An Analysis on the Effectiveness of Classifiers on Tuberculosis Identification

Arlies Valdespino

April 28, 2024

Abstract

This report presents an investigation into the effectiveness of various classifiers for tuberculosis (TB) identification from chest X-ray (CXR) images. Through comprehensive evaluation, I compared the performance of Stochastic Gradient Descent (SGD), Random Forest Classifier (RFC), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Convolutional Neural Network (CNN) models. My findings reveal that SVC demonstrates the highest precision score of 99.2%, making it suitable for minimizing false positive cases, while CNN achieves the highest recall score of 96.4%, indicating its efficacy in early TB detection and disease spread prevention. These insights contribute to the development of robust diagnostic tools for TB screening and surveillance, with implications for improving patient outcomes and public health interventions.

1 Introduction

Tuberculosis (TB) remains a significant global health challenge, with timely and accurate diagnosis playing a crucial role in disease management and control. Chest X-ray (CXR) imaging is a commonly used diagnostic tool for TB due to its accessibility and cost-effectiveness. In this study, I conducted a comprehensive evaluation of various machine learning classifiers to assess their effectiveness in detecting TB from CXR images.

The dataset [1] consisted of a diverse collection of CXR images, including both TB-positive and TB-negative cases. I explored the performance of several classifiers: Stochastic Gradient Descent (SGD), Random Forest, Support Vector Machine Classifier (SVC), Multilayer Perceptrons (MLPs), and Convolutional Neural Networks (CNNs). Each classifier was trained on a subset of the dataset and evaluated using standard metrics such as accuracy, precision, recall, and area under the ROC curve (AUC).

My results revealed variations in the performance of different classifiers for TB detection. While some classifiers achieved high accuracy and AUC scores, others demonstrated better precision or recall rates. Additionally, I observed differences in computational efficiency and scalability across the classifiers, with some models requiring more computational resources for training and inference.

Overall, my study provides valuable insights into the strengths and limitations of various classifiers for TB detection in CXR images. These findings have implications for the development of robust and scalable diagnostic tools for TB screening and surveillance, ultimately contributing to improved patient outcomes and public health interventions in the fight against tuberculosis.

2 Methods

The dataset used in this study was obtained from Kaggle [2], a popular platform for hosting datasets and data science competitions. The dataset consists of a large collection of chest X-ray (CXR) images, including both TB-positive and TB-negative cases. Each image was properly categorized into two different folders: one containing TB-positive cases and the other containing TB-negative cases. This, thus, became a binary classification problem where the negative class was labeled as "0" and the positive class was labeled as "1".

Prior to model training and evaluation, the dataset underwent preprocessing to ensure consistency and suitability for machine learning tasks. This included resizing the images to 256x256 pixels to cut down on computational time and also switching from a monochromatic (black and white) image to an RGB image to enhance the contrast and brightness of the images, which can vary due to different equipment being used to collect the CXR images. images were also scaled so that the pixels fell in some range between 0-1, with 1 being completely filled with color (white) and 0 being completely void of color (black). Furthermore, to train these binary classifiers, the shapes of the training and testing images were changed to be two dimensional. This could be done by taking the first dimension to be the amount of samples and the second dimension to be 256x256x3.

Of the 4200 total images - 3500 TB-negative and 700 TB-positive - 3360 of them were used for training and the rest was used for testing. This was an 80% - 20% split for three-fold cross validation. In instances where hyperparameters needed to be tuned, this was flexed into five-fold or three-fold cross validation depending on the computational scalability of the model. A note will be made in the results section. The following sections will cover the specifics of what was done with each model tested.

2.1 SGD Classifier

An SGD Classifier was built with no initial specification of hyperparameters. I fit the model and performed three-fold cross validation to avoid extra computational time. Results of initial hyperparametrization will be discussed in the results section. Thinking there was room for improvement, I decided to do some fine-tuning with the α hyperparameter (regularization hyperparameter) - ranging from 0.0001 to 0.01 in steps of 0.001. In this case, I used five-fold cross validation and a grid search to test for the best performing α. I avoided testing for different loss functions and learning rates since this would increase computational time. However, this is an obvious flaw in my study, which I would look to rectify in a future study of more classifiers. The results for the α hyperparameter will be provided in the results section as well.

2.2 Random Forest Classifier

A Random Forest classifier was built with no initial specification of hyperparameters. I fit the model and once again performed three-fold cross validation. To obtain high recall and high precision, I decided to allow for the probability scores to be above 45% for the positive class. That is, any probability score obtained from cross validation above this probability would be classified as TB-positive and anything below would be classified as TB-negative. This gave me the best performing metrics on the training data. With good results, I decided that fine-tuning the RFC would not be worth the computational effort for minimal improvement. Evaluating on the test set, however, led me to believe that the model had overfitted on the training data. Regardless, SGD and RFC had clear strengths and weakness that could be used for different tasks. A clear comparison and discussion of the two will be provided in the results section.

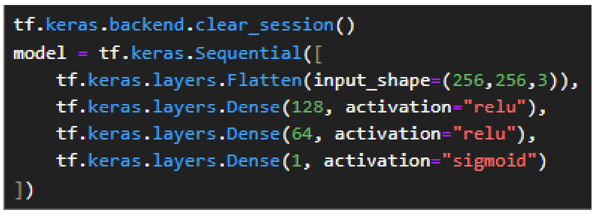
2.3 Support Vector Machine

I first decided to train a linear SVC using LinearSVC from scikit-learn. I needed to pass 10000 iterations for convergence. I fit the data, but did not proceed to tune the hyperparameters - to once again avoid unnecessary computational time. I did, in fact, train a regular SVC with no specified kernel, opting for only the default radial basis function (rbf) kernel. I once again proceeded to fit the data but avoided tuning hyperparameters for the same reason.

2.4 Multilayer Perceptron

Switching over to TensorFlow for deep learning, figure 1 shows the relatively simple build for the MLP. The number of neurons were chosen without loss of generality and to keep the computational time relatively low. I will note here that computational time for the MLP and its hyperparameter optimization were far better than that for SGD and the attempts for SVC.

The model was compiled using accuracy as its metric and "Adam (Adaptive Moment Estimation)" as its optimizer. I then fit the MLP with 12 epochs and using the test set as my validation set (the set of images were quite small so no need to



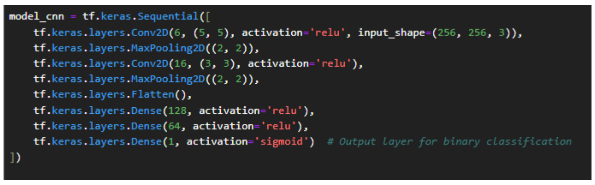
```
tf.keras.backend.clear_session()
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(256,256,3)),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(1, activation="sigmoid")
])
```

Figure 1: Construction of MLP

split into validation set). With relatively low accuracy and validation accuracy, optimization would only improve upon the results. Using the Keras tuner, I randomly searched for the best configuration of the number of neurons (ranging from 16 to 300) in each layer as well as the hidden layers (ranging from 0 to 8). The optimizers were chosen between "SGD" and "Adam". A total of 10 trials were used to find the best combination of all those hyperparameters with, once again, using the test set as our validation set. As expected, the optimized model had performed significantly better than the initial model.

2.5 Convolutional Neural Network

Figure 2 shows the complex build of the CNN. The number of filters in the convolution layers were chosen with respect to code found on the Kaggle dataset [2]. However, the kernel size was chosen by what I felt would produce the best results.



```
model_cnn = tf.keras.Sequential([
    tf.keras.layers.Conv2D(6, (5, 5), activation='relu', input_shape=(256, 256, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(16, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')  # Output layer for binary classification
])
```

Figure 2: Construction of CNN

The model was compiled similar to the MLP's hyperparameters. There was no hyperparameter tuning since the CNN had achieved good results.

3 Results

Table 1 summarizes the results from the initial run while table 2 summarizes the results from optimization, if applicable.

Classifier	Accuracy	Precision	Recall	AUC
SGD	0.981	0.962	0.921	0.957
RFC	0.964	0.951	0.829	0.980
LinearSVC	0.961	0.923	0.943	0.964
SVC	0.970	0.992	0.943	0.971
MLP	0.963	0.847	0.950	0.958
CNN	0.983	0.938	0.964	0.976

Table 1: Metrics for Initial Classifiers

Classifier	Accuracy	Precision	Recall	AUC
SGD	0.969	0.875	0.950	0.961
RFC	N/A	N/A	N/A	N/A
LinearSVC	N/A	N/A	N/A	N/A
SVC	N/A	N/A	N/A	N/A
MLP	0.979	0.984	0.886	0.941
CNN	N/A	N/A	N/A	N/A

Table 2: Metrics for Optimized Classifiers

I have highlighted the overall best performing models in table 1. Unfortunately, through optimization, SGD and MLP slightly worse in their respective strengths. For example SGD had originally achieved high precision, but with fitting it had decreased significantly. This is most likely due to overfitting on the training data when conducting GridSearch. Also, the α hyperparameter is not the most impactful. Due to computational time constraints, this was a clear weakness of the study. However, un-optimized SGD is still a good model if you want high accuracy and precision. MLP had performed significantly better in precision but decreased in recall. This is again most likely due to overfitting on the training data when conducting the randomized search. Also, MLP does not perform nearly as well as other deep language models (like CNN) due to its nature in not being scaled to images. Nevertheless, after optimization, it had beaten most of the other models in accuracy and precision.

Of the most noteworthy models, SVC and CNN, unsurprisingly, had performed the best among all four metrics. Notably, CNN had achieved the highest accuracy score and SVC had achieved the highest precision score. Therefore, if the main goal of a practitioner were to have low false positives (model identifies image to have TB when in reality it does not), then they would most likely prefer to use SVC. This could avoid unnecessary treatments and expenses. However, if a medical practitioner were to want low false negatives (model identifies image to not have TB when in reality it does), then they should prefer to use a CNN model. This could help detect TB early and prevent its spread.

4 Discussion and Conclusions

The results of my study offer valuable insights into the effectiveness of various classifiers for tuberculosis (TB) identification from chest X-ray (CXR) images. Through comprehensive evaluation, I compared the performance of Stochastic Gradient Descent (SGD), Random Forest Classifier (RFC), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Convolutional Neural Network (CNN) models.

My findings indicate that while each model exhibits strengths and weaknesses, both SVC and CNN stand out as the most promising classifiers for TB identification. SVC demonstrated the highest precision score of 99.2%, making it particularly suitable for minimizing false positive cases and avoiding unnecessary treatments. On the other hand, CNN achieved the highest recall score of 96.4%, suggesting its efficacy in detecting TB cases early and preventing disease spread by minimizing false negatives.

Despite the notable performance of SVC and CNN, it's essential to acknowledge the limitations of my study. I encountered challenges such as overfitting during hyperparameter optimization, especially for SGD and MLP models. Additionally, computational constraints limited the depth of my exploration into hyperparameter tuning, potentially leaving performance improvements untapped.

In conclusion, my study underscores the importance of selecting appropriate classifiers for specific diagnostic tasks in tuberculosis identification. While SVC offers precision-focused performance, CNN excels in achieving high overall accuracy. Future research should aim to address the limitations identified in this study and further refine the performance of classifiers for tuberculosis detection in CXR images.

References

[1] T. Rahman, A. Khandakar, M. A. Kadir, K. R. Islam, K. F. Islam, R. Mazhar, T. Hamid, M. T. Islam, S. Kashem, Z. B. Mahbub, M. A. Ayari, and M. E. H. Chowdhury, "Reliable tuberculosis detection using chest x-ray with deep learning, segmentation and visualization," *IEEE Access*, vol. 8, pp. 191 586–191 601, 2020.

[2] Tuberculosis (TB) Chest X-ray Database. Accessed: 2024-03-04. [Online]. Available: https://www.kaggle.com/datasets/tawsifurrahman/tuberculosis-tb-chest-xray-dataset/