MEDMNIST CLASSIFICATION: DAM GENERALIZATION IMPROVEMENT ON SMALL DATASET FOR MEDICAL IMAGE ANALYSIS

Priyanka Reddy Chukka (633003352),

Quoc Vu (827007671)

Introduction:

Deep learning techniques for medical image classification have been vastly improved in the last decade. Deep AUC maximization (DAM) is a robust machine learning algorithm that aims to optimize the area under the receiver operating characteristic (ROC) curve (AUC: a common metric for binary classification tasks) using deep learning architectures. Even though DAM has achieved great success for large datasets, it tends to overfit on small datasets in terms of AUC score. The goal of this project is to improve the generalization of the DAM paradigm over medical image classification tasks.

We will test our DAM generalization ability over seven different MedMNIST datasets which includes the 2D data set: BreastMNIST, PneumoniaMNIST, ChestMNIST and 3D data set: NoduleMNIST3D, AdrenalMNIST3D, VesselMNIST3D, SynapseMNIST3D. These datasets are used to train machine learning models for disease detection and classification.

We aim to improve the benchmark performance reported in the MedMNIST paper using the same network structure. In this project, we choose to use ResNet-18 and ResNet-18 + 3D architectures to train machine learning models using these datasets. ResNet-18 is a convolutional neural network (CNN) architecture that has been used extensively for image classification tasks. ResNet-18 can be used for deep AUC maximization by fine-tuning the network on a binary classification task and optimizing the AUC using a suitable loss function. ResNet-18 + 3D is an extension of ResNet-18 that can handle 3D images.

Motivation for the study:

Medical image analysis is an important field that plays a critical role in the diagnosis and treatment of various diseases. Machine learning algorithms have shown great promise in this field, but they often require large amounts of data to achieve high accuracy. However, obtaining large datasets in medical imaging can be challenging due to privacy concerns and the cost of acquiring and annotating images. Therefore, there is a need for algorithms that can achieve high accuracy even with small datasets. In this study, we aim to improve the generalization of the DAM algorithm on small medical image datasets, which can have significant implications for disease diagnosis and treatment.

Dataset Description:

The MedMNIST dataset is a collection of medical image datasets in the form of 28x28 grayscale images, with each dataset corresponding to a specific task in medical imaging. Here, we choose to use seven pre-processed datasets which consist of three 2D datasets and four 3D datasets.

1. BreastMNIST: This dataset contains mammogram images that are classified into two categories: malignant and benign.

2. PneumoniaMNIST: This dataset contains chest X-ray images that are classified into two categories: normal and pneumonia.

3. ChestMNIST: This dataset contains chest X-ray images that are classified into 14 categories, including normal, bacterial pneumonia, and COVID-19.

4. NoduleMNIST3D: This dataset contains 3D CT scan images of lung nodules that are classified into two categories: malignant and benign.

5. AdrenalMNIST3D: This dataset contains 3D CT scan images of adrenal glands that are classified into two categories: normal and abnormal.

6. VesselMNIST3D: This dataset contains 3D MRI images of blood vessels that are classified into two categories: normal and abnormal.

7. SynapseMNIST3D: This dataset contains 3D MRI images of synapses that are classified into two categories: normal and abnormal.

We first converted the images and labels into arrays, and then processed them to create datasets. Next, we normalized the datasets by subtracting the mean and dividing by the standard deviation, both of which were set to 0.5. Finally, we fed the normalized datasets into the model for training or inference.

Network Structure:

ResNet-18 is a CNN architecture that is characterized by its use of residual blocks, which enable the network to learn residual mappings instead of directly learning the underlying mappings. This allows the network to learn deeper representations with fewer parameters and better generalization performance. It has been used extensively for image classification tasks, and can be used for deep AUC maximization by fine-tuning the network on a binary classification task and optimizing the AUC using a suitable loss function.

The ResNet-18 architecture used in this report contains 18 layers, including a convolutional layer, a max pooling layer, and 16 residual blocks. The residual blocks in ResNet-18 have two 3x3 convolutional layers, followed by batch normalization and a ReLU activation function, with a skip connection that bypasses the convolutional layers.. The input images are resized to 224 x 224 pixels.

ResNet-18 + 3D Architecture: ResNet-18 + 3D is an extension of ResNet-18 that can handle 3D images. It consists of two parts: a 2D CNN that processes slices of the 3D image, and a 3D CNN that processes the entire 3D image. The 2D CNN is used to extract features from each slice of the 3D image, and the 3D CNN is used to combine the features from all the slices to classify the entire 3D images.

The ResNet-18 + 3D architecture used in this report consists of a 2D CNN and a 3D CNN. The 2D CNN is the same as the ResNet-18 architecture, but is applied to each slice of the 3D image. The 3D CNN consists of three residual blocks that process the features extracted from the 2D CNN.

The ResNet-18 + 2D and ResNet-18 + 3D architecture is now fine-tuned for deep AUC maximization by optimizing the AUC using a suitable loss function on a binary classification task.

Method:

In this project, we used the ResNet-18 structure for 2D and 3D to train these MedMNIST dataset. We used the LibAUC library (https://libauc.org), which has implemented a series of algorithms for optimizing AUROC, AUPRC, partial AUC, ranking measures, and other contrastive losses. We combined the PESG, SOAP and SOPA optimizer with the AUCMLoss, AUPRCLoss and PartialAUCLoss functions to further optimize the AUC.

The PESG optimizer is an optimizer that is designed to efficiently search for global optima in high-dimensional optimization problems. It is a population-based optimizer, which means that it maintains a population of candidate solutions and updates them over time based on their fitness. The AUCMLoss function is based on the observation that the AUC metric can be expressed as the probability that a positive example is ranked higher than a negative example by the classifier. The AUCMLoss function uses this observation to compute a surrogate loss function that directly maximizes the probability of ranking positive examples higher than negative examples.

SOAP (Second-Order Approximation-based optimizer for deep neural networks) is an optimizer that is designed to improve the training of deep neural networks. The optimizer is based on a second-order approximation of the Hessian matrix, which allows it to more accurately estimate the curvature of the loss surface and optimize the network parameters accordingly. The AUPRC loss is a loss function that is used to optimize binary classification models for imbalanced datasets. It is calculated based on the precision and recall of the model predictions, and it places more emphasis on correctly identifying positive samples (i.e., those belonging to the minority class) than on correctly identifying negative samples.

We have tried several methods to improve the generalization of the algorithms and the performance bench written in the MedMNIST report:

Data augmentation: a technique used in machine learning and deep learning to artificially increase the size of a training dataset by creating additional, modified versions of the original data (horizontal flip, vertical flip, rotation of the original images). We thought that we can improve the robustness and generalization ability of the model by exposing it to a larger and more diverse set of training examples. However, at the end, the technique only improves the performance on 3D dataset but not 2D dataset. We suspect this was because the 3D dataset got more complex structures and variations in anatomy in comparison to the 2D dataset. That is why data augmentation on 3D datasets works better than that of 2D.

Control overfitting: There are several approaches that we tried to control the overfitting:

- Dropout: Set some of the neurons in a layer to zero during training.We have tried this method using dropout probability of 0.2, 0.3, 0.5 but it did not work on all datasets. There are two reasons that we think this did not work on some of the datasets. The first one is

that it could be because the number of epochs is a bit large so further regularization can not prevent overfitting. The second reason is because this model is quite complex so dropout is not efficient as expected.

- Weight and epoch decay: Weight decay works by adding a penalty term to the loss function to encourage the weights of the model to be small. Epoch decay, also known as learning rate scheduling, works by decreasing the learning rate of the optimizer over time. Both methods worked for all of the datasets. Weight decay works well in this situation because the weights in the model are too large and make the model easily overfit on training data. Epoch decay also performs effectively because we chose the right schedule that helps with this specific problem.
- Tuning learning rate: The learning rate is a hyperparameter that controls the size of the updates made to the model's weights during training. The slower we tuned the learning rate, the better the model generalized. We started with a high learning rate and decreased it at 50 and 75 epochs. By starting with a higher learning rate at the beginning of training, the model can make larger updates to the weights, which can speed up the convergence to the optimal solution.
- Early stopping: Monitor the performance of the model on a validation set during training and stop the training process when the performance of the model on the validation set starts to degrade. We think that this technique did not work because we are working with small datasets and there is more variance in the validation sets.

Distributional shift between training/validation and testing datasets: All of the datasets were imbalanced so we decided to use DualSampler with Imratio to handle this. DualSampler works by oversampling the positive examples (i.e., minority class) and undersampling the negative examples (i.e., majority class) simultaneously. Imratio is a hyperparameter that controls the degree of oversampling and undersampling applied to the minority and majority classes. We tried this method to balance out all samples and avoid the model to only generalize on positive samples.

Optimizer: We have tried with different optimizers such as: SGD, Adagrad, Adam, Rmsprop but PESG performs the best for all datasets with AUCMLoss, SOAP with AUPRCLoss and SOPA with PartialAUCLoss.

Batch size: We have tried training with different batch sizes and realized that the lower the batch size, the better the model performs. We suspect this is because a smaller batch size will raise more noise in the model and helps reduce overfitting. Also, with smaller batch size, the model makes more updates to its parameters per epoch, since it processes more batches of data.

Data centric: The validation and training datasets represent the test well even though both of them were overfitting on the validation so we did not try this approach.

Different Loss Functions:

AUPRC Loss with PESG Optimizer: The AUPRC loss is particularly useful when the positive class is rare, it helps to ensure that the model is able to correctly identify these samples even when they are heavily outnumbered by the negative samples, and with datasets being heavily imbalanced, using this to train the network along with data augmentation, weight and epoch decay helped in improving the model performance both for 3D and 2D datasets. After tuning the entire model using above hyperparameters along with this loss function, we were still facing some difficulty in getting the model to perform above baseline for 3D, so we tried other two loss functions.

AUPRC Loss with SOAP Optimizer: We also tried to train our model with partial AUC Loss and Soap Optimizer and realized that our performance increases with these two methods combined. We are not quite sure how this works but we assume that the data augmentation for 3D data helps creating larger and more complex datasets and Soap optimizer is memory-efficient which combined with partial AUC loss helps focusing on most imbalanced areas especially for 3D.

Partial AUC Loss with SOPA Optimizer: Unlike the AUC which measures the overall performance of a classifier across all false positive rates (FPR), the partial AUC focuses on a region of the receiver operating characteristic (ROC) curve, typically a range of low FPR values that are relevant to a particular application or domain. We were hoping to use this to help with tuning parameters and reduce overfitting in the regions that are more important to the model. However, tuning with the partial AUC did not work as well as that of AUC. This could be because the partial AUC is more sensitive to class imbalance than AUC and the datasets were all imbalanced.

To train and evaluate our models, we split the available data into three sets: a training set, a validation set, and a test set. The training set was used to train the model, while the validation set was used to evaluate its performance during training. We used 80% of the available data for training and the remaining 20% for validation. We reported the Mean AUC score achieved during training, and to ensure that our results were generalizable, we evaluated our models on a separate test set.

In addition to the hyperparameters that we tuned (batch size, weight decay, epoch decay, imratio, and starting learning rate), we also explored data augmentation techniques and different loss functions to see which would help our model generalize better. Specifically, we used 3D data augmentation techniques to augment the training data, which included random rotations, translations, and scaling. We also tried different loss functions, such as binary cross-entropy and focal loss, to see which worked best for our specific problem.

In conclusion, we found that a batch size of 128, weight decay of 1e-03, epoch decay of 1e-01, imratio of 0.25, and starting learning rate of 0.1 with a learning rate schedule that decreased by a factor of 100 in epochs 50 and 75 worked well for all the datasets we experimented with. Furthermore, by using 3D data augmentation techniques and experimenting with different loss functions, we were able to improve the generalization performance of our models.

Finally, we compared our results to the benchmark performance reported in the MedMNIST paper using the same network structure.

Final Results:

Mean AUC performance on 2D Validation set with different methods using AUCMLoss:

Methods	*Breast*	*Pneumonia*	*Chest*
Without sampler & No Data Aug	*0.9198*	*0.9957*	*0.7176*
With sampler & No Data Aug	*0.9399*	*0.9959*	*0.5607*
Data aug, sampler, weight and epoch decay	*0.3484*	*0.3925*	*0.4897*
No Data Aug, sampler, weight and epoch decay	*0.9490*	*0.5045*	*0.7277*

Mean AUC performance on 3D Validation set with different methods using AUCMLoss:

	Nodule	*Adrenal*	*Vessel*	*Synapse*
Without sampler & No Data Aug	*0.8623*	*0.8331*	*0.8839*	*0.4629*
With sampler & No Data Aug	*0.8419*	*0.8032*	*0.7265*	*0.5924*
Data aug, sampler, weight and epoch decay	*0.5579*	*0.2835*	*0.4917*	*0.6951*

Mean AUC Performance on Validation set with different Loss Functions:

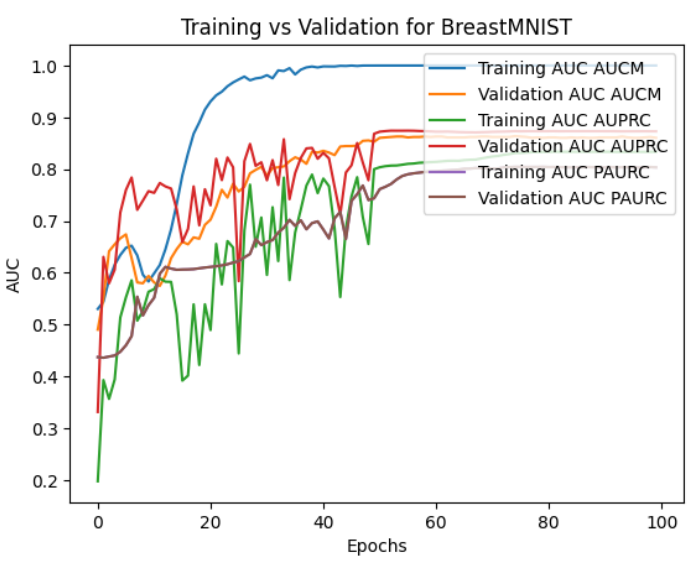
	Breast	*Pneumonia*	*Chest*	*Nodule*	*Adrenal*	*Vessel*	*Synapse*
AUCML Loss + PESG	*0.9312*	*0.9972*	*0.7277*	*0.5579*	*0.8223*	*0.8373*	*0.7285*
AUPRC Loss + SOAP	*0.8730*	*0.9935*	*0.768*	*0.8378*	*0.8386*	*0.9175*	*0.6471*
PAURC Loss + SOPA	*0.8034*	*0.9953*	*0.768*	*0.8171*	*0.7540*	*0.6975*	*0.5678*

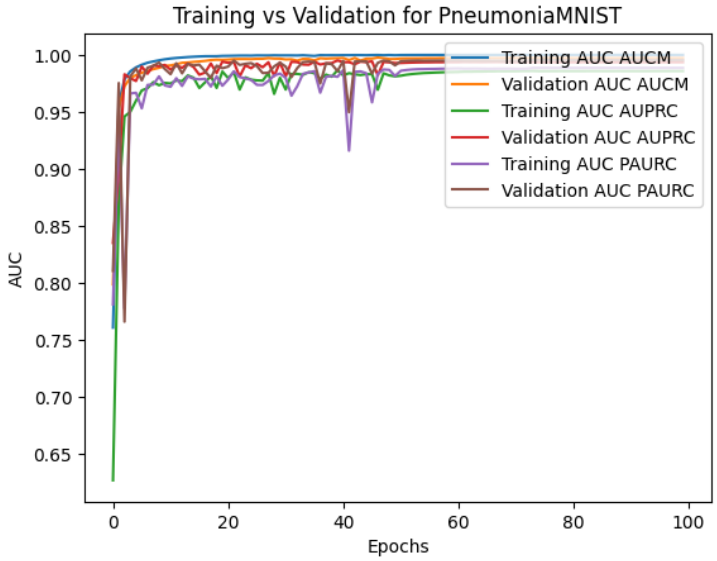
Mean AUC Final Performance on Test set:

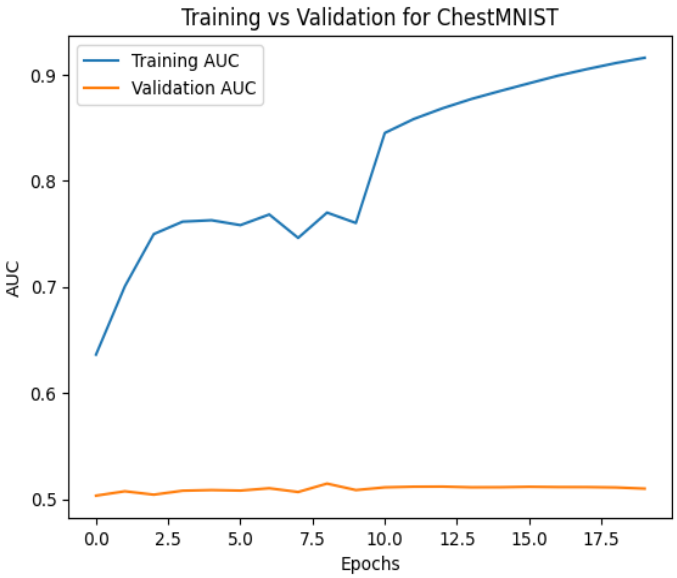
	Breast	***Pneumonia***	***Chest***	***Nodule***	***Adrenal***	***Vessel***	***Synapse***
Final AUC score:	***0.9313***	***0.9625***	***0.7422***	***0.9482***	***0.9405***	***0.9752***	***0.7929***
AUC score in MedMnist paper	***0.901***	***0.944***	***0.768***	***0.863***	***0.827***	***0.874***	***0.82***

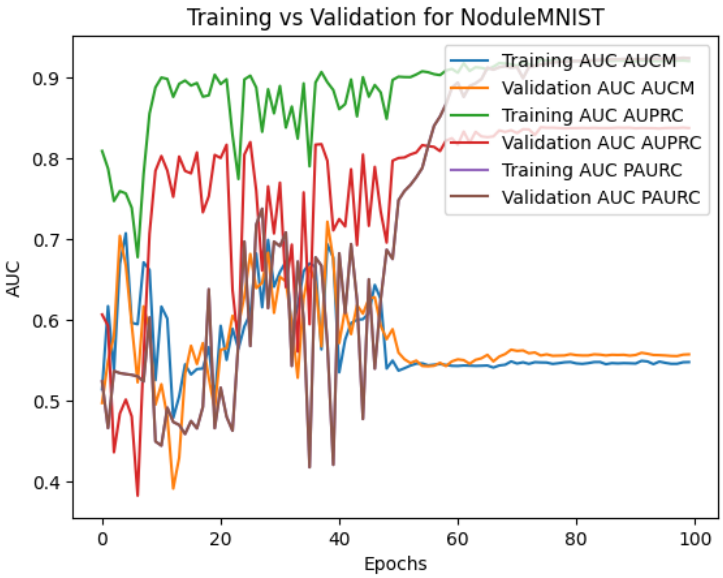
Training vs Validation Graphs for different Loss functions:

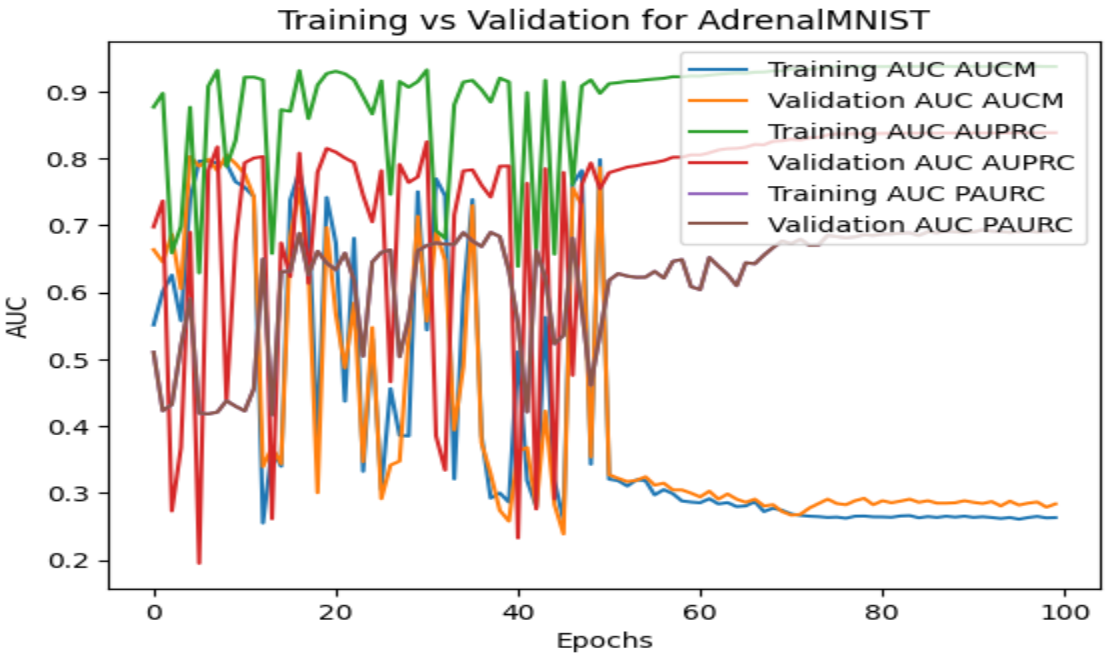
As depicted in the graphs below, the training AUC experiences a swift increase during the initial epochs but eventually plateaus as the model begins to overfit the training data. Conversely, the validation AUC decreases for a few more epochs before also reaching a plateau. These trends suggest that the model is learning to generalize effectively to the validation data. The performance of the model with various loss functions can be observed. Notably, the AUCMLoss performs well with 2D datasets, while the AUPRCLoss performs well with 3D datasets after hyperparameter tuning.

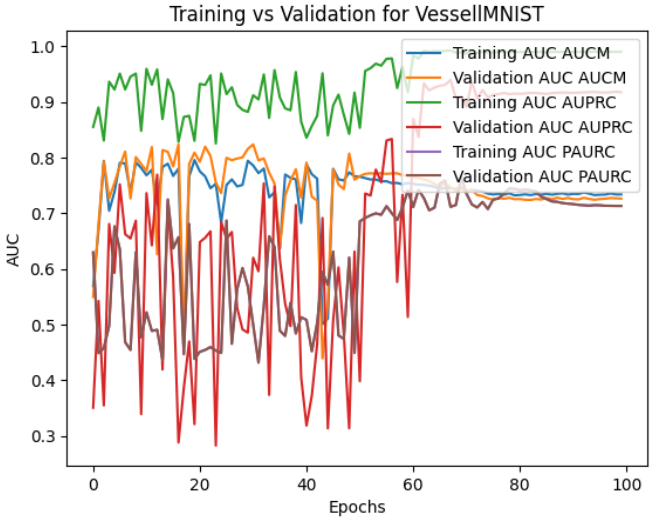


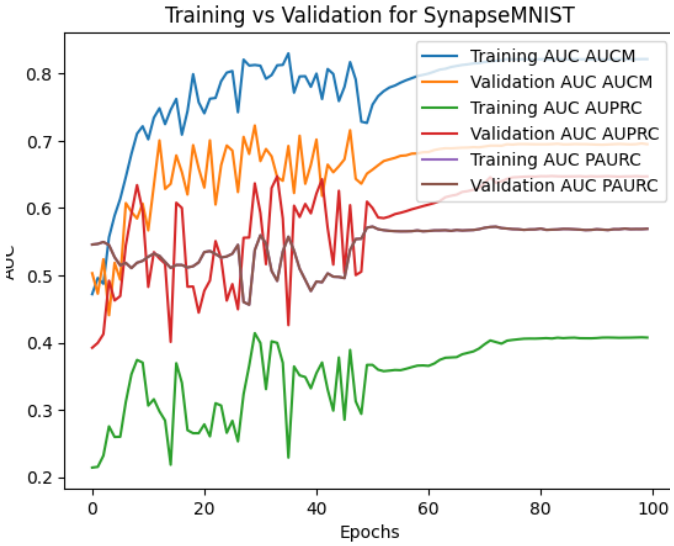







Training vs Validation for AdrenalMNIST
Training AUC AUCM
Validation AUC AUCM
Training AUC AUPRC
Validation AUC AUPRC
Training AUC PAURC
Validation AUC PAURC
AUC
Epochs


Training vs Validation for VesselMNIST
Training AUC AUCM
Validation AUC AUCM
Training AUC AUPRC
Validation AUC AUPRC
Training AUC PAURC
Validation AUC PAURC
AUC
Epochs


Training vs Validation for SynapseMNIST
Training AUC AUCM
Validation AUC AUCM
Training AUC AUPRC
Validation AUC AUPRC
Training AUC PAURC
Validation AUC PAURC
Epochs

Conclusion and thoughts:

In conclusion, this project aimed to improve the generalization performance of the Deep AUC Maximization (DAM) algorithm on small medical image datasets using the MedMNIST dataset. The outcomes demonstrate that with hyperparameter tuning, controlling overfitting, the ResNet-18 architecture outperforms the ResNet-18 baseline on 5 out of 7 datasets. This indicates that ResNet-18 is a suitable architecture that is effective in dealing with various types of medical images and achieving high AUC in disease detection and classification.

Additionally, we discovered that fine-tuning the hyperparameters, along with implementing data augmentation techniques such as random cropping and horizontal flipping, boosts the performance of the models. This finding suggests that data augmentation tuned with right parameters and correct loss function can be a valuable technique for enhancing the generalization of deep learning models on small 3D datasets.

Moreover, we employed the LibAUC library to optimize the AUC metric and discovered that it significantly improved the AUC scores of our models. Particularly, the AUCMLoss performed well with 2D datasets, while the AUPRCLoss worked well with 3D datasets.

Overall, our findings suggest that combining the ResNet-18 architecture with data augmentation and the LibAUC library can be a promising approach for improving the generalization of the DAM algorithm on small medical image datasets. Future research could investigate the effectiveness of our method on other datasets and explore the utilization of other deep learning architectures for enhancing the generalization of DAM, including ensembling and transfer learning.