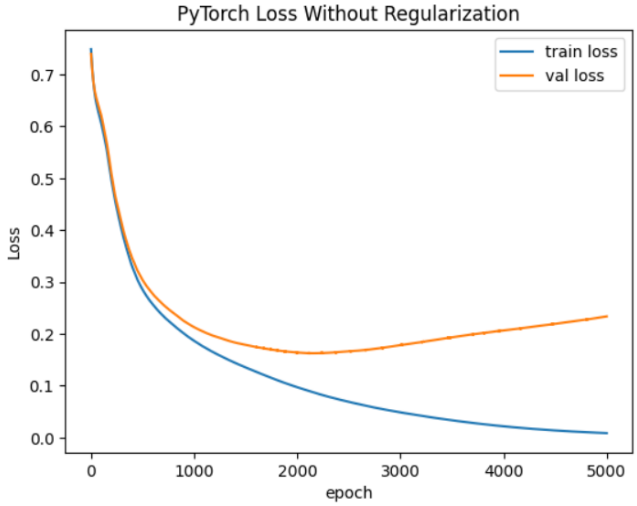

Figure 9: PyTorch Model Loss Without Regularization

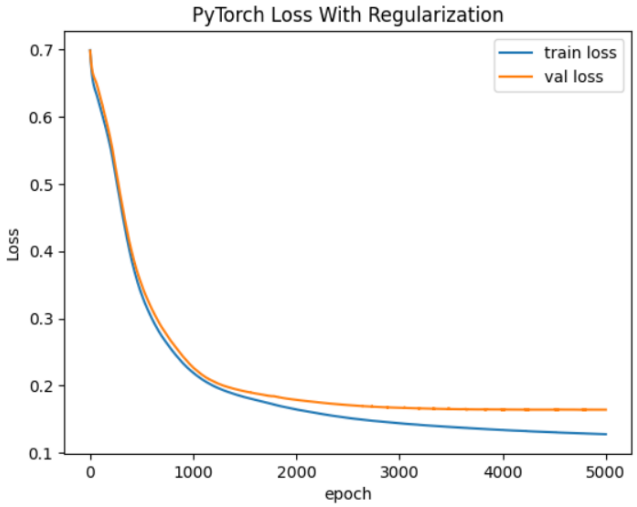

Figure 10: PyTorch Model Loss With Regularization

6 Discussion/Limitations

6.1 Data Quality

This dataset comes from extracted features from hundreds of images. While these features are extremely high quality, helping produce models with over 97% accuracy, they lose information when condensed down from the original image. A more comprehensive approach to this problem would be to feed the entire image to the neural network. In this scenario, the model would have even more information to base its decisions off of, and may find patterns not seen by the researchers creating the feature set. This stronger approach would implement a convolutional neural network (CNN) on the original images. This would allow the neural network to learn the best convolutions to use to extract the necessary features from each image, allowing for more accurate results.

6.2 Ethics

There are real implications of these models. For example, models using this dataset as created by Street et al. were used in University of Wisconsin Hospitals, as of 1993 [3]. If the model in practice led to false negative diagnosis, patients would be told that their tumor was benign when it was really malignant. This would mean that their cancer was undetected, and left untreated it could progress towards more serious symptoms and higher stages of cancer.

The other incorrect diagnosis, the false positive, is a more favorable result. In this case, the patient is told that they have cancer, but in reality their tumor is benign. This would likely result in stress, but after further testing, no cancer would not be found and treatments would not progress. Thus, there are significant impacts in the type of misdiagnosis the model makes.

There are also ethical concerns for how the effectiveness of the model is measured. Using the 'accuracy' of the model might not be the best metric. Say the model was tested on a random sample of $1,000$ people, of which 17 had breast cancer. If the model told every patient that they did not have cancer, it would have an accuracy of 98.3 percent, which appears high. However, in this example the model only predicts benign tumors, resulting in 17 false negatives where the cancer goes undetected. It is for this reason that these models should use other metrics such as the f1 score, precision and recall in order to see whether the model is truly predicting what it should be.

This data should also be confidential so that patient privacy is protected. This data should also only be used with consent, as the subject matter of the data is sensitive information. Of course, in an applied setting doctors should be able trace back diagnoses to IDs in order to report the results to patients. However, there should be clear consent from patients in order to use their data for training a classification model. With this consent, we conclude that there are low risks in using this data for the purposes of our project. The IDs are anonymous and cannot be traced back to patients, and their data is being used to ideally benefit other patients with tumors and improve diagnostic measures.

Finally, there are possible ethical concerns with the demographics of the original data and whether this resulting model would be truly representative and generalizable to other populations. Obermeyer et al. discuss how algorithms used in healthcare can have bias due to their training data, causing negative impacts on patients who are minorities [2]. One example of this was in an algorithm that calculated a comprehensive 'risk' score that measures health and short/long term mortality [1]. However, in this algorithm, Black patients had worse health and sickness conditions compared to white patients at the same calculated risk score. This means that the model was reporting Black patients to be healthier than they actually were, potentially limiting the amount of treatment and care they were receiving. This example highlights how algorithms, and models such as the one we created, may perpetuate bias, and emphasizes how medical algorithms should serve all populations and use diverse data to improve treatment.

7 Conclusion

Overall, our models demonstrate how neural networks can be used in binary classification tasks to increase accurate diagnoses in the medical field. Using our JAX model, we had 93.9%, with four false negatives. Using a more sophisticated library with PyTorch improved this, with an accuracy of 96.5%, and only one false negative. As discussed in the ethics section, it is important that in actual practice that these models mitigate false negatives as much as possible in order to prevent breast cancer from developing unchecked. Our results have only a single false negative out of 569 data points in the PyTorch model, indicating that it can be fairly effective in classifying between malignant and benign tumors while having lower risks from misdiagnoses. We conclude that neural networks can be an effective tool for classification tasks, especially in medicine and improving diagnostics, and has the potential to become more accessible and available to all.

Figure Credits

Figure 1 is an image of breast cancer cell nuclei grouped using an edge detection model taken from Street et al. [3].

References

[1] J. J. Gagne, R. J. Glynn, J. Avorn, R. Levin, and S. Schneeweiss. A combined comorbidity score predicted mortality in elderly patients better than existing scores. *Journal of clinical epidemiology*, 64(7):749–759, 2011. 6

[2] Z. Obermeyer, B. Powers, C. Vogeli, and S. Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019. 6

[3] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, vol. 1905, pp. 861–870. SPIE, 1993. 1, 2, 6, 7