Detecting Mycobacterium Tuberculosis from Smear Microscopy

Joshua Foster jfoste81@charlotte.edu
Sophia Godfrey sgodfr13@charlotte.edu
Adam Brewster abrewst2@charlotte.edu
Landon Amick lamick1@charlotte.edu

GitHub: https://github.com/jfoste81/Mycobacterium-Tuberculosis-Classifier

I. Introduction and Motivation

Tuberculosis (TB) is a disease that affects millions of people every year, especially in poorer countries that do not have strong healthcare systems. Because TB is one of the top causes of death from infections, early diagnosis is crucial for patient treatment and preventing further spread of the disease. Unfortunately, many places do not have enough trained doctors or lab workers to diagnose it quickly.

TB is most commonly diagnosed through smear microscopy. A doctor or lab technician takes a sputum sample and stains it so that Mycobacterium tuberculosis bacilli appear as red rod-shaped structures against a blue background. This process is very time-consuming, and successful diagnosis depends on the skill of the technician. Many of the technicians have high workloads and fatigue which can lead to mistakes, delaying treatment, and allowing the disease to spread further.

Using machine learning can increase successful diagnosis and decrease diagnosis times. Algorithms trained on medical images can analyze samples quickly and consistently to reduce the chance of human error. Once trained, it can look at images and decide if TB bacteria are present or not. These automated systems can eventually be scaled to clinics without trained specialists, provide reliable results, and allow treatment to begin earlier than normal diagnosis. These systems can also reduce the burden on healthcare workers by making it possible to start treatment earlier, which is important because more advanced strains of TB are significantly harder and more expensive to treat.

The goal is not only to identify the most effective model for TB detection but also to demonstrate how different machine learning approaches can complement one another. Comparing these models will show the strengths and weaknesses of each approach, giving more insight into how machine learning can assist in medical diagnosis in the future. By the end of the project, the aim is to demonstrate that machine learning can improve TB diagnosis and make it more consistent and accessible in regions where testing is most needed.

II. Approach

The dataset of smear microscopy images is first divided into two parts: 80% of the images are used for training and 20% are used for testing. A Convolutional Neural Network (CNN) is trained directly on the raw image data to learn patterns that show whether TB bacteria are present or not.

Once the CNN is trained, it also produces feature vectors that describe important details from each image. These features are then used as input for other machine learning models, including Support Vector Machines (SVM), Random Forests, and Logistic Regression. This allows the classical models to work with the same data in a way they can understand.

The performance of all models is measured and compared using accuracy, precision, recall, and F1-score. Special focus is placed on minimizing false negatives, since missing a TB-positive case could mean the patient does not get treated and continues to spread the disease. By comparing results across models, the approach shows which algorithm performs best and how different methods can complement each other in TB detection.

III. Dataset and Training Setup

The dataset used in this project contains 31,484 smear microscopy images, published by Dr. Marly Costa, that are designed for detecting pulmonary tuberculosis [1]. Each image is small, with a size of 40×40 pixels and three-color channels (RGB), resulting in 4,800 features. When all images are combined, the dataset holds a total of 151,123,200 features.

Feature extraction is a method of taking an image and breaking it down into important details that a computer can understand. Instead of looking at the whole picture, the computer focuses on certain parts, like colors, shapes, or textures. These details are turned into numbers, making it easier for machine learning models to process the data.

The data was run through Sci-kit Learn standard scaler [2] to regularize the data for Logistic Regression and SVC. Libraries used in this analysis included NumPy [3], Pandas [4], MatPlot [5], SciPy [6], Seaborn [7], and TensorFlow [8].

IV. Methodology

A. Convolutional Neural Network

To identify the optimal architecture for tuberculosis detection, two distinct deep learning architectures were evaluated: a custom sequential CNN and a residual network

(ResNet-10). Both models were designed to process the 40x40 pixel input patches.

The baseline model was constructed as a sequential feature extractor followed by a dense classification head. The feature extractor consists of three convolutional blocks. Each block contains a Conv2D layer (with 32, 64, and 128 filters respectively), a Rectified Linear Unit (ReLU) activation function, and a MaxPooling2D layer to reduce spatial dimensions. The resulting feature maps are flattened into a 1D vector and passed to a dense layer of 128 neurons. To dissuade overfitting, a 50% dropout regularization layer was inserted before the final single-neuron output, which uses a Sigmoid activation function to output a probability score [0,1].

A transfer learning approach was applied to bridge deep learning and classical machine learning. After training the baseline CNN, the final classification head was truncated. The pre-trained model was then used as a feature extractor, converting raw image pixels into 128-dimensional feature vectors. These dense vectors serve as the input for the remaining classical models, allowing them to train on the learned semantic features rather than raw pixel data.

To address potential degradation in deeper networks and improve training stability, a ResNet-10 model was implemented. Unlike the sequential CNN, the architecture utilizes skip connections that allow the gradient to flow through the network more effectively during backpropagation. This helps to minimize the vanishing gradient problem.

The implemented ResNet-10 consists of an initial convolutional block followed by four residual stages with filter sizes increasing from 64 to 256. Global Average Pooling was utilized instead of flattening to minimize parameter count and reduce risk of overfitting.

A grid search experiment was conducted to determine the optimal regularization strategy. Both the Custom CNN and the ResNet-10 were trained across a range of dropout rates [0.1, 0.2, 0.3, 0.4, 0.5]. To ensure efficiency, early stopping with a patience of 3 epochs was employed.

Additionally, to address the dataset imbalance (2,636 positive samples vs 28,848 negative samples), a class weighing scheme was implemented for training. A penalty factor of 4.0 was assigned to the positive class, which forces the loss function to treat a single missed positive case as equivalent to four missed negative cases.

B. Support Vector Classification [9]

A Support Vector Classification (SVC) was selected to compare results with a classical ML algorithm and to see if the complexity of a CNN/ResNet was excessive for the task.

Four kernels were run on the extracted feature data (Figure 1), including Linear (Figure 1.A), Polynomial (Figure 1.B), Sigmoid (Figure 1.C), and Radial Basis Function (RBF) (Figure 1.D). Due to the data provided by the CNN, the accuracy of all models was high. Because the data is used to classify life threatening contagious disease, the penalty was set to a high value of 1000 to make the model consider every data point a high priority. This allows the model to have the best chance of maximizing its F1-score by minimizing false positives and false negatives.

C. Random Forest

The Random Forest model was built with 100 decision trees, which gave the model enough variety to make accurate predictions and give the best results without overfitting. Each tree looks at a random part of the data, and then all the trees vote together on the final answer. More trees can make the model more stable, but too many trees can have slower processing times and provide a similar result. It was determined that 100 trees resulted in a good balance between accuracy and speed. The model also used a balanced class weight to handle the fact that there were fewer TB-positive samples than TB-negative samples. A random state of 42 was used to make the results reproducible.

D. Linear Classification

Logistic Regression was implemented as a core binary classifier. This linear classifier predicts probabilities through its use of the Sigmoid (or logistic) function. This function is essential because it maps the continuous output of the linear equation ($w \cdot x+b$) into a probability score strictly bounded between 0 and 1, representing the likelihood of an image belonging to the positive class (TB-positive). The model's training utilized the principle of maximum likelihood estimation, which minimizes the log-loss (or binary cross-entropy) function. To ensure robust training and convergence, the Limited-Memory Broyden-Fletcher-Goldfarb-Shannon (lbfgs) solver was chosen, with a maximum iteration limit set to 5000 (*max_iter*=5000). The model applied default L2 regularization, maintaining the inverse regularization strength C at 1000.

V. Results

A. Convolutional Neural Network

	Model	Dropout	AUC Score	F1-Score	Accuracy
1	ResNet-10	0.2	0.9994	0.9735	0.9956
2	ResNet-10	0.3	0.9995	0.9716	0.9952
3	ResNet-10	0.4	0.9990	0.9615	0.9935
4	Custom CNN	0.3	0.9978	0.9534	0.9921
5	Custom CNN	0.1	0.9968	0.9495	0.9914

Table 1: Top 5 Performing Neural Networks.

The experiment across five dropout rates revealed a distinct performance gap between the two architectures in terms of stability, detailed in Table 1.

The ResNet-10 demonstrated incredible robustness. Across all dropout rates, the model maintained an F1 above 0.92 and an AUC above 0.996. Its peak performance came with 20% dropout, achieving an F1 of 0.9735 and an AUC of 0.9994. This suggests that the residual connections mitigate any risks of underfitting, even with regularization.

Conversely, the baseline CNN was extremely volatile, exhibiting competitive performance at lower dropout rates, (with an F1 of 0.9534 at 30% dropout) but suffered catastrophic failure at other dropout values. At 40% dropout, the CNN only had an F1 of 0.1481 and an AUC of 0.7728, which indicates that the model struggled to converge with the higher regularization. This instability disqualifies the baseline CNN for clinical deployment, as this setting requires a certain level of reliability and stability.

Based on the experimental results, the ResNet-10 (20% Dropout) was selected as the final model. This configuration offers an optimal balance of extremely high sensitivity and stability, as seen in Figure 1.

Model Configuration	AUC	F1-Score	Recall	Precision	Accuracy
ResNet-10 (Dropout 0.2)	0.9997	0.9789	0.9700	0.9900	0.9965

Figure 1: Top performing ResNet-10 Model Configuration.

B. Support Vector Classification

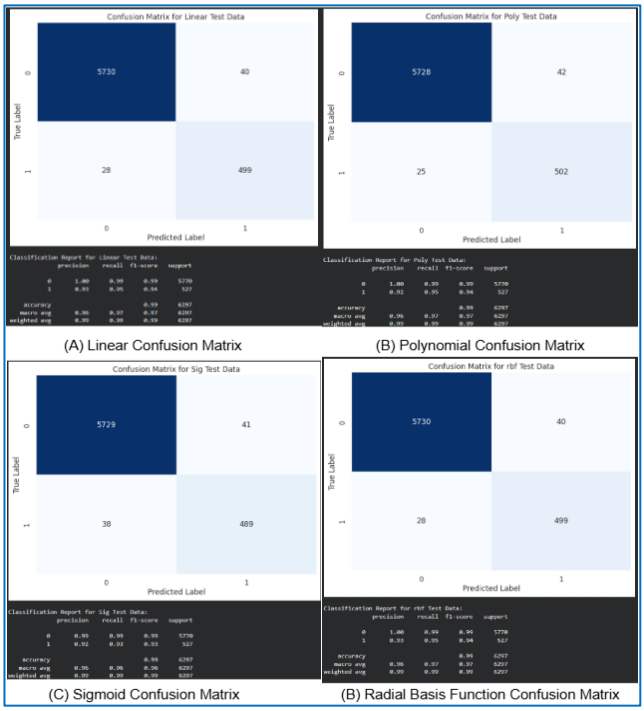

Figure 2: SVC Kernel Confusion Matrices.

Kernel Type	F1 Score (%)	Recall (%)	False Negatives
Polynomial	96.5	95	25
Linear	96.5	95	28
RBF	96.5	95	28
Sigmoid	96.0	93	38

Table 2: Kernel Rankings.

Given the intended focus of this study to minimize false negatives, this run shows that the Polynomial kernel leads to the best result. This information does change slightly between trainings as the feature extraction makes the margin of difference in models to be slim. Based on the results, the Linear, RBF, and Polynomial Kernels all exhibit similarly good performance, but the tiebreaker goes to polynomial as it has the fewest false negatives.

C. Random Forest

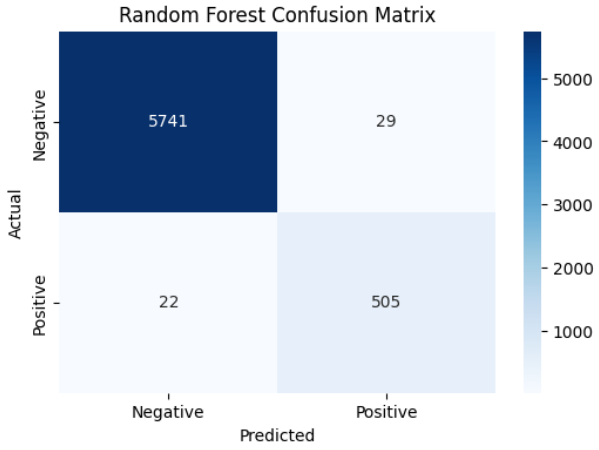

Figure 3: Random Forest Confusion Matrix.

When analyzing the confusion matrix, the model identified that only 29 TB-negative samples were incorrectly flagged as positive and failed to detect 24 TB-positive cases. The recall score of 0.96 highlights its strong ability to correctly identify true TB-positive cases. With 95% precision, the model avoids false alarms, showing how many of the predicted TB-positive cases were actually correct. Together, the 95% precision and recall produced an F1-score of 0.95, reflecting a balanced performance. Overall, the accuracy was 99%, and the AUC-ROC score was 0.9963, showing that the model was excellent at separating positive and negative cases.

D. Logistic Regression

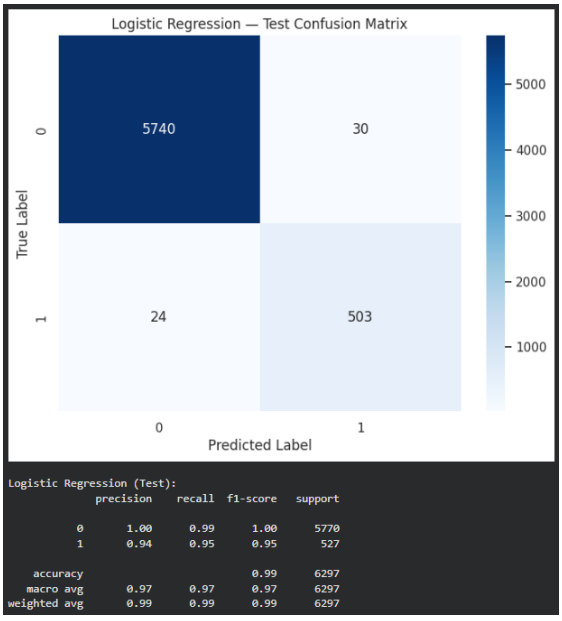

Figure 4: Logistic Regression Confusion Matrix.

The Logistic Regression model demonstrated robust performance in classifying Tuberculosis (TB) cases, achieving an overall Accuracy of 99%. The model showed high effectiveness in identifying positive cases, yielding a strong Recall (Sensitivity) of 96%. Furthermore, the model maintained a high degree of certainty in its positive predictions, achieving a Precision of 94%, which indicates a minimal rate of false alarms. The model's balanced performance is indicated by an F1-score of 0.95.

Analyzing the confusion matrix, of the 5764 actual TB-positive samples, the model correctly identified 5740 True Positives while only missing 24 cases (False Negatives). Correspondingly, the model identified 503 True Negatives and generated only 30 False Positives.

While Logistic Regression was the lowest-performing model among the four investigated, its performance is still exceptionally high.

VI. Lessons Learned

The primary takeaways from this project include the difficulty of acquiring medical datasets in niche domains, the benefit of feature extraction on complex datasets, and the utility of the Random Forest Algorithm. Upon selection of this topic, we had not interacted with medical data in this medium before. This allowed us to learn where data can be found and how hard it can be to determine accurate and updated data. The next beneficial lesson was how helpful feature extraction is on basic algorithms. Feature extraction allows us to keep contextual pixel information when passing the images through a non-neural model. This allows the models to gain a better understanding of the information in the image beyond the base color values of the pixels. The Random Forest algorithm was new to us as this information wasn't able to be taught in the lectures. This algorithm helped us look at another powerful classification tool that gave us another perspective on ML techniques.

Acknowledgment

The authors of this study would like to thank Hamed Tabkhivayghan for providing the knowledge required to complete this report. We would also like to thank the University of North Carolina at Charlotte for providing access to the tools used in this study including Google Colab Pro, and Microsoft Office Suite. The authors would also like to acknowledge the use of Google Gemini and OpenAI ChatGPT for code debugging and IEEE publication assistance. This paper was written as a final project for ECGR-4105 Fall 2025 at the University of North Carolina at Charlotte.

References

[1] *M. G. F. Costa, "DDS1 - Dataset of patch images derived from sputum smear microscopy images for the development of bacillus detection methods". Figshare+, 15-Feb-2025, doi: 10.25452/figshare.plus.27209667.v1.*

[2] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[4] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Austin, TX, 2010, pp. 56–61.

[5] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[6] P. Virtanen et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," Nature Methods, vol. 17, pp. 261–272, 2020.

[7] M. L. Waskom, "Seaborn: statistical data visualization," Journal of Open Source Software, vol. 6, no. 60, p. 3021, 2021.

[8] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. 12th USENIX Symp. Oper. Syst. Des. Implement. (OSDI), Savannah, GA, USA, 2016, pp. 265–283.

[9] H. Tabkhi, "Lecture 6: Support Vector Machine," Class Lecture, Topic: "Introduction to ML," Dept. of Electrical and Computer Engineering, Univ. of North Carolina at Charlotte, Charlotte, NC, 2025.