ONLINE LEARNING FOR ACID-FAST BACILLI DETECTION IN HISTOPATHOLOGICAL IMAGES

By

Shizhao Wang

Bachelor of Science - Computer Science
University of Nevada, Las Vegas
2022

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2024

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 4, 2024

This thesis prepared by

Shizhao Wang

entitled

Online Learning for Acid-Fast Bacilli Detection in Histopathological Images

is approved in partial fulfillment of the requirements for the degree of

Master of Science – Computer Science
Department of Computer Science

Abstract

The acid-fast stain is frequently used for laboratory diagnosis of tuberculosis. It is a labor intensive task requiring thorough examination of extremely high-resolution images to pinpoint the presence of the mycobacteria. This paper presents a machine learning assisted slide image analysis tool with the aim of aiding histopathology professionals in the accurate diagnosis of tuberculosis in patients through the analysis of microscopic imagery. The proposed tool combines a digital whole slide image viewer with an online learning framework. We also conducted a survey of different state-of-the-art online learning methods, and found that MIR with pre-training has the best performance on the CIFAR-10 dataset.

Keywords: Histopathology, Online Learning, Tuberculosis, Acid-Fast Staining.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

TB	Tuberculosis
AFB	Acid-Fast Bacilli
ZN	Ziehl-Neelsen
WHO	World Health Organization
COVID-19	Coronavirus disease
WSI	Whole Slide Image
CNN	Convolutional Neural Network
OL	Online Learning
IDD	Independent and Identically Distributed
LwF	Learning without Forgetting
iCaRL	Incremental Classifier and Representation Learning
ER	Experience Replay
MIR	Maximally Inferred Retrieval
GDumb	Greedy Sampler and Dumb Learner

1. Introduction

The acid-fast staining test of sputum sample smears to detect acid-fast bacilli (AFB) is a commonly used method of diagnosing Tuberculosis (TB), recommended by WHO for its cost efficiency [1-2]. However, the process of diagnosing a slide image is notoriously very time consuming and labor intensive [2-3]. The traditional way of diagnosing TB through this method uses the Ziehl-Neelsen (ZN) staining. After obtaining the stained sample, analysis is an exhaustive task that requires skilled technicians to examine the slide under a microscope to identify AFB.

While TB is preventable and curable, it still remains a major global health concern to this day. The World Health Organization (WHO) stated that TB ranked as the second highest cause of global mortality following the Coronavirus (COVID-19) in 2022, causing 1.3 million deaths. [5]. Despite COVID-19 disruptions to social events causing a large decrease in the number of recorded TB cases globally, as the world recovers from the pandemic, TB cases have climbed back to reach previous levels [1].

Several studies have been made on diagnosing TB via automated detection of AFB on ZN stains. Earlier proposed methods use images captured with cameras over a microscope [6-7]. Ayas et al. used a SVM pixel classifier to segment AFBs, and Khutlang et al. proposed a random forest method that uses decision trees to classify segments. More recently, digital whole slide image (WSI) technology has gained popularity due to having increased fidelity in the captured images [4]. There have been studies using convolutional neural network (CNN) models for analyzing WSIs [8-9], but there is a lack of publicly available data for the task of detecting AFB in WSIs.

In this thesis, a tool is proposed with the aim of assisting histopathologists in analyzing WSIs. First we will discuss the design and implementation of the proposed framework, the frontend application and the technologies used, as well as the functionality of the backend server. Due to the lack of publicly available labeled data, it is not feasible to train an image classification model with traditional methods from scratch. Instead, we propose an online learning (OL) framework that can incrementally learn from a constant stream of incoming data. We conduct a survey of different state-of-the-art online learning techniques and present experiment results comparing their effectiveness.

2. Methods

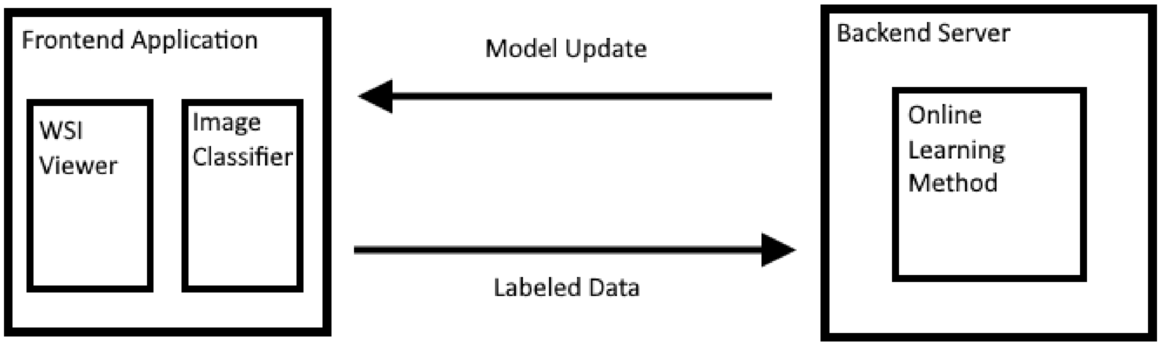


Figure 1. Overview of the proposed framework design.

The proposed OL framework contains two parts, the frontend user interface application and the backend OL model training and hosting server (Figure 1). The main purpose of the frontend interface is to be a lightweight WSI viewer that supports a variety of different vendor formats, with the capability to use a trained model to analyze WSIs. As the user provides labels for image patches through the frontend application, the labeled data is sent to a backend server to perform online learning. After training is complete, the server will distribute the updated model back to the frontend application, improving future analysis results.

During the ZN staining process, the acid-fast nature of AFB allows it to resist coloration from the blue dye, thus they remain a red color. A naive unsupervised algorithm to detect AFB is to simply look for pixels with higher red value. The application contains an implementation of an unsupervised clustering algorithm as a starting point before a model can be trained.

2.1 Frontend Application

Digital slide images are most often saved as an Aperio format SVS file, a format which is unsupported by most image viewers. The Aperio format is based on the TIFF format and utilizes the tiled image capabilities in order to display very large images. However, other vendors may use different formats. The OpenSlide library is designed to interface with a variety of different vendor formats [12]. Utilizing OpenSlide for reading slide images, another software, OpenSeadragon [13], is used to display the zoomable pyramid structured image.

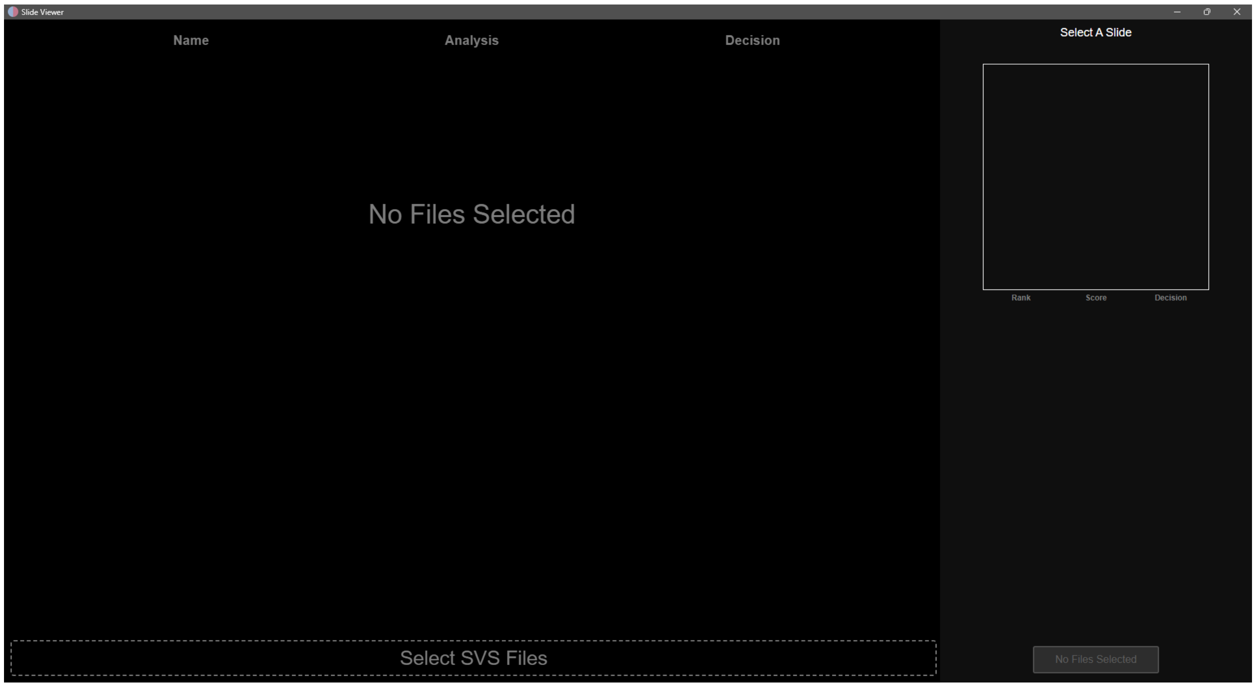
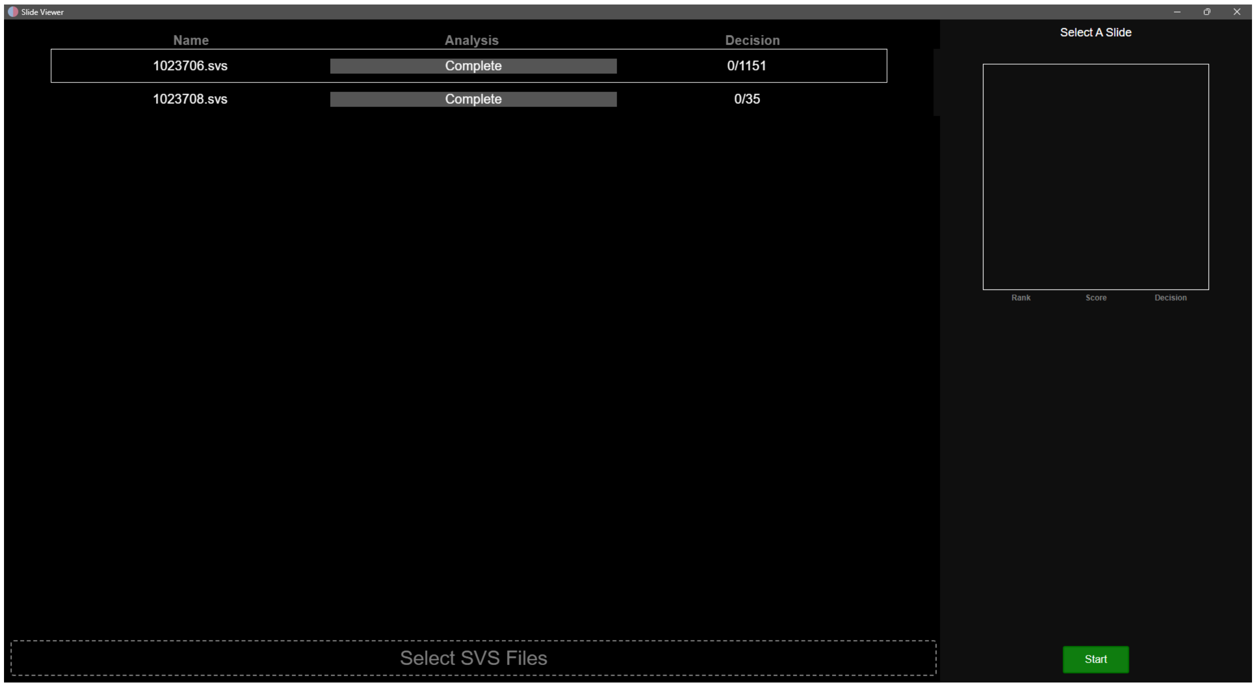


Figure 2. Slide file selection screen.

Upon starting the application, the user will be met with a message to select a slide image file to start (Figure 2). Clicking the “Select SVS Files” button will open a file selection dialog box.



Name	Analysis	Decision
1023706.svs	Complete	0/1151
1023708.svs	Complete	0/35

Figure 3. Slide file selection screen after multiple queued slide analysis.

When a slide is opened, the user can choose to analyze the slide with the provided model. Currently, large images at about 3 GB in size can take from 30 minutes to an hour to complete analysis depending on hardware. Because the operation on large WSIs can take a long time to finish, a queue is implemented such that the user can select and queue up multiple slide images, and the application will perform analysis for all of them in sequence (Figure 3). If a trained model is provided, the application will prioritize analysis using the model, otherwise, the unsupervised clustering algorithm will be used.

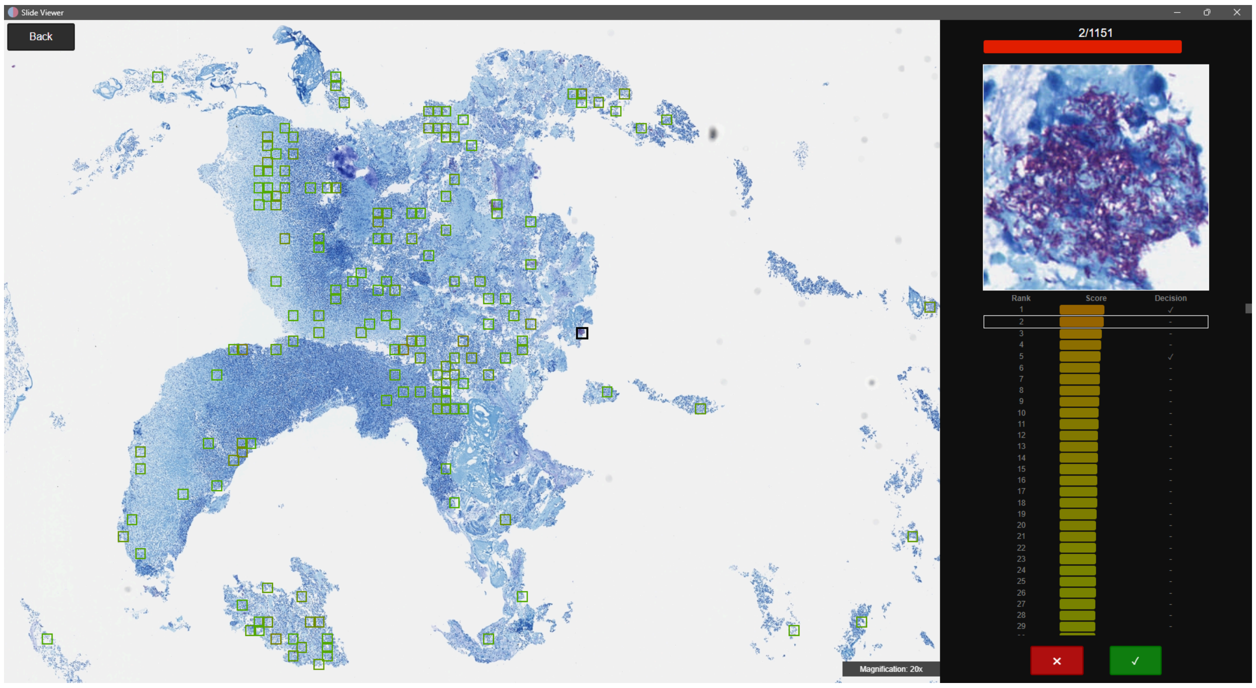


Figure 4. WSI viewer with analysis results.

After analysis, patches with high probabilities of containing AFB are presented to the user to make a decision (Figure 4). After a decision is made, the user labels the patch and the patch image along with the label is uploaded to the backend server[1].

1 Discussions with medical professionals suggest that directly sending segmented slide patch images with decision labels and no other metadata should not violate any patient privacy laws, as there is no identifying information. Additionally, it is not possible to extract any original training data from the trained AI model. However, hospitals should notify patients and ask for permission before using samples taken from them in training AI models. It is recommended to check with HIPAA and other related laws and regulations.

2.2 Server Backend

The backend server is written in Python, it can receive patch images along with a label when a histopathologist using the frontend application makes decisions about a slide. Multiple users of the frontend application are supported. When labeled data is received, the server will temporarily store it and use it to train the model using an OL method implemented through PyTorch. After training, the updated model will be saved, and future uses of the frontend application will receive the updated model.

3. Online Learning

Online learning techniques can keep updating and improving a model's performance by incrementally learning from new data as it becomes available, without needing to be retrained from scratch. The key differences between OL and traditional offline training lie in the assumptions that can be made about training data. In offline machine training, it is generally assumed that we have an acceptably sized dataset, the data is independent and identically distributed (IID), and the learner is allowed to learn through multiple passes of the entire training dataset. However, in OL, these assumptions cannot be made. Data may arrive intermittently through a continuous stream, it will likely not be IID, and aside from a limited memory buffer, the learner can only make one single pass through the arriving training data [14]. These harsh restrictions on the training data is one of the challenges of OL.

Since OL does not keep all of the training data, it is less computationally expensive compared to offline training, as well as having much less memory demands for large datasets. For potentially sensitive data, discarding old data after training is good for data privacy. Additionally, OL methods are designed to be able to generalize concepts better than traditional machine learning methods [10]. This opens up the possibility of future research in lifelong online learning where a model learns many different slide analysis tasks.

A major challenge in OL is the phenomenon coined Catastrophic Interference, the tendency of a model to lose previously learned information as it is trained with new information [15]. Many algorithms have been developed to combat this problem. We researched different baseline and state-of-the-art OL methods and their performance on the CIFAR-10 dataset in different situations with modifications on the data.

3.1 Related works

Learning without Forgetting (LwF) - LwF [16] is a non memory based method that combines fine tuning with knowledge distillation [17]. After the model learns some information, it is used as a "teacher" model which will supervise the next iteration of the model as a "student" model as it learns new information.

Incremental Classifier and Representation Learning (iCaRL) - iCaRL [8] attempts to learn the representations of classes. Using knowledge distillation to combat catastrophic interference, it uses the memory buffer to hold samples and trains a CNN feature extractor to classify images using the nearest class mean of each class.

Experience Replay (ER) - ER [19] is replay based method. It uses reservoir sampling [20] to ensure that the probability of each sample from the data stream to be stored in the memory buffer is equal, and trains the model on incoming data as well as randomly retrieved old samples from the memory buffer.

Maximally Inferred Retrieval (MIR) - MIR [21] is an extension of ER, instead of randomly retrieving samples from the memory buffer, it instead determines which samples to store and replay by determining the loss increase after estimating the parameter updates of the model.

Greedy Sampler and Dumb Learner (GDumb) - GDumb [22] greedily pulls samples from the data stream into the memory buffer, keeping a balanced distribution between all classes. Whenever a prediction is requested it trains a new model at using the samples stored in the memory buffer. Effectively, GDumb performs offline training over stored samples from a data stream.

3.2 Experiments

3.2.1 Experimental Results

We show the potential applications of different online learning methods to the AFB detection problem, using general image datasets. To assess the online learning performance, we trained the ResNet-18 [23] model, one of the most popular deep learning models, using several different state-of-the-art online learning techniques. Due to the lack of publicly accessible labeled AFB slide image datasets, we performed the training and evaluation using the CIFAR-10 [24] image dataset.

We conducted the experiments with five settings: (1) Training through the entire dataset as a continuous stream. (2) Training on only a subset of the total dataset as a continuous stream. Training is stopped at 10%, 25%, 50%, and 75% of the total dataset and the resulting model evaluated. (3) Pre-training an initial model on 25% of the dataset with offline training, then using another 25% of the dataset as a continuous stream. (4) Incrementally adding noise into the training images as the stream goes on. (5) Incrementally blurring the images as the stream goes on.

In the first test, we train each method over a data stream containing the entire dataset, compared with offline training for 20 epochs, which will be the baseline upper bound for the first three tests. Results are the average across 5 runs of each test. Memory buffer size on memory based methods is 5000 samples.

Table 1. Results after training from the entire CIFAR-10 dataset.

Method	Accuracy (over 5 runs)
Offline (20 epochs)	0.736 ± 0.009
LwF (Li. et al, 2017)	0.583 ± 0.025
iCaRL (Rebuffi. et al, 2017)	0.604 ± 0.016
ER (Rolnick. et al, 2019)	0.599 ± 0.051
MIR (Aljundi. et al, 2019)	**0.641 ± 0.051**
GDumb (Prabhu. et al, 2020)	0.601 ± 0.013

At the beginning of training, only very few samples of labeled data may exist. It is important for the model to quickly learn from a limited number of samples. Test two measures model performances as the number of samples increases in continuous batches.

Table 2. Results after training on only a subset of the data.

Method	10% of data	25% of data	50% of data	75% of data
Offline (20 epochs)	0.522 ± 0.045	0.631 ± 0.015	0.677 ± 0.007	0.715 ± 0.004
LwF	0.276 ± 0.028	0.334 ± 0.065	0.448 ± 0.049	0.508 ± 0.024
iCaRL	0.338 ± 0.019	0.396 ± 0.028	0.491 ± 0.015	0.546 ± 0.032
ER	0.285 ± 0.055	0.417 ± 0.02	0.485 ± 0.064	0.527 ± 0.109
MIR	0.303 ± 0.025	0.422 ± 0.013	0.518 ± 0.031	0.591 ± 0.048
GDumb	**0.601 ± 0.007**	**0.6 ± 0.01**	**0.583 ± 0.035**	**0.603 ± 0.008**

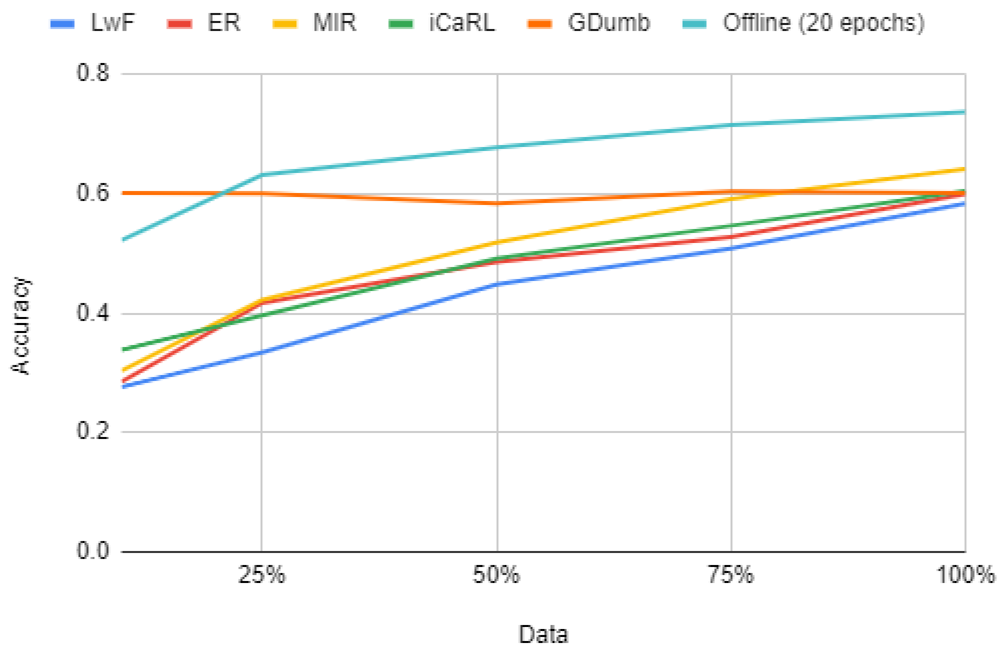


Figure 5. Results after training on different amounts of data.

If some labeled data already exists, then it is possible to use offline training to pre-train a model, then use OL on future data using the pre-trained weights. Test three shows how each method performs starting with a model trained on 25% of the dataset using offline learning, then on another 25% of the dataset using the respective OL method.

Table 3. Results using pre-training.

Method	Accuracy (over 5 runs)
Offline (50% data)	0.677 ± 0.007
LwF	0.683 ± 0.012
iCaRL	**0.689 ± 0.008**
ER	0.674 ± 0.018
MIR	0.678 ± 0.019
GDumb	0.661 ± 0.007

Considering the offline training accuracy over 50% of data after 20 epochs was 0.677, the pre-training test results are very impressive. All of the OL methods achieve similar to or even exceeding offline accuracy with pre-training. GDumb suffers in this test due to not being designed to take advantage of previous knowledge in the form of model weights. Inversely, LwF and iCaRL perform better in this test compared to tests without any pre-training, since these methods use regularization terms to preserve old weights better.

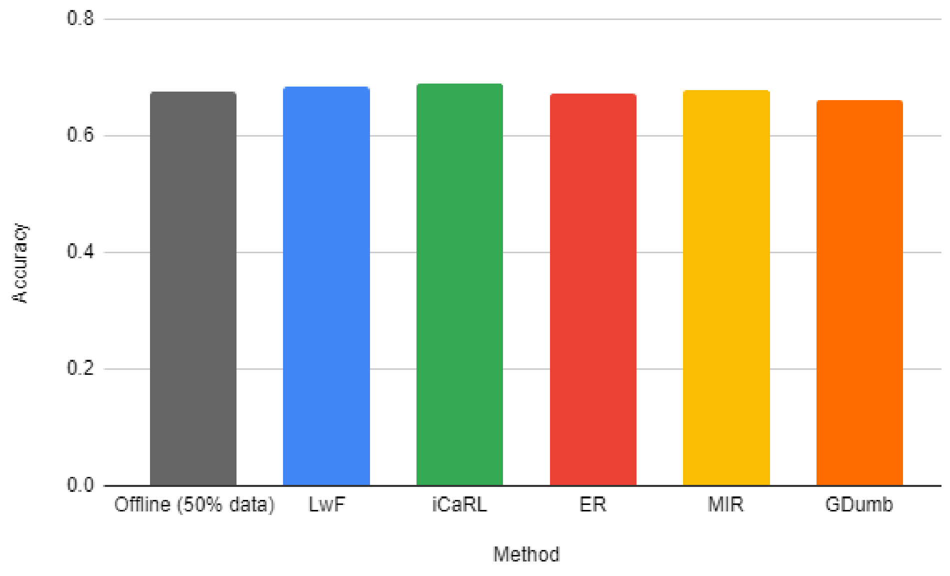


Figure 6. Results using pre-training.

If the staining method is not perfectly consistent, samples may be stained with varying quality. Tests four and five measures how methods perform as the data degrades from varying degrees of noise and blurriness respectively.

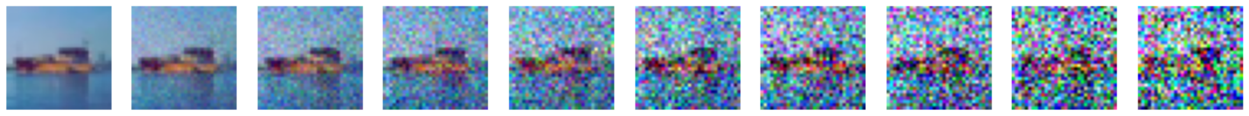


Figure 7. Incrementally adding noise to data.

Table 4. Results after training with incrementing amounts of gaussian noise to data.

Method	10% data	20% data	30% data	40% data	50% data	60% data	70% data	80% data	90% data
LwF	0.2624	0.3346	0.3621	0.4008	0.4133	0.4146	0.4170	0.4025	0.3870
ER	0.3069	0.2788	0.4348	0.4286	0.4200	0.5110	**0.5531**	**0.5430**	**0.5516**
MIR	0.2589	0.3133	0.3690	0.4340	0.4726	**0.5216**	0.5146	0.5444	0.4923
iCaRL	0.2725	0.2940	0.3920	0.4170	**0.4918**	0.5020	0.5290	0.5339	0.4804
GDumb	**0.4471**	**0.4460**	**0.4355**	**0.4440**	0.4388	0.4171	0.4293	0.3943	0.4300

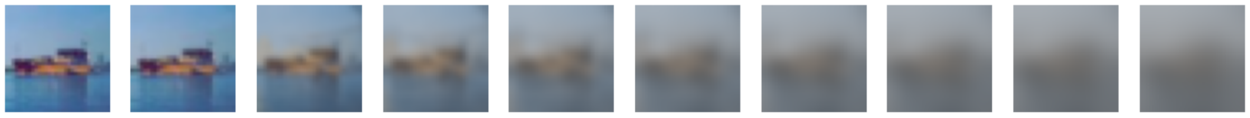


Figure 8. Incrementally adding blur to data

Table 5. Results after training with incrementing amounts of gaussian blur to data.

Method	10% data	20% data	30% data	40% data	50% data	60% data	70% data	80% data	90% data
LwF	0.1566	0.1759	0.1823	0.1911	0.1793	0.1816	0.1820	0.1770	0.1700
ER	0.1549	0.1636	0.1869	0.2194	0.2067	0.2011	0.2076	0.2252	0.1883
MIR	0.1687	0.1782	0.1464	0.2018	0.2021	0.2218	0.2149	0.2254	**0.2260**
iCaRL	0.1428	0.1428	0.1095	0.1568	0.1633	0.1868	0.1864	0.1857	0.1904
GDumb	**0.2103**	**0.2322**	**0.2239**	**0.2312**	**0.2311**	**0.2287**	**0.2315**	**0.2338**	0.2237

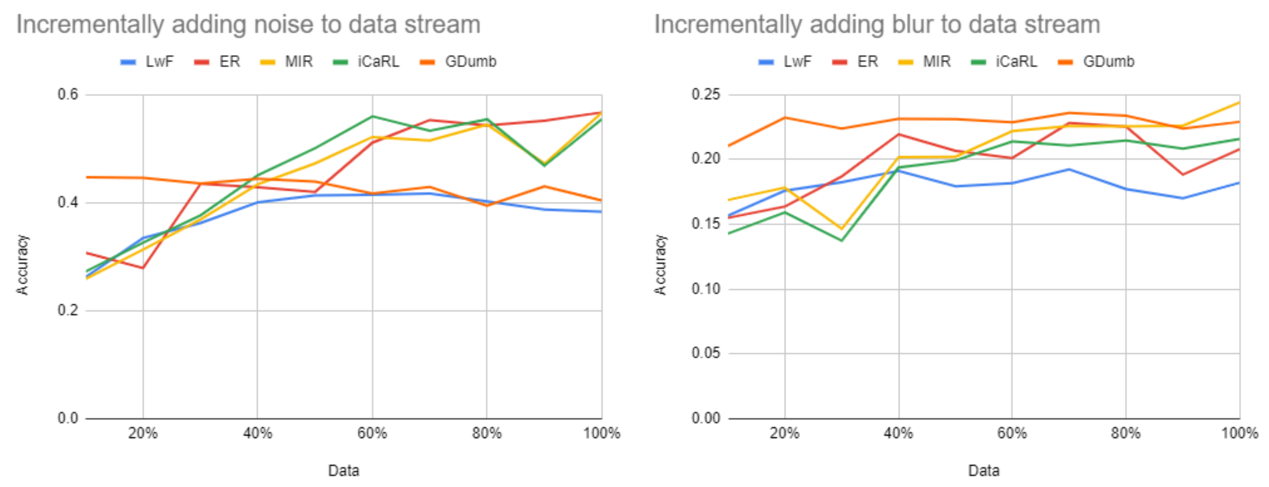


Figure 9. Results after incrementally adding noise and blur to the data stream.

In the data degradation tests, MIR performed most consistently well, and is a competitive and versatile method. ER has a similar trend but is generally worse than MIR due to its random retrieval method. However, ER is computationally more efficient than MIR. It is notable that blurring the image destroys identifying features much faster than adding noise, thus the accuracy of every model was significantly lower for each OL method, and some even regressed as it trained on blurry samples. To prevent this regression in accuracy, a simple solution would be to roll back to the previous model state if the accuracy decreases.

In these tests, LwF performs worse than the memory based methods with a 5000 sample memory buffer size. If memory constraints are tight and fewer samples can be stored by the memory based methods, LwF may perform more favorably by comparison.

While the results for GDumb are impressive, due to the method needing to retrain a new model at each inference point, it is very computationally inefficient. Another pitfall of GDumb is its greedy sample grabbing strategy, which prevents it from being able to continue improving after the memory buffer becomes full. In all of the tests, GDumb performs consistently with itself, with no noticeable improvement as more samples are made available, while other methods show increasing accuracy as they learn more data.

4. Conclusion

This study has presented a novel application designed to provide AI assistance to histopathologists in analyzing WSIs for detection of AFB. Utilizing OL methods to train models, Histolearn has the potential to incrementally improve its diagnostic model as new data becomes available through continued use. This capability addresses the challenge of the lack of publicly available labeled data at the outset.

Our evaluation of various OL methods under different conditions shows their applicability to potential real-world scenarios. It was demonstrated that while LwF offers benefits in memory-constrained environments, it fails to match the accuracy of memory-based methods. Conversely, while GDumb provides a surprisingly effective baseline in accuracy, it is computationally inefficient and is unable to improve past the allocated memory buffer bounds due to its data sampling strategy. Based on these findings, MIR shows the most promise as a highly competitive and versatile OL method.

4.1 Future works

Due to the lack of data, we have not evaluated the performance of the proposed framework on real AFB slide images. Future work can be directed at obtaining a sufficiently large set of labeled images. With the dataset, further testing can be performed on the effectiveness of pre-training.

To improve the effectiveness of learning on few labeled data, the OL framework may be modified with an active learning query method. Instead of presenting the user with patches that are most likely to contain AFB, the active learning query method will prompt the user to instead provide labels on the most uncertain patches, or the most informative patches based on estimated parameter updates.

OL methods tend to be able to generalize to multiple concepts well, another possible direction of future research is to utilize the proposed framework and extend the model prediction capabilities to beyond the detection of AFB, and transfer the knowledge to predicting other diseases.

References

1. Fu, Hsiao-Ting et al. "Evaluation of an AI-Based TB AFB Smear Screening System for Laboratory Diagnosis on Routine Practice." *Sensors (Basel, Switzerland)* vol. 22,21 8497 (2022).
2. Campelo, Thales Alves et al. "Revisiting the methods for detecting Mycobacterium tuberculosis: what has the new millennium brought thus far?." *Access Microbiology* vol. 3,8 000245 (2021).
3. Vodovnik, Aleksandar. "Diagnostic time in digital pathology: A comparative study on 400 cases." *Journal of pathology informatics* vol. 7 4 (2016).
4. Kumar, Neeta et al. "Whole Slide Imaging (WSI) in Pathology: Current Perspectives and Future Directions." *Journal of Digital Imaging* vol. 33 4 (2020).
5. World Health Organization. *Global Tuberculosis Report 2023.*
6. Ayas, S.; Ekinci, M. "Random forest-based tuberculosis bacteria classification in images of ZN-stained sputum smear samples." *Signal Image Video Process* (2014).
7. Khutlang, R.; Krishnan, S.; Dendere, R.; Whitelaw, A.; Veropoulos, K.; Learmonth, G.; Douglas, T.S. "Classification of Mycobacterium tuberculosis in images of ZN-stained sputum smears." *IEEE Trans. Inf. Technol Biomed* (2010).
8. Yang, Mu et al. "A CNN-based active learning framework to identify mycobacteria in digitized Ziehl-Neelsen stained human tissues." *Computerized medical imaging and*

graphics : the official journal of the Computerized Medical Imaging Society vol. 84 (2020).

9. Zaizen, Yoshiaki et al. "Deep-Learning-Aided Detection of Mycobacteria in Pathology Specimens Increases the Sensitivity in Early Diagnosis of Pulmonary Tuberculosis Compared with Bacteriology Tests." *Diagnostics* 12, no. 3 (2022).
10. L. Wang, X. Zhang, H. Su and J. Zhu, "A Comprehensive Survey of Continual Learning: Theory, Method and Application." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).
11. Mai, Zheda, et al. "Online continual learning in image classification: An empirical survey." *Neurocomputing* 469 (2022).
12. Goode, Adam et al. "OpenSlide: A vendor-neutral software foundation for digital pathology." *Journal of Pathology Informatics* vol. 4 27 (2013).
13. OpenSeadragon. "OpenSeadragon." https://github.com/openseadragon/openseadragon
14. Hadsell, Raia et al. "Embracing Change: Continual Learning in Deep Neural Networks." *Trends in Cognitive Sciences* vol. 24,12 (2020).
15. McCloskey, Michael and Neal J. Cohen. "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem." *Psychology of Learning and Motivation* 24 (1989).
16. Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.12 (2017).

17. G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network.” *NIPS Workshop* (2014).

18. S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C. H. Lampert, “iCaRL: Incremental classifier and representation learning.” *IEEE conference on Computer Vision and Pattern Recognition* (2017).

19. Rolnick, David, et al. "Experience replay for continual learning." *Advances in Neural Information Processing Systems* 32 (2019).

20. Vitter, Jeffrey Scott. “Random sampling with a reservoir.” *ACM Transactions on Mathematical Software* 11 (1985).

21. R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, L. Page-Caccia, “Online continual learning with maximal interfered retrieval.” *Advances in Neural Information Processing Systems* 32 (2019).

22. A. Prabhu, P. H. Torr, P. K. Dokania, “GDumb: A Simple Approach that Questions Our Progress in Continual Learning.” *European Conference on Computer Vision*, Springer, (2020).

23. He, Kaiming et al. “Deep Residual Learning for Image Recognition.” *IEEE Conference on Computer Vision and Pattern Recognition* (2016).

24. Krizhevsky, Alex. “Learning Multiple Layers of Features from Tiny Images.” (2009).

Curriculum Vitae

Shizhao Wang

shizhaowang@ymail.com

Education:

B. S. Computer Science, University of Nevada, Las Vegas.

Minor in Mathematical Sciences.

Teaching Experience:

Instructor, University of Nevada, Las Vegas.

Courses: Computer Science I, Computer Science II.

Graduate Assistant, University of Nevada, Las Vegas.

Courses: Analysis of Algorithms, Software Product Development & Design I.