Effect of Stage Training for Long-Tailed Multi-Label Image Classification

Yosuke Yamagishi, Shohei Hanaoka
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, Japan
{yamagishi-yosuke0115, hanaoka-tky}@g.ecc.u-tokyo.ac.jp

Abstract

In this study, we focus on the multi-stage training approach for training image classification models in the ICCV CVAMD 2023 Shared Task CXR-LT: Multi-Label Long-Tailed Classification on Chest X-Rays. In the proposed approach, the input image size and batch size are adjusted at each stage of the training process. In the first stage, we use a smaller input image size and a larger batch size for model training. Following that, we increase the image size and reduce the batch size in the second stage. A thorough search of the related literature did not yield validations of a similar approach for data with a long-tailed distribution. We successfully balance accelerated model training and performance by combining the proposed technique with various enhancements, such as oversampling, postprocessing using view positions, and ensemble methods, despite using a smaller model architecture and smaller input image size.

1. Introduction

In image recognition, several previous studies have investigated the construction of deep learning models, where increasing the image size has gradually led to reduced training time and an improvement in the final accuracy [7, 16, 12, 11]. We demonstrated it by adopting a simpler approach, specifically via stage training, where the training is divided into multiple stages instead of gradually increasing the input size during training. Consequently, high-performing models can be constructed more efficiently.

The setting of the batch size during the first stage of training was another interesting factor. It is well-known that increasing the batch size accelerates the training models. On the other hand, there are contrasting opinions on the impact of batch size changes on the performance of the model [10, 15]. Increasing the batch size can accelerate the training process; however, limitations due to the VRAM of the GPU may force us to lower the image size. The performance can be maintained by lowering the input image size and further increasing the batch size during the first stage of stage training, enabling efficient optimization of the training process.

Previous studies have applied stage training by dividing the training process into stages on different datasets [8]. There have been no reported investigations specifically focusing on the same dataset. Prior studies have addressed long-tailed data in chest X-ray images [6], exploring various loss functions, and data augmentation techniques. We validated the proposed method using the dataset from the ICCV CVAMD 2023 Shared Task CXR-LT: Multi-Label Long-Tailed Classification on Chest X-Rays [5, 4].

In this study, we propose the use of a multi-stage training approach to enhance performance in the context of long-tailed multi-label classification tasks. Specifically, we aim to examine how the batch size and input image size in the first stage of stage training affect the performance of the final model. To this end, we conducted experiments using the CXR-LT dataset, which contains chest X-rays with a long-tailed distribution of multi-label classes. Our findings shed light on whether increasing the batch size during the training of the model on low-resolution images in the first stage accelerates the training and maintains the performance.

2. Dataset

In this study, we used only the MIMIC-CXR-JPG [4, 9] data and labels provided in the ICCV CVAMD 2023 Shared Task CXR-LT. No external data were used. We employed the holdout method and split 80% of the data for training and 20% for validation. The MIMIC-CXR-JPG dataset contains duplicated studies, incorporating image data acquired concurrently from the same patients and including images captured from various perspectives, including frontal and lateral views. To ensure that the same study does not mix both the training and validation data, we performed the splitting using the ”GroupKFold” function in scikit-learn. Consequently, the training data were divided into 211,879 images, and the validation data into 52,970 images. The distribution of the training and validation data is listed in the Table 2.

Label	All	Train	Valid
Support Devices	0.337	0.334	0.346
Lung Opacity	0.302	0.303	0.298
Cardiomegaly	0.290	0.290	0.293
Pleural Effusion	0.261	0.261	0.263
Atelectasis	0.255	0.255	0.257
Pneumonia	0.182	0.181	0.182
No Finding	0.158	0.158	0.157
Edema	0.146	0.146	0.145
Enlarged Cardiomediastinum	0.114	0.114	0.114
Consolidation	0.061	0.060	0.063
Pneumothorax	0.057	0.057	0.054
Fracture	0.045	0.045	0.044
Infiltration	0.039	0.039	0.039
Nodule	0.029	0.029	0.031
Mass	0.021	0.021	0.020
Calcification of the Aorta	0.016	0.016	0.018
Emphysema	0.016	0.016	0.015
Hernia	0.015	0.015	0.015
Pleural Thickening	0.013	0.013	0.013
Tortuous Aorta	0.013	0.014	0.012
Lung Lesion	0.010	0.010	0.009
Subcutaneous Emphysema	0.009	0.009	0.009
Fibrosis	0.004	0.004	0.004
Pleural Other	0.003	0.003	0.003
Pneumomediastinum	0.003	0.003	0.003
Pneumoperitoneum	0.002	0.002	0.002

Table 1. Overall label ratio is stored in the ”All” column, while the training data ratio is stored in the ”Train” column, and the validation data ratio is stored under the ”Valid” column.

3. Method

In this study, we conducted multi-stage training to train the model. Specifically, we created models under various learning conditions and compared their performance on the validation data and development data. The summarized information is presented in Table 5.

3.1. Multi-Stage Training

For training the model, we adopted multi-training with varying image resolutions and batch sizes. In this study, we used a maximum of three stages for multi-staging. A summary of the training is presented in Table 5.

In the first stage, the model was trained using the lowest resolution of 224x224 pixels. To compare the impact of the batch size, we created two models with batch sizes of 256 and 128, respectively.

In the second stage, we divided the batch size into two options. The first option has a batch size of 128 and a resolution of 320x320 (referred to as model2-5 in Table 5), and the second option has a batch size of 80 and a resolution of 384x384 (referred to as model6 in Table 5).

In the third stage, we further increased the resolution to 512x512 and set the batch size to 48.

Oversampling, as described below, was not applied in the first stage. In the second stage, oversampling was added to some models, while others did not.

3.2. Oversampling

For some models, we applied simple oversampling by duplicating data for certain labels. Determining which data to oversample was based on the predictions made on the validation data of ”model2”, as indicated in Table 5. Specifically, the AP was calculated for each label based on the predictions on the validation data, and we oversampled the labels' data that were below the threshold of 0.10 or 0.15. When the threshold is set to 0.1, the following labels' data are doubled: ”Infiltration,” ”Lung Lesion,” ”Pleural Other,” ”Pleural Thickening,” ”Pneumomediastinum,” ”Pneumoperitoneum,” and ”Tortuous Aorta.” When the threshold is set to 0.15, the data of the following labels are doubled: ”Calcification of the Aorta,” ”Fibrosis,” ”Fracture,” ”Infiltration,” ”Lung Lesion,” ”Nodule,” ”Pleural Other,” ”Pleural Thickening,” ”Pneumomediastinum”, ”Pneumoperitoneum”, and ”Tortuous Aorta”.

3.3. Test Time Augmentation

Test time augmentation (TTA) was performed during inference. TTA is a well-known technique for improving the accuracy at inference time [14]. It entails generating augmented versions of the original image and feeding both the original and augmented images into the model for prediction. The predictions from these multiple images are then aggregated to obtain the final prediction. In this study, we used the original images and their horizontally flipped versions, utilizing them as inputs to the model to obtain prediction values. We then obtained the mean of these prediction values as the final prediction.

3.4. Postprocessing

The MIMIC-CKR-JPG dataset includes the position of the image captured in the ”ViewPosition” column, which is broadly classified into frontal and lateral views. It was observed that some labels in the competition had better performance when evaluated from the frontal view. As indicated in Table 5.3, the frontal view achieved better results for a majority of labels. Therefore, we first calculated the performance difference between the frontal and lateral views using the validation data of model3. For studies where both frontal and lateral views were available, we replaced the predictions with those from the frontal view for labels that performed better in the frontal view. In cases where multiple frontal views were captured, we calculated the mean

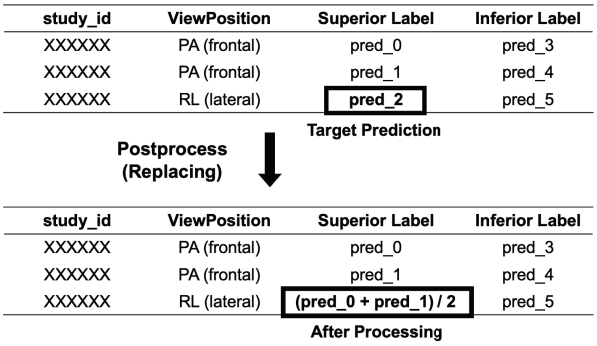

Figure 1. Postprocessing flowchart. "Superior Label" refers to values where the prediction AP of frontal images was superior to those of lateral images, while "Inferior Label" refers to values where the prediction AP is inferior. In cases where both frontal and lateral images were available in the same study, we calculated the mean of the frontal image predictions and replaced it with the profile image predictions.

of their predictions as the final prediction. We summarize the AP per label for the effects of the base model used, ensemble techniques, and post-processing. Figure 1 is the flowchart for this postprocessing.

3.5. Ensemble

It is well known that ensemble methods, which aggregate predictions from multiple models, can improve predictive performance. We developed a total of four models to obtain the final prediction via an ensemble of these four models (model3, 5, 6 and 7). To obtain the final prediction, we used a simple average of the predicted values from the four models and applied postprocessing.

4. Experiment

4.1. Environment

The model trainings and inferences were conducted on an RTX 3090 GPU with 24GB of VRAM. Python 3.7.12 was used to construct the deep learning models, and the PyTorch framework version 1.12.1 was employed.

4.2. Preprocessing

First, we resized the images uniformly. In the first stage, the input image size was set to 224x224 pixels. In the second stage, it was increased to either 320x320 or 384x384 pixels. Lastly, for the third stage, the image size was set to 512x512 pixels. Data augmentation was performed using the Albumentations library [1]. We applied random horizontal flipping ("HorizontalFlip" function), vertical flipping("VerticalFlip" function), and cutout ("CutOut" function) [3], which black out up to 5% of the image size with a maximum of five cutout regions. The probability of applying each augmentation was set to 50%. Finally, we normalized the images using the mean and standard deviation of the ImageNet dataset before using them as inputs to the model [2]. This data augmentation was consistent across all models during training.

4.3. Model

We utilized the EfficientNetV2-S architecture as the model [16]. EfficientNetV2 was introduced as an architecture that achieves high performance with smaller models. In the EfficientNetV2 series, we adopted the model architecture with 24M parameters, which has the lowest performance but also the fewest parameters. The architecture and pre-trained weights were obtained from the timm library [17]. Among the available pre-trained weights, we selected those that were pre-trained using the 22k ImageNet dataset and fine-tuned using the 1k ImageNet dataset [2].

4.4. Training Settings

For the scheduling of the learning rate, we used a warm-up period, reaching the maximum learning rate, followed by a cosine curve decay. We employed binary cross-entropy as the loss function. AdamW with a weight decay of 0.000001 was used as the optimizer [13]. We calculated the average precison (AP) for each label and the mean average precision (mAP) at the end of each epoch and selected the best one among all epochs as the model used for final predictions. In this process, we used the scikit-learn "label_ranking_average_precision_score" function to calculate the mAP. The detailed parameters, including the number of warm-up epochs, total epochs, and the highest learning rate, are listed in Table 5.

5. Results

The mAP for all models in both validation data and development data is documented in Table 5. Additionally, the results from multi-stage training, oversampling, postprocessing, and ensemble techniques are presented in Table 5.4, including the results for each label.

5.1. Effect of Multi-stage Training

In the first stage of training, two variations were employed for the batch size: 256 and 128 (model0 vs model1). Upon completing the first stage, the model with a batch size of 256 outperformed the model with a batch size of 128, including validation data (0.282 vs 0.268) and development data (0.298 vs 0.256).

As mentioned above, oversampling was applied using the weights of model0 and model1, and the performance of the second stage model was compared. When comparing the model constructed using the larger batch size in stage 1

Model	Stage	Pre-Weight	Image size	Batch size	Max LR	OS-threshold	Val mAP	Dev mAP
model0	1st	ImageNet	224x224	256	0.001	N/A	0.282	0.298
model1	1st	ImageNet	224x224	128	0.001	N/A	0.268	0.256
model2	2nd	model0	320x320	128	0.001	N/A	0.300	0.302
model3	2nd	model0	320x320	128	0.001	0.10	0.303	0.304
model4	2nd	model1	320x320	128	0.001	0.10	0.297	0.298
model5	2nd	model0	320x320	128	0.001	0.15	0.299	N/A
model6	2nd	model0	384x384	80	0.001	0.10	0.301	0.306
model7	3rd	model3	512x512	48	0.001	0.10	0.305	0.304

Table 2. All models employ EfficientNetV2-S as the backbone with 30 epochs, utilizing the first epoch for learning rate warm-up. The "Model" column lists the names of the models mentioned in this paper. The "Stage" column indicates the training stage, and for the 2nd and 3rd stage, the total number of epochs is 60 and 90, respectively. The "Pre-Weight" column represents the pre-trained weights used at the start of training. For stage 1 models, ImageNet was used, while either model0 or model1 was used for stage 2 models. "Image size" refers to the dimensions of the input images used for training the model. "Batch size" represents the batch size used during model training. "Max LR" denotes the maximum learning rate after warm-up. The "OS-threshold" column contains the threshold value used for oversampling. For instance, if the threshold is set to 0.1, it indicates that any label in the validation set of "model2" with an mAP below 0.1 is duplicated, and N/A means no oversampling was applied. "Val mAP" represents the average mean average precision for all labels on the validation data, while "Dev mAP" represents the average mean average precision for all labels on the development data. The models in bold are the four used in the ensemble for the final predictions.

with the model that uses the smaller batch size (model3 vs model4), the former outperformed the latter in both validation data (0.303 vs 0.297) and development data (0.304 vs 0.298).

Furthermore, model3 was utilized to train the third stage (model7). The input image size for the third-stage model was expanded to 512x512. The results for the valid data were slightly better in model7 compared to model3 (0.305 vs 0.303); however, no improvement was observed in the development data (0.304 vs 0.304).

In the first stage of multi-stage training, we accelerated the learning process by using a smaller input image size and a larger batch size (input image size: 224x224, batch size: 256). Furthermore, we compared the behavior of the loss function and scores when we used a larger input size and a smaller batch size for stage 1 (input image size: 320x320, batch size: 128). The results are presented in Figure 2. The training was conducted until epoch 30 for stage 1, and from epoch 31, it transitioned to stage 2 training.

At epoch 31, the scheduler was reset, resulting in a temporary degradation in the scores for both cases. However, ultimately, better results were achieved in both scenarios compared with the final outcome of the first stage. The final values for the two models are as follows: the validation loss function was 0.1846 for the model with input image size 224x224 and batch size 256, and the validation loss function was also 0.1846 for the model with input image size 320x320 and batch size 128, both values are equal.

Regarding mAP, the former achieved a value of 0.3003, while the latter scored 0.2995. This indicates that the model with an smaller input image size of 224x224 and a larger batch size of 256 was slightly superior in terms of mAP performance.

Therefore, our proposed method can complete the training faster than the conventional approach, and the obtained results are nearly identical or slightly better.

5.2. Effect of Oversampling

We also evaluated the impact of oversampling by dividing the threshold for the procedure. The results for the validation data were calculated for three models: one without oversampling, one with oversampling at a threshold of 0.10, and one with oversampling at a threshold of 0.15 (model2 vs model3 vs model5). The model with oversampling performed at a threshold of 0.10 exhibited the best performance, with scores of 0.300 vs 0.303 vs 0.299. Moreover, the performance of the models without oversampling and with oversampling at a threshold of 0.10 was also compared using the development data (model2 vs model3), resulting in scores of 0.302 vs 0.304, where the model with oversampling again outperformed the other.

5.3. Difference in AP per View Position

As mentioned in the methods section, the MIMIC-CXR-JPG dataset contains multiple views of the same study. These views can be broadly categorized into frontal and lateral images. In many labels, it is expected that the frontal images are easier to interpret compared to lateral images. Therefore, following the method described, a post-processing step was performed to replace the predictions of lateral images with the predictions of frontal images.

The labels for performing the post-processing were determined based on the performance of model3 on the validation data. The results are summarized in Tabel 5.3.

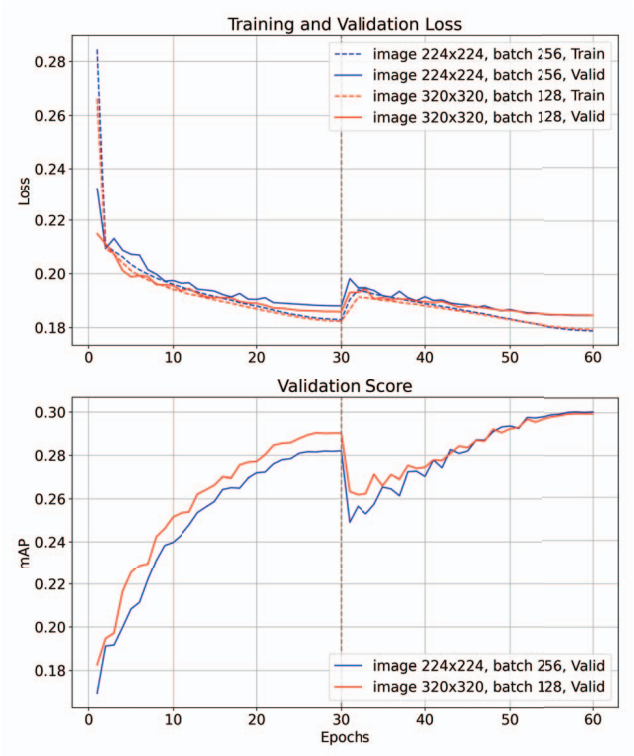

Figure 2. Upper graph illustrates the curves of the loss function during both training and validation, while the lower graph represents the mAP curve on the validation data. The first stage encompasses the training process up to the 30th epoch, and the second stage extends up to the 60th epoch. In the first stage of training, the input image size was set to 224x224, and the batch size was 256 ("image 224x224, batch 256"). In the same way, during the first stage, another experiment was conducted with an input image size of 320x320 and a batch size of 128 ("image 320x320, batch 128"). In the second stage, both experiments used an input image size of 320x320 and a batch size of 128, which were kept the same across both experiments.

Label	Front	Lateral	Dif
Support Devices	0.909	0.732	0.177
Lung Opacity	0.573	0.538	0.035
Cardiomegaly	0.671	0.624	0.047
Pleural Effusion	0.808	0.811	-0.003
Atelectasis	0.600	0.509	0.091
Pneumonia	0.293	0.302	-0.009
No Finding	0.450	0.457	-0.007
Edema	0.540	0.399	0.141
Enlarged Cardiomediastinum	0.183	0.150	0.033
Consolidation	0.228	0.181	0.047
Pneumothorax	0.420	0.242	0.178
Fracture	0.128	0.088	0.040
Infiltration	0.051	0.055	-0.004
Nodule	0.097	0.080	0.017
Mass	0.215	0.112	0.103
Calcification of the Aorta	0.109	0.093	0.016
Emphysema	0.182	0.213	-0.031
Hernia	0.486	0.647	-0.161
Pleural Thickening	0.085	0.061	0.024
Tortuous Aorta	0.053	0.050	0.003
Lung Lesion	0.035	0.038	-0.003
Subcutaneous Emphysema	0.489	0.238	0.251
Fibrosis	0.127	0.093	0.034
Pleural Other	0.055	0.013	0.042
Pneumomediastinum	0.258	0.036	0.222
Pneumoperitoneum	0.185	0.065	0.120
Mean	0.317	0.263	0.054

Table 3. We calculated the mean average precision for the predictions of model3 on the validation data, separating them into two groups: "Front" (images with ViewPosition of AP or PA) and "Lateral" (images with ViewPosition other than AP or PA). "Dif" represents the result obtained by subtracting the AP of lateral from the AP of frontal. The labels with a higher AP in the frontal view than in the lateral view are indicated in bold.

As expected, the predictions of frontal images were superior to the predictions of lateral images in the majority of the labels. A significant difference could be observed, particularly in "Subcutaneous Emphysema" with a value of 0.251 and "Pneumomediastinum" with a value of 0.222.

Overall, the difference in mAP between frontal and lateral images was 0.054, indicating that the predictions for frontal images were superior. Specifically, the mAP for frontal images was 0.317, while the mAP for lateral images was 0.263.

5.4. AP for each label

We verified the changes in AP for each label using the development data and test data.

Through training from the 1st stage model (model0) to the 2nd stage model (model2), we observed improvements in AP for both data sets across 22 out of 26 labels. This indicates significant score enhancements in a substantial portion of data with a long-tail distribution. The overall mAP improved from 0.286 to 0.302 for the development data and from 0.291 to 0.308 for the test data.

By applying oversampling, improvements in AP could be observed for the minority labels "Pneumoperitoneum" (validation data ratio of 0.002) and "Pneumomediastinum" (validation data ratio of 0.003) in both data sets. Overall, the mAP improved from 0.302 to 0.305 for the development data, but a slight decrease was observed from 0.308 to 0.307 for the test data.

Furthermore, through post-processing, we witnessed AP improvements for 15 out of the 26 labels in both datasets. Moreover, the overall mAP improved from 0.313 to 0.315.

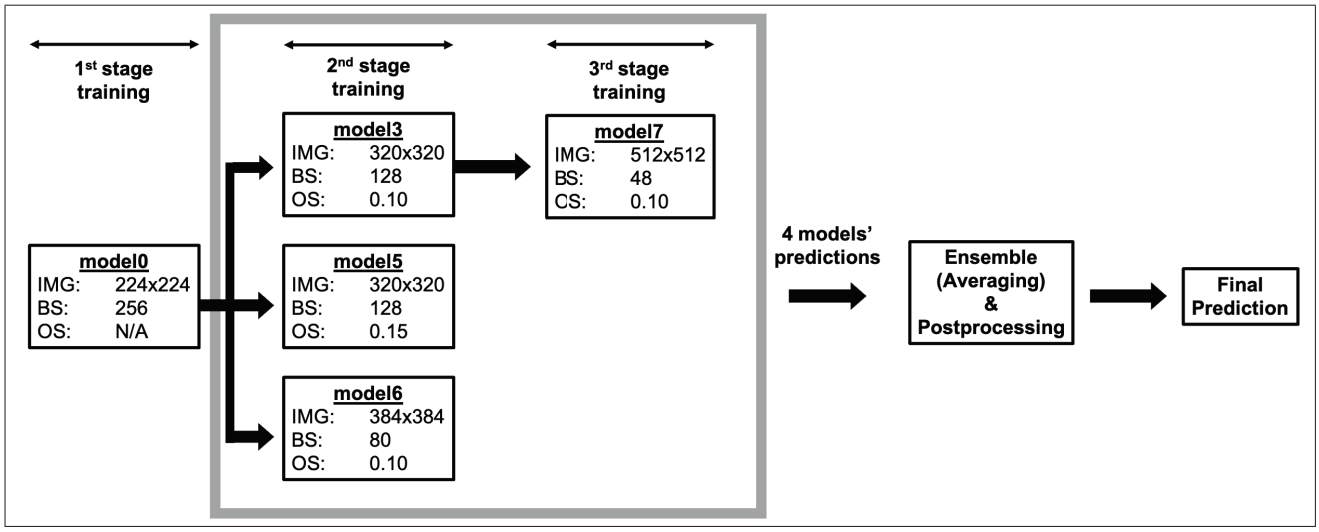

Figure 3. This figure depicts the flowchart of the model's stage training and inference. Firstly, in the 1st stage, training is conducted with an input image size (IMG) of 224x224 and a batch size (BS) of 256. The 2nd stage models (model3, model5, model6) used for the final inference are represented, and OS denotes the threshold used during oversampling. Furthermore, during the training of the 3rd stage, model7 was created. Inference from the four models (2nd stage and 3rd stage) is combined through averaging to perform an ensemble. Finally, post-processing is applied to obtain the final prediction.

The ensemble improved scores for 23 out of the 26 labels. The overall mAP improved from 0.313 to 0.326 for the development data and from 0.315 to 0.330 for the test data.

The flowchart for multi-stage training and inference for making the final prediction is summarized in Figure 3.

6. Conclusion

In the first stage, when tested with a consistent input image size of 224x224, larger batch sizes consistently yielded better accuracy. We further leveraged the first-stage model's superior-performing weights as pretrained weights for the second-stage model, leading to enhanced performance in the second stage. Interestingly, when comparing two first-stage models — one trained with an input size of 224x224 and a batch size of 256, and another with an input size of 320x320 and a batch size of 128 — the former, despite underperforming at the end of the first stage, matched the latter's performance in the second stage. This indicates that, even with the acceleration of the initial training phase, our approach establishes a pathway to an efficient learning process without compromising the final performance.

Furthermore, we demonstrated that employing multiple techniques, such as oversampling, postprocessing, and ensemble, can improve scores even with long-tail data. In particular, the ensemble method improved scores for 23 out of 26 labels. Although the models used in this ensemble all share the same architecture and were trained using the same data, which might suggest limited model diversity, the significant score improvements indicate that even minor differences in training settings or input image sizes can lead to increased model diversity. This makes these results particularly valuable.

As a limitation in this study, the learning rate was kept the same even when changing the batch size. It has been suggested that adjusting the learning rate when changing the batch size would be beneficial; however, this would also depend on the scheduler, which would require thorough considerations.

In this experiment, we achieved efficient learning by setting a larger batch size and a smaller input image size in the first stage of training. This approach potentially overcomes the limitations of GPU VRAM by suppressing memory consumption and optimizing learning speed and efficiency.

The method used in this study is facile and has high generality, making it applicable to various types of data, not just those with long-tail distributions. There is also potential to apply this method to multitask scenarios.

Label	1st stage	2nd stage	Oversampling	Postprocess	Ensemble
Support Devices	0.870 / 0.865	**0.887 / 0.884**	0.885 / 0.881	**0.888 / 0.885**	**0.897 / 0.894**
Lung Opacity	0.580 / 0.561	**0.593 / 0.571**	0.588 / 0.569	**0.600 / 0.581**	**0.611 / 0.590**
Cardiomegaly	0.646 / 0.647	0.646 / **0.648**	0.644 / 0.646	**0.663 / 0.665**	**0.669 / 0.668**
Pleural Effusion	0.802 / 0.810	**0.808 / 0.817**	0.805 / 0.816	0.807 / 0.816	**0.812 / 0.822**
Atelectasis	0.569 / 0.575	**0.579 / 0.583**	0.578 / 0.583	**0.593 / 0.596**	**0.600 / 0.602**
Pneumonia	0.288 / 0.281	**0.296 / 0.289**	0.290 / 0.286	0.283 / 0.286	**0.301 / 0.292**
No Finding	0.459 / 0.462	**0.461 / 0.467**	**0.464** / 0.464	0.460 / 0.464	**0.470 / 0.471**
Edema	0.525 / 0.521	**0.536 / 0.532**	0.533 / **0.533**	**0.545 / 0.543**	**0.551 / 0.551**
Enlarged Cardiomediastinum	0.166 / 0.171	**0.168 / 0.177**	0.166 / 0.176	0.168 / **0.181**	**0.171 / 0.183**
Consolidation	0.210 / 0.203	0.210 / **0.206**	0.205 / 0.202	**0.212 / 0.216**	**0.216 / 0.225**
Pneumothorax	0.350 / 0.367	**0.399 / 0.418**	0.393 / 0.416	**0.407 / 0.431**	**0.421 / 0.451**
Fracture	0.102 / 0.107	**0.123 / 0.120**	**0.124** / 0.120	**0.146 / 0.133**	**0.186 / 0.171**
Infiltration	0.061 / 0.053	**0.064 / 0.056**	0.063 / 0.055	0.064 / 0.055	0.064 / 0.056
Nodule	0.098 / 0.093	**0.113 / 0.105**	0.111 / 0.103	**0.117 / 0.112**	**0.153 / 0.137**
Mass	0.164 / 0.152	**0.166 / 0.168**	0.163 / 0.165	**0.185 / 0.175**	**0.198 / 0.187**
Calcification of the Aorta	0.081 / 0.075	**0.110 / 0.103**	0.109 / 0.098	**0.141 / 0.112**	**0.156 / 0.130**
Emphysema	0.268 / 0.149	0.263 / **0.155**	**0.278** / 0.146	**0.284** / 0.146	**0.279 / 0.161**
Hernia	0.478 / 0.457	**0.506 / 0.487**	**0.513** / 0.481	**0.491** / 0.481	**0.531 / 0.499**
Pleural Thickening	0.063 / 0.063	**0.078 / 0.073**	0.069 / **0.078**	**0.094 / 0.105**	**0.114 / 0.119**
Tortuous Aorta	0.059 / 0.052	**0.064 / 0.055**	**0.065** / 0.055	**0.069 / 0.058**	**0.066 / 0.063**
Lung Lesion	0.055 / 0.029	**0.068 / 0.030**	0.062 / 0.028	0.055 / 0.028	0.064 / **0.031**
Subcutaneous Emphysema	0.338 / 0.457	**0.427 / 0.488**	0.416 / 0.457	**0.450** / 0.471	**0.465 / 0.507**
Fibrosis	0.087 / 0.105	**0.117 / 0.108**	0.114 / **0.110**	0.117 / 0.106	**0.148 / 0.116**
Pleural Other	0.017 / 0.024	**0.022 / 0.036**	**0.037** / 0.026	0.028 / 0.032	**0.050 / 0.059**
Pneumomediastinum	0.075 / 0.223	0.075 / **0.266**	**0.139 / 0.296**	**0.152 / 0.319**	0.149 / **0.326**
Pneumoperitoneum	0.020 / 0.076	**0.085 / 0.175**	**0.123 / 0.195**	**0.129 / 0.206**	**0.134 / 0.262**
Mean	0.286 / 0.291	**0.302 / 0.308**	**0.305** / 0.307	**0.313 / 0.315**	**0.326 / 0.330**

Table 4. The table above shows the AP for each label and the mAP. Each AP is presented in the format of Development Data / Test Data. Moreover, the AP that achieved the best performance up to that point is highlighted in bold. The table labels are arranged in descending order of frequency in the data, following the same order as the Tabel 2 The "1st stage" refers to model0 (trained with a size of 224x224 and a batch size of 256), while the "2nd stage" refers to model2 (based on the 1st stage model and trained with a size of 320x320 and a batch size of 128). "Oversampling" indicates the model trained with oversampling in addition to the training conditions of model2 (model3). "Postprocess" represents model3 with the post-processing method described in the paper. "Ensemble" denotes the AP obtained by averaging the predictions from model3, 5, 6, and 7.

References

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[3] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[4] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, ..., and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[5] Gregory Holste, Song Wang, Ajay Jaiswal, Yuzhe Yang, Mingquan Lin, Yifan Peng, and Atlas Wang. CXR-LT: Multi-Label Long-Tailed Classification on Chest X-Rays. https://physionet.org/content/cxr-lt-iccv-workshop-cvamd/1.0.0/.

[6] Gregory Holste, Song Wang, Ziyu Jiang, Thomas C. Shen, George Shih, Ronald M. Summers, Yifan Peng, and Zhangyang Wang. Long-tailed classification of thorax diseases on chest x-ray: A new benchmark study. In *Lecture Notes in Computer Science*, pages 22–32. Springer Nature Switzerland, 2022.

[7] Jeremy Howard. Training imagenet in 3 hours for usd 25; and cifar10 for usd 0.26. https://www.fast.ai/posts/2018-04-30-dawnbench-fastai.html, 2018. Accessed: July 15, 2023.

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[9] Alistair Johnson, Matthew Lungren, Yifan Peng, Zhiyong Lu, Roger Mark, Scott Berkowitz, and Steven Horng. Mimic-cxr-jpg - chest radiographs with structured labels (version 2.0.0). PhysioNet, 2019. Accessed: 2023-07-15.

[10] Ibrahem Kandel and Mauro Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6(4):312–315, 2020.

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018.

[12] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution, 2017.

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[14] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. Better aggregation in test-time augmentation, 2021.

[15] Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size, 2018.

[16] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training, 2021.

[17] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.