Medical AUC Maximization

Vicente Balmaseda Lance Fletcher

May 2023

1 Introduction

AUC (Area Under the Curve) is a common metric used when evaluating models used for classification problems. It measures the performance of a classification model by calculating the area under the receiver operating characteristic (ROC) curve. Recently, there has been development in optimizing models by maximizing their AUC score on datasets [4]. These Deep AUC Maximization (DAM) models have proven to outperform many other approaches, especially in medical imagery [6].

DAM has successfully been applied to many problems; however, it is limited when presented with a small dataset. When evaluating the AUC metrics of models trained on small datsets, cross entropy (CE) optimization often has better results. As stated in the project description, this is because a DAM model likely overfits the training data due to it typically having a higher AUC on training data, but a lower AUC on testing data. This paper presents attempts to combat this issue ranging from hyperparameter tuning to ensemble methods. As instructed and shown in Table 1, 7 of the 14 datasets contained in MedMNIST v2 [3] were used to evaluate our methods. To see the results for our submission for the incentive portion of this project, see Section 5.

Table 1: Datasets used.

Name	**# Samples**	**Train**	**Validate**	**Test**
BreastMNIST	780	546	78	156
PneumoniaMNIST	5,856	4,708	524	624
ChestMNIST	112,120	78,468	11,219	22,433
NoduleMNIST3D	1,633	1,158	165	310
AdrenalMNIST3D	1,584	1,188	98	298
VesselMNIST3D	1,909	1,335	192	382
SynapseMNIST3D	1,759	1,230	177	352

2 Experiments

We have separated our work between the incentive program and some additional experiments. For the incentive program, we have focused on achieving higher AUC scores, while for the additional experiments, our objective is to study the impact of different techniques to overcome overfitting for AUC maximization.

For MedMNIST2D we use ResNet18 with 3 input channels and a batch size of 128. We use the image size that performed best for the datasets as shown in the MedMNIST v2 paper [3].

That is, for BreastMNIST we use image size 28, and for PneumoniaMNIST 128. Throughout this work, we treat ChestMNIST separately due to resource constraints and to speed up training. For ChestMNIST, we used image size 28, mixed-16 precision, and early stopping (50 epochs) for ChestMNIST.

For MedMNIST3D we use ResNet18 3D [2] with 3 input channels and batch size 32. We resize the images to 80 (28x80x80) and use early stopping (50 epochs) and mixed-16 precision to speed up training and reduce memory usage.

These parameters are used in most experiments. Additionally, we use AUCMLoss [6] with PESG optimizer as implemented in LibAUC [5]. We also normalize the images with a mean and standard deviation of 0.5. We train for 200 epochs, reducing the learning rate by a factor of 0.1 after 100 and 150 epochs. We compute the validation AUC every 5 epochs, which is used for those scenarios that use early stopping or keep track of the best validation AUC score. To ensure reproducible results to the extent possible, we have seeded training and used PyTorch 2.0 and PyTorch Lightning 2.0.1.

2.0.1 Pre-Training

For AdrenalMNIST3D we first run pretraining (see Section 2.7) and then optimize AUC on the pretrained model. This results on the model presented for AdrenalMNIST3D.

2.1 Cross Entropy Baseline

We first trained ResNet 18 (MedMNIST2D) and ResNet18 3D (MedMNIST3D) with Binary Cross Entropy to obtain the baseline model. For the hyperparameters, we follow the instructions from the MedMNIST paper [3]. That is, we use Binary Cross Entropy loss for 100 epochs with an initial learning rate of 1e-3 and AdamW optimizer. After 50 and 75 epochs we reduce the learning rate by 0.1. We use an image resolution of 28 with 3 input channels and a batch size of 128. We compute the validation AUC every 5 epochs, selecting the epoch that achieved the highest validation AUC.

2.2 Optimizer and AUCMLoss Tuning

We want to study whether tuning the optimizer and AUCMLoss helps with overfitting and, thus, improves the test AUC when using AUCMLoss. Specifically, we have run a grid search using different values for weight decay, epoch decay, learning rate, and margin. For each combination of hyperparameters, we evaluate the validation AUC every 5 epochs and select the epoch with the highest validation score. For each dataset, we select the combination of hyperparameters that achieves the highest validation AUC.

For MedMNIST2D, we test combinations for the following hyperparameters: learning rate (1e-1), weight decay (1e-3, 1e-4, 1e-5), epoch decay (3e-2), and margin (1.0, 0.8, 0.6).

For MedMNIST2D, we use early stopping (patience 50 epochs) and test combinations for the following hyperparameters: learning rate (1e-1, 1e-2), weight decay (1e-4, 1e-5), epoch decay (3e-2, 3e-3), and margin (1.0, 0.6).

For ChestMNIST, we use early stopping (patience 50 epochs) and mixed-16 precision for faster training. We test combinations for the following hyperparameters: learning rate (1e-2), weight decay (1e-4, 1e-5), epoch decay (3e-2), and margin (1.0, 0.6).

The best models obtained in this section will be used for later experiments to compare whether the following techniques improve generalization. Moreover, running the grid search on the 7 datasets is expensive, so this helps keep computational costs affordable.

2.3 Dropout

Dropout [1] is a regularization technique that can be used to prevent overfitting. The idea behind dropout is to randomly set to zero a certain percentage of neurons during training. Doing so, dropout prevents relying excessively on any single set of neurons, forcing them to learn more robust and general features rather than memorizing the training data, which can help improve generalization to unseen data. This technique is widely used to reduce overfitting and has shown considerable results.

We have added dropout layers with a dropout rate of 0.1 (10%) after every ReLU layer in ResNet18 to test whether dropout helps AUCMLoss overcome overfitting. We have rerun the grid search from Section 2.2 and selected the highest validation AUC for each dataset. However, for a certain combination, we do not choose the epoch with the highest validation AUC but instead the last epoch. The hypothesis is that additional epochs might help the model become more stable and robust, even though it does not improve validation AUC.

2.4 Resize

We tested the effect of resizing on all datasets. Although MedMNIST [3] shows results for ResNet18 using image resolutions of 28 and 224 for MedMNIST2D, it does not study the effect of resizing on MedMNIST3D. Moreover, those experiments were run using Binary Cross Entropy. Our objective is to know whether resizing improves performance on MedMNIST3D and whether the results obtained on MedMNIST2D are similar for AUCMLoss and BCELoss.

Increasing the image resolution increases the computational cost, slowing down training and requiring more resources. However, it also increases the size of the feature maps, which can capture more context and spatial information, thus improving performance.

We use the optimal hyperparameters obtained in Section 2.2 and compare image resolutions of 28 and 128 for MedMNIST2D, and 28 and 80 for MedMNIST3D (we only resize two dimensions, i.e., 28x80x80). We used mixed-16 precision for MedMNIST3D with an image resolution of 80 due to memory constraints.

2.5 Augmentations

DAM can overfit to small training data in terms of AUC score. One way to alleviate this is by adding more data. However, this can be costly or even unfeasible (such as in this case). Instead, we can augment our dataset by including transformations of the training images into our training set. Doing so results in an increase in the training dataset size, thus alleviating the lack of data. Moreover, augmentations can improve generalization by making the model invariant to a variety of transformations that do not affect the task at hand. In other words, it makes the model robust to noise and non-important variability, thus reducing overfitting and making the model less biased toward training data.

We have tested a set of augmentations on each dataset to observe what transformations provide a higher improvement to the generalization of the models. We expect different datasets to perform better with different augmentations since what augmentations to use is highly dependent on the task at hand. Since different datasets have different kinds of data (chest x-rays, breast ultrasound, chest CT, brain MRA, etc.), each data may be affected in different ways by the same transformation. Moreover, some transformations may remove important information from the image. Therefore, augmentations are one of the pre-processing techniques where field knowledge can be very useful to improve the models.

We have tested 2D augmentations for both MedMNIST2D and MedMNIST3D. Although 3D images can have their own augmentations, since the additional dimension provides more ways of augmenting the data, we have not explored this area due to time constraints. For MedMNIST3D, the augmentations are applied to the height and width dimensions, leaving the depth dimension invariant. Each model is trained using only one augmentation. We use the implementations from torchvision. After training, we use the last epoch to compute the validation AUC, and choose the augmentation with the highest score for each dataset.

For MedMNIST2D, we have tested the following augmentations:

- Random rotations in the range $[-5, 5]$ degrees
- Gaussian blur with sigma in the range $[0.1, 2]$
- Random resized crops with a scale chosen randomly in the range $[0.7, 1]$

For MedMNIST3D, we use early stopping to speed up training. We have tested the following augmentations:

- Random rotations in the range $[-5, 5]$ degrees
- Elastic transform with alpha 0.1 and sigma 0.1
- Random resized crops with a scale chosen randomly in the range $[0.7, 1]$
- Regularization from MedMNIST paper [3], i.e., multiply the training set by a random value in the range $[0, 1]$ during training and by a fixed coefficient of 0.5 during evaluation.

Since the ChestMNIST dataset has proven to be especially difficult, we have tested additional augmentations. Moreover, we observed that, during training, the loss first decreases and then starts increasing again while the validation AUC does not follow the same trend. Moreover, the validation AUC does not become stable over time. Thus, we use early stopping with a patience of 50 epochs and select the epoch with the highest validation AUC. We have tested the following augmentations:

- Random rotations in the range $[-5, 5]$ degrees
- Gaussian blur with sigma in the range $[0.1, 2]$
- Random resized crops with a scale chosen randomly in the range $[0.7, 1]$
- Elastic transform with alpha 0.1 and sigma 0.1
- Color jitter with brightness and contrast to 0.1 (grayscale images)

2.6 Training on Train and Validation Datasets

As discussed earlier, the lack of data for many machine learning problems can hamper a model's ability to correctly classify new data. Depending on the application, collecting new data is often not an option. Specifically, medical imagery is expensive and can be difficult to curate for many ethical, economic, and legal reasons.

Therefore, we took the approach of using both the train and validation datasets to train a model. For each dataset, we utilized the optimal hyperparameters found by the grid search described in Section 2.2. Since we used the validation set in the training, it does not make sense to choose the model which performs best on the validation set; therefore, we trained the models for 200 epochs and used the final "stabilized" model. Due to time constraints, we were only able to apply this method to the BreastMNIST and PneumoniaMNIST datasets.

2.7 Pre-Training Cosine Similarity

Instead of directly optimizing the AUC loss, we first try to find good representations for the training data. Since using a large external dataset is out of scope, we focus on using the available data. The objective is to obtain representations that will make it easier to learn the task at hand. This approach is focused on trying to improve the performance of datasets in which directly optimizing AUC does not perform as well.

We first train a ResNet18 without the classification head to maximize the cosine similarity between samples with the same label and minimize the similarity between samples with different labels. For each sample i, ResNet18 computes a hidden vector h_i of size 512 which is the input of the classification head. If A is the matrix where row i is h_i after normalizing it, such that $A_{i:}A_{j:}$ is the cosine similarity between samples i and j, and y is the true label of each sample, then the objective is to minimize the following loss:

$$\sum_{i,j}[(AA^T - Y) * (1 - I)]_{ij}$$

where $Y_{ij} = y_i == y_j$ (1 if i and j have the same label, 0 otherwise). After pertaining, we add back the classification head and optimize the AUCMLoss as we have been doing.

Due to time constraints, we have only tested this approach on AdrenalMNIST3D.

2.8 Ensemble Methods

Ensemble methods in machine learning involve combining multiple models to improve predictive performance. The basic idea is to use several weaker models that individually may not perform well, but when combined, they can produce a stronger model that is better at predicting outcomes. There are a variety of ensemble methods such as bagging, boosting, and stacking. We utilized bagging due to ease of implementation and its lesser computational cost compared to other ensemble methods. All the ensemble approaches utilized three basic AUC optimization models where each model was trained on a unique subset of the training data. We trained three different bagging-based ensemble models using methods specifically related to the training datasets. Below are the three different approaches taken.

- Basic Split: Each of the three models in the ensemble model is trained on an equal share of the original training data.
- Oversampling: Same as Basic Split, but each model then oversamples the minority class to account for the unbalanced datasets.
- Train+Val: Combine both the train and validation datasets and then equally split the data as done in Basic Split.

2.9 Additional Experiments

Initially, we tried using another loss function in the LibAUC library, specifically CompositionalAUCLoss. The compositional loss combines AUC and CE to create a metric which can account for both. We performed a hyperparameter grid search which tested many different combinations of learning rates and beta values; however, this was unsuccessful. There were other loss functions that could have been evaluated, but we decided to focus on mitigating the overfitting by attempting more data-centric techniques.

Additionally, we compared different ways to handle the 3-Dimensional data 4 of the 7 datasets contained. One way to handle the additional spatial data is to treat it as another channel of an image and apply 2D convolutions. A more common technique is to apply a 3D convolution over the data. The 3D convolution quickly proved to superior to the channels method and all the models described in this paper utilized 3D convolutions.

3 Results

Tables 2, 7, and 4 present the results for the different experiments.

Table 2: MedMNIST2D: Test AUC for the best model for each experiment and dataset

Experiment	BreastMNIST	PneumoniaMNIST	ChestMNIST
Cross Entropy (28)	0.901	0.961	**0.747**
AUCMLoss (28)	0.904	0.966	0.706
Dropout (28)	0.882	0.957	0.645
Comp	0.868	0.943	0.612
Resize (128/80)	0.875	0.973	0.689
Augmented	0.916	0.983	0.716
Train+Val	**0.923**	**0.987**	-

Table 3: MedMNIST3D: Test AUC for the best model for each experiment and dataset

Experiment	Nodule3D	Adrenal3D	Vessel3D	Synapse3D
Cross Entropy (28)	0.904	**0.88**	**0.949**	0.756
AUCMLoss (28)	0.892	0.863	0.889	0.73
Dropout (28)	0.891	0.822	0.931	0.733
Comp	0.869	0.857	0.919	0.728
Resize (128/80)	0.889	0.875	0.93	**0.802**
Augmented	**0.925**	-	0.937	0.79
Pretrained	-	0.869	-	-

Table 4: Mean test AUC each experiment

Name	Mean
Cross Entropy (28)	0.871
AUCMLoss (28)	0.850
Dropout (28)	0.837
Comp	0.828
Resize (128/80)	0.862
Augmented	**0.878**

3.1 Baseline Cross Entropy vs AUCLoss

Just by tuning the hyperparameters for the PESG optimizer and the AUCMLoss, we are able to beat BCE on BreastMNIST and PneumoniaMNIST. However, the mean test AUC over the 7 datasets is lower than for BCE. However, the mean validation AUC is the same for AUCMLoss and BCE. This confirms the issue with overfitting that DAM is facing. Thus, the AUCMLoss and PESG optimizers by themselves do not achieve as good AUC performance as BCE and require additional techniques to overcome overfitting.

3.2 Dropout

Dropout performs, in general, worse than AUCMLoss by itself, only outperforming on some specific datasets (SynapseMNIST3D and VesselMNIST3D). Although regularization can sometimes help, it can also hurt performance due to obtaining a suboptimal solution. In our scenario, dropout is not a good option. Some causes for this might be not enough tuning of the dropout rate, not enough training time, or that ResNet18 does not have enough neurons for dropout to help, i.e., regularization is not helping since the test score for the optimal of the original problem is already the maximum.

3.3 Resize

The results do not provide any conclusive information. While a higher resolution does help for PneumoniaMNIST, same as it did with BCE, it does not help for BreastMNIST and ChestMNIST. For ChestMNIST, BCE did improve with a higher resolution. For MedMNIST3D, a higher resolution achieves a higher score in general. One of the possible reasons for this difference between 2D and 3D images might be in the model being used, i.e., ResNet18 3D is able to take advantage of the increased information while ResNet18 is not. However, further testing would be required to confirm this statement.

Overall, resizing did improve testing AUC, with higher resolution images achieving higher AUC. This is due to the additional information available in the larger feature maps due to the increase in resolution. However, higher resolution models only outperform BCE in PneumoniaMNIST and SynapseMNIST3D, achieving a lower mean test AUC.

3.4 Augmentations

Augmentations achieve the highest results of all techniques tested. Using a single augmentation achieves a mean test AUC of 0.878, outperforming BCE (0.871). Moreover, augmented DAM is able to beat BCE on 4 out of 7 datasets. Different augmentations perform differently depending on the dataset. Table 5 shows the augmentations that performed best on each dataset. Random cropping seems to perform well for most datasets. This might be because it is not changing the image, but just providing different crops, thus not significantly modifying the information in the image (does not change color, orientation, etc.). This suggests that random scaling can also perform well. We hypothesize that further improvements can be obtained through a combination of augmentations and including some others, such as random scaling, combined with higher training times. Moreover, field knowledge can be added when choosing augmentations by adding transformations that should not affect the task.

Table 5: Best augmentation for each dataset

Dataset	Best Augmentation
BreastMNIST	Random cropping
PneumoniaMNIST	Random cropping
ChestMNIST	Color jitter
NoduleMNIST3D	Random cropping
VesselMNIST3D	Random cropping
SynapseMNIST3D	Elastic transformation

3.5 Training on Train and Validation Datasets

Due to time constraints, we only applied the *Train+Val* approach to the BreastMNIST and PneumoniaMNIST datasets. For these specific datasets, the technique proved to significantly help increase the AUC. As shown in Table 2, it achieved the highest score for both datasets. Given the best hyperparameters were used by each model, the additional training data likely helped the model learn a better decision boundary when compared to models just trained on the training set.

3.6 Pre-Training

Pretraining obtains slightly worst results than directly optimizing AUCMLoss. The reason might be that pretraining by itself can overfit even further or get stuck in a suboptimal solution. However, we hypothesize that pretraining could achieve better results when combined with augmentations and ensemble methods. Moreover, it can also be interesting to test the performance of pretraining across datasets to maximize the similarity between samples from the same dataset and minimize the similarity between different datasets.

3.7 Ensemble Methods

Table 6: MedMNIST2D: Test AUC for various ensemble methods.

Experiment	BreastMNIST	PneumoniaMNIST	ChestMNIST
Basic Split	0.79	0.957	-
Oversampling	**0.843**	0.941	-
Train+Val	0.828	**0.959**	-

Table 7: MedMNIST3D: Test AUC for various ensemble methods.

Experiment	Nodule3D	Adrenal3D	Vessel3D	Synapse3D
Basic Split	0.858	0.806	0.81	0.719
Oversampling	**0.899**	**0.815**	**0.839**	**0.727**
Train+Val	0.862	0.777	0.753	0.597

As seen in Tables 6 and 7 the ensemble models did not perform well in comparison to the other non-ensemble models. Ideally, combining "weaker" models would result in lower variance

in the predictions; however, since DAM already struggles with overfitting small datasets, training three models on even smaller datasets likely led to an overfit model. Of the techniques tried, oversampling performed the best of and shows the inbalance in the datasets likely contributes to the models performance. If time permitted, it would have been interesting to apply oversampling to a basic non-ensemble AUC optimization model.

Note: We did not apply any ensemble methods to the ChestMNIST dataset due to its large size and the inherent computational cost of ensemble models. Similarly, very little hyperparameter tuning was performed due to the large amount of time it takes to train the ensemble models.

4 Conclusions

Improving the AUC test score proved to be a challenging task. It can easily be seen that we attempted to improve the AUC models in a variety of ways. We were able to successfully surpass the AUC scores of the CE loss based model presented in the MedMNIST v2 paper [6]. Augmenting the training data had the largest average AUC amongst all datasets and proved to be the most effective method of improving the DAM-base models. Each augmentation was tested independently, but we believe a combination of the prove augmentations could show added model performance.

5 Incentive Program

We run a multi-step process to obtain the models submitted for the incentive program[1][2]. Table 8 presents the test AUC scores obtained.

Table 8: Incentive program: Test AUC scores

Dataset	**Test AUC**
BreastMNIST	0.923
PneumoniaMNIST	0.987
ChestMNIST	0.716
NoduleMNIST3D	0.925
AdrenalMNIST3D	0.869
VesselMNIST3D	0.937
SynapseMNIST3D	0.79
Mean	**0.878**

5.1 Hyperparameter Tuning

We first run a grid search to tune hyperparameters using images with size 28. For each combination of hyperparameters, we evaluate the validation AUC every 5 epochs and select the epoch with the highest validation score. For each dataset, we select the combination of hyperparameters that achieves the highest validation AUC.

For MedMNIST2D, we test combinations for the following hyperparameters: batch size (128), learning rate (1e-1), weight decay (1e-3, 1e-4, 1e-5), epoch decay (3e-2), and margin (1.0, 0.8, 0.6).

[1]The code and the best models are accessible at https://github.com/vibalcam/medical-max-auc

[2]The folder with the best models can be accessed at https://tamucs-my.sharepoint.com/:f:/g/personal/vibalcam_tamu_edu/Ekjod78yCmBAoN_ZQ0xSGxIBstQgvc8xgWBb7suabOhdaw?e=VL2yax

For MedMNIST3D, we use early stopping (patience 50 epochs) and test combinations for the following hyperparameters: batch size (32), learning rate (1e-1, 1e-2), weight decay (1e-4, 1e-5), epoch decay (3e-2, 3e-3), and margin (1.0, 0.6).

For ChestMNIST, we use early stopping (patience 50 epochs) and mixed-16 precision for faster training. We test combinations for the following hyperparameters: batch size (128), learning rate (1e-2), weight decay (1e-4, 1e-5), epoch decay (3e-2), and margin (1.0, 0.6).

5.2 Image Resolution

Some datasets will use image size 28 while others resize the image. Those datasets in which the models obtained in the MedMNIST v2 paper [3] with image size 28 outperformed size 224, will not use resize (thus, image size 28). This affects BreastMNIST. Otherwise, we use resize to 128x128 for 2D datasets (PneumoniaMNIST) and to 28x80x80 for 3D datasets (NoduleMNIST3D, AdrenalMNIST3D, VesselMNIST3DA, SynapseMNIST3D). The exception is ChestMNIST, which uses 28x28 due to resource constraints and to speed up training.

5.3 Augmentation Choice

A grid search to find the best augmentation to use for each dataset. For each augmentation and dataset, we train a model with the best hyperparameters obtained in previous steps for that dataset. We train for 200 epochs and choose the augmentation that provides the highest validation AUC.

For MedMNIST2D, we have tested the following augmentations:

- Random rotations in the range $[-5, 5]$ degrees
- Gaussian blur with sigma in the range $[0.1, 2]$
- Random resized crops with a scale chosen randomly in the range $[0.7, 1]$

For MedMNIST3D, we use early stopping and mixed-16 precision to speed up training. Additionally, for SynapseMNIST3D, we select the epoch with the highest validation AUC (validation AUC is not stable). We have tested the following augmentations:

- Random rotations in the range $[-5, 5]$ degrees
- Elastic transform with alpha 0.1 and sigma 0.1
- Random resized crops with a scale chosen randomly in the range $[0.7, 1]$
- Regularization from MedMNIST paper [3], i.e., multiply the training set by a random value in the range $[0, 1]$ during training and by a fixed coefficient of 0.5 during evaluation.

For **ChestMNIST**, we used early stopping (patience 50 epochs) and mixed-16 precision for faster training. Additionally, for each augmentation, we select the epoch with the highest validation AUC. We tested the following augmentations:

- Random rotations in the range $[-5, 5]$ degrees
- Gaussian blur with sigma in the range $[0.1, 2]$
- Random resized crops with a scale chosen randomly in the range $[0.7, 1]$
- Elastic transform with alpha 0.1 and sigma 0.1

- Color jitter with brightness and contrast to 0.1 (grayscale images)

This step results in the model presented for ChestMNIST and all 3D datasets except AdrenalMNIST3D, for which we did not run the grid search due to time constraints. Instead, see subsection 5.

5.4 Combining Train and Validation Datasets

For BreastMNIST and PneumoniaMNIST (which are the less expensive to train), we use the best hyperparameters and augmentation from previous steps and train the model on the combination of the train and validation datasets. We save the last epoch, which results in the model presented.

References

[1] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 3

[2] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. *CoRR*, abs/1711.11248, 2017. 2

[3] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. MedMNIST v2 - a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1), jan 2023. 1, 2, 3, 4, 10

[4] Tianbao Yang and Yiming Ying. Auc maximization in the era of big data and ai: A survey. *ACM Comput. Surv.*, 55(8), dec 2022. 1

[5] Zhuoning Yuan, Zi-Hao Qiu, Gang Li, Dixian Zhu, Zhishuai Guo, Quanqi Hu, Bokun Wang, Qi Qi, Yongjian Zhong, and Tianbao Yang. Libauc: A deep learning library for x-risk optimization, 2022. 2

[6] Zhuoning Yuan, Yan Yan, Milan Sonka, and Tianbao Yang. Large-scale robust deep auc maximization: A new surrogate loss and empirical studies on medical image classification, 2021. 1, 2, 9