ISTANBUL TECHNICAL UNIVERSITY

Istanbul Technical University
Faculty of Computer and Informatics

CoronaHack - Chest X-Ray Classification Challenge

Deep Learning Final Project Report

Authors
Batuhan Faik Derinbay - 150180705 - derinbay18@itu.edu.tr
Ufuk Demir - 150170710 - demiruf17@itu.edu.tr

July 24, 2021

Contents

Introduction

1.1 Abstract

Convolutional neural networks lead to a more accurate and more accessible diagnosis of various diseases on chest radiography. With it, publicly available radiography datasets became more common and easy to access. In this report, state of the art deep learning techniques are explored and evaluated on the problem of classifying chest x-ray images using CoronaHack Chest X-Ray dataset that is publicly available for all to see on the Kaggle CoronaHack web page.

1.2 Background

COVID-19 global pandemic caused massive difficulties in densely populated nations. It resulted in nearly all countries to go under lock-downs and apply restrictive measures to people. Yet, over 50 million people got affected by the virus, and more than 2 million people died because of it. Because of this, researchers around the globe have collaborated to better identify and treat COVID-19 patients.

One of the most widely available methods for identifying COVID-19 affected patients is using chest radiography. Even though computed tomography (CT) is the best radiography method for diagnosing coronavirus, X-Ray is the first step for doctors to analyze because of its high availability and low cost. With the increasing demand for analysis of diseases using x-rays due to the global pandemic, fellow enthusiast Praveen Govindaraj put together a chest x-ray dataset, named CoronaHack, and put it on Kaggle for all researchers throughout the globe to work on.

In the scope of the final project, students are asked to select among one of three problems and to work on that specific problem. Knowing the similarities of the challenges between their interim project and the final project, classifying healthy and infectious chest x-rays provided in the CoronaHack dataset, using deep learning techniques was selected. By implementing various state of the art novelties, the team composed of Batuhan Faik Derinbay and Ufuk Demir wanted to work on this challenge in order to increase the

performances achieved by existing models.

Methodology

2.1 Data Preparation

Dataset used within the scope of the project is provided by Praveen Govindaraj and can be found on the website. It features 5910 labelled chest x-ray images with a total size of 1.3 gigabytes and various resolutions. The number of images per class and their relative percentages to the dataset are given in Table 2.1. By examining the table, it is clear that there is a huge data imbalance between classes. To overcome this problem, the team tried many techniques and novelties, which are explained in detail in section 2.2.

Moreover, since the given dataset consists of training and test sets, the team decided to do a train-val split with $80 - 20$ percent respectively on the training set. After some inspection, abnormal and outlier images from **only** the training set are removed. Examples of the outliers and the normal images found in the dataset are given in Figure A.1. It should be noted that only the images from different viewing angles are removed. For others, such as inaccurate colored x-rays, data augmentation methods (explained in section 2.2) are applied. In the end, the team ended up with 624 never seen test data, 1057 validation data, and 4225 training data. These numbers represent 11%, 18%, and 71% of the total dataset, respectively. Each implemented method and combinations of methods are trained on the same training and validation set and never seen the 11% test data.

	# of Images	**Percentage**
Healthy	1576	26.67
Stress-Smoking	2	0.03
Virus	1493	25.27
COVID-19	58	0.98
SARS	4	0.07
Bacteria	2772	46.90
Streptococcus	5	0.08
Total	5910	100

Table 2.1: Distribution of the Data

2.2 Data Augmentation, Model Selection and Hyperparameter Tuning

Data augmentation was one of the biggest challenges of this dataset. Despite the low number of samples, there was a huge data imbalance between classes, and images were not consistent. Even though the test set was sufficient, the training set had many outlier images with blue-toned x-rays, top and side view of the chests, various sizes of chests with not-so-well aligned spinal cords are just a few examples. Moreover, both the training and the test set were composed of images taken from different x-ray machines with diverse resolutions.

After inspecting the test set, since there were no top or side viewed chest x-rays, such outliers were removed from the training set. Then one of the six classes (Stress-Smoking) was removed due to low data count, and the remaining five classes were merged into three classes as described in the metadata CSV of the dataset. In the end, there were three distinct labels, healthy, virus-infected (included data from virus, COVID-19, and SARS labels), and bacteria-infected (included data from bacteria and streptococcus labels). Besides, the team implemented the widely used oversampling technique to overcome the data imbalance problem.

When it comes to the classification task, the team decided to test three different methods to see which one performs the best. The first one is to train binary classification models by merging virus and bacteria labels and classifying healthy and infected images. The second one is to train multi-class classification models with three classes. The third one is a novel approach that trains multi-class classification models with binary classes (healthy, infected) called multi-to-binary classification, which is explained in more detail in section 2.3.

Baseline models for all three approaches were trained using randomly initialized DenseNet-121 [1] and ResNet-101 [2] models. These baseline models were selected thanks to their high performances on x-ray classification tests and state of the art designs. Unlike the interim project, good results were more substantial than novel implementations, so there was not a need to train so many models. Therefore, relatively large models were selected due to their high accuracies, and both baseline models were trained with and without oversampling, with added novelties and their combinations.

All models are fed with the same transformed input data. Applied transformations are, resize the short edge to 224 pixels (preserving the aspect ratio), center crop an area of 224×224 pixels, convert to grayscale, and normalize using the pre-calculated mean and standard deviation values. During training, random perspective distortion with a scale of 0.1 was also applied.

All the baseline models' training runs used randomly initialized weights with the same seed and the ADAM optimizer with cross-entropy loss. After using grid search for the best set of hyperparameters, the best-obtained parameter set is kept constant between models in order to get more comparable results from the models. Due to highly imbalanced classes, accuracy, sensitivity, specificity, and weighted F1 scores were considered when choosing the best set of parameters. The training runs used a learning rate of $\alpha = 0.001$ and reduced it by a factor of 10 for every 5 epoch that did not decrease the loss. Loss curves given in Figure A.3 are obtained. Note that only loss curves of the models that

are given in Tables 3.1, 3.2 and 3.3 are present in Figure A.3. In addition, for the last two models, confusion matrices are given in A.4.

2.3 Label Augmentation

The big difference in the number of samples belonging to the classes affects the model performance. To avoid this problem, methods such as oversampling, under-sampling are used. With these methods, it is aimed to balance the number of samples by creating new data belonging to the minority class or reusing them, or reducing the number of samples belonging to the majority class. As stated in [3], new self-supervised label augmentation techniques are used to avoid the effect of data imbalance. As stated in [3], new self-supervised label augmentation techniques are also used to avoid the effect of data imbalance.

The most important part of the self-supervised label augmentation techniques is learning a single unified task concerning the original and self-supervised labels' joint distribution. To learn the unified task, labels are augmented via self-supervision of input transformation. In the [3], rotation and color permutation of images are used to augment the labels.

In this project, the rotation and color permutation-based self-supervised label augmentation techniques are not used directly because these techniques are not suitable for the images used in the project. In the dataset used in this project, there is also information about what causes the illness. Using these pieces of information, the number of labels is increased to three as healthy, virus, and bacteria (stress smoking is ignored due to the number of samples). When a model is trained with augmented labels, the model learns the joint probability distribution on all possible combinations of 3 labels. A simple illustration of the used label augmentation technique is given in Figure 2.1.

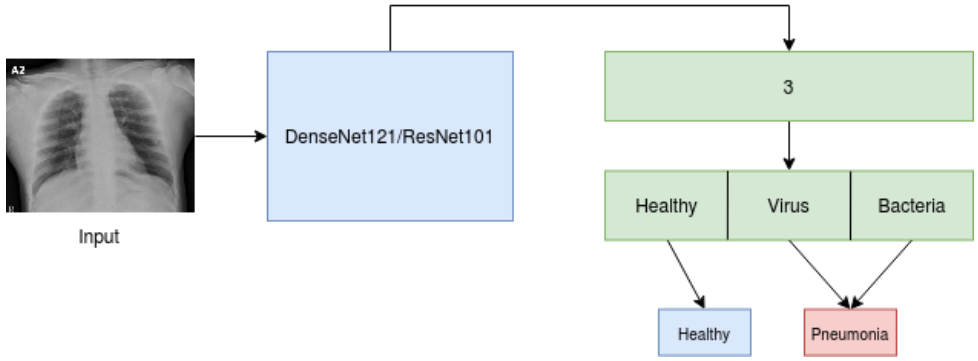


Figure 2.1: Illustration of label augmentation technique

2.4 Chest X-Ray Normalization (CXR Norm)

Chest X-Ray normalization is a method for unifying and normalizing the input data. It aims to normalize and increase the uniformity of the input chest x-ray images using traditional computer vision techniques by adjusting the pixel values. It has been showcased on ISBI 2020 [4] with a similar use case, and presenters addressed that it helped increase the stability of their models by a respectable amount.

CXR Norm starts off by applying adaptive histogram equalization to grayscale images in order to improve contrasts of the x-rays. Then it applies median filtering for noise removal and continues with contrast stretching by increasing the range of intensity values to span a specific percentile range of the image. In Figure A.2, the steps applied are presented on an exemplary training image.

To implement CXR Norm, the team has written both a preprocessor and a loader that takes in parameters for the CXR Norm and applies either to the entire dataset or to the batch, respectively. By doing so, tests with and without the CXR Norm are a lot more versatile, and hyperparameter search for the CXR Norm can be applied easily. Note that the preprocessor handles the whole dataset once and creates a new (or overwrites) preprocessed dataset, whereas the loader handles batches during training/testing time, which increases training times significantly. It also turns the CXR Norm process into a relatively compact package, which can be obtained through Python packaging index such as `PyPI`.

Using grid search to find the best CXR Norm parameters resulted in a clipping limit of 2 (the threshold for contrast limiting), the grid size of 8×8 for histogram equalization, a 5×5 median filter kernel, and 2 through 98 percentiles of contrast stretching. After finding a suitable set of parameters and training baseline models using the CXR Norm applied dataset, the team achieved a 3.4% accuracy and 3.9% F1 score increase, which is explained with more details in chapter 3.

2.5 Sharpness-Aware Minimization

Sharpness-aware minimization (SAM) [5] is a state of the art procedure that minimizes the loss value and the loss sharpness simultaneously. It has been proposed by the Google research team and is recently accepted by the International Conference on Learning Representations (ICLR) 2021. In order to turn the cost into a min-max optimization problem for gradient-descent to carry out effectively, SAM searches for parameters that lie in areas having a uniformly low loss. By optimizing the way loss calculations are carried out, SAM achieves the state of the art results on various datasets. Further details and mathematical derivations are given in the original SAM paper "Sharpness-Aware Minimization for Efficiently Improving Generalization", please refer if necessary.

After examining loss curves of the models (E.g., Figure 2.2), the team decided to implement SAM within their training procedure in order to deal with abnormalities seen with the loss. Models that use sharpness-aware minimization with stochastic gradient descent (SGD) optimizer had 0.6% higher accuracy and 0.6% higher F1 score than the baseline models, which are explained in more details in chapter 3. It should be noted that when training with SAM, the team used the SGD optimizer with a learning rate of $\alpha = 0.1$ and reduced it by a factor of 10 for every 5 epoch that did not decrease the loss using a scheduler.

The increase in the accuracy of the models is due to the fact that SAM helps generalize the data better by minimizing both the loss value and the sharpness. However, it comes with a cost. SAM requires gradient calculations of the loss twice. Calculating gradients of the loss two times increases the training time and the need for higher computational power. Moreover, SAM comes with a single hyperparameter ρ, which is tuned via grid

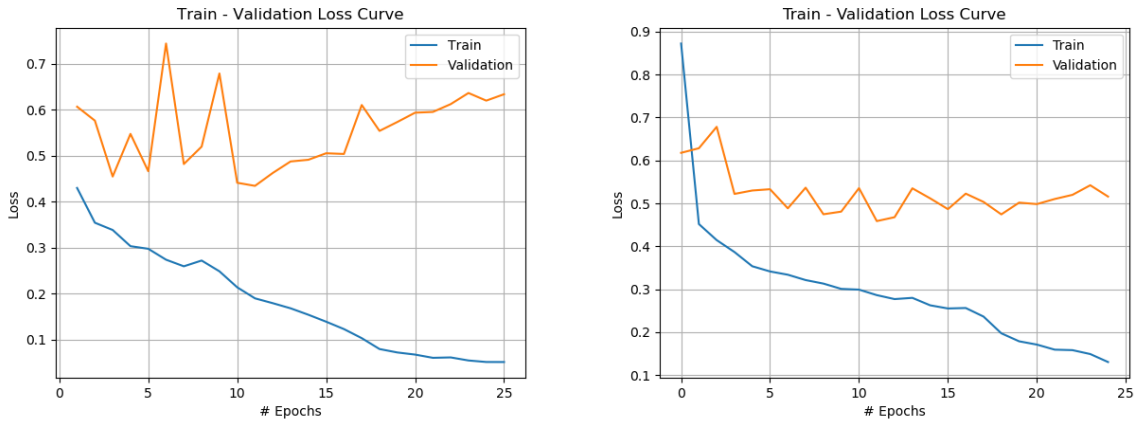
search and is found to be $\rho = 0.05$ as an optimal value in the original paper. In this project, the team did not tune this hyperparameter because of the satisfactory results obtained when using the optimal value.

Within this research project's scope, implementation of the SAM is adopted from a GitHub repository [6], which in turn is also adopted from the GitHub repository of the paper [7]. The code below is from the project sources and depicts the two-step loss gradient calculation in order to calculate the sharpness-aware minimization presented in the paper.

```
# First pass
loss_value = loss(model(img), img_class)
loss_value.backward()
optimizer.first_step(zero_grad=True)

# Second pass
loss(model(img), img_class).backward()
optimizer.second_step(zero_grad=True)
```

Two Step Loss Gradient Calculation



(a) DenseNet-121 Multi-to-Binary with CXR Norm.

(b) DenseNet-121 Multi-to-Binary with CXR Norm + SAM

Figure 2.2: Loss Curves of Identical Models With and Without SAM

Evaluation and Results

In this project, the classification task is tried to handle with three different approaches, which are binary, multi-class to binary, multi-class classification. All methods as mentioned in Part 2 and their combinations are applied to the dataset. Since the dataset

is imbalanced, accuracy is not the only calculation metric. In addition to the accuracy, sensitivity, specificity, and F1 score are also used to measure the performance of the model more accurately. In all variations of the experiments, both non-pre-trained ResNet-101 and DenseNet-121 are used separately, and each model is trained for up to 50 epochs.

As seen in Table 3.1, when comparing the performance of the DenseNet-121 model trained with both oversampled and non-oversampled dataset, it is observed that the oversampling method increases the accuracy and reliability of the model for all task. It is observed that the model does not make a random guess. For the ResNet-101 model, the performance increase is not observed for the binary and multi-class to the binary task but for the multi-class classification task, model performance and reliability increased. It is understood that increasing the number of samples belonging to the minority class affects the performance of the model positively.

It was observed that when the models trained with the dataset, which is applied both oversampling and CXR normalization methods, models performance and reliability increased for all tasks. When Tables 2 and 3 are analyzed, the combination of oversampling and CXR normalization methods have significantly increased the performance of the ResNet-101 model. It is understood that these two methods increased the generalization ability of the models.

As can be seen in Table 3.3, when the above-mentioned methods are applied to the dataset, and the model is trained with SAM, it can be concluded that there is a slight increase in the performance of the model for multi-class to the binary classification task. As suggested in the paper, minimizing loss value and sharpness has enabled the data to be better generalized. Although SAM required high computational power, it made an essential contribution to the classification task.

TEST SCORES	**Binary Classification**			
Models	**Accuracy**	**Sensitivity**	**Specificity**	**Weighted F1**
Baseline DenseNet-121 w/o OS	76.44	75.64	77.78	76.75
Baseline DenseNet-121 w/ OS	88.12	91.04	84.32	88.12
Baseline ResNet-101 w/o OS	86.06	97.18	67.52	85.47
Baseline ResNet-101 w/ OS	85.58	96.15	67.95	85.03
	Multi to Binary Classification			
Models	**Accuracy**	**Sensitivity**	**Specificity**	**Weighted F1**
Baseline DenseNet-121 w/o OS	69.55	57.95	88.89	69.75
Baseline DenseNet-121 w/ OS	87.66	97.95	70.51	87.18
Baseline ResNet-101 w/o OS	89.42	94.10	81.62	89.32
Baseline ResNet-101 w/ OS	87.18	94.87	74.36	86.89
	Multi Class Classification			
Models	**Accuracy**	**Weighted Recall**	**Weighted Precision**	**Weighted F1**
Baseline DenseNet-121 w/o OS	44.07	25.97	37.66	34.30
Baseline DenseNet-121 w/ OS	82.42	82.18	84.10	82.38
Baseline ResNet-101 w/o OS	75.16	75.16	73.99	74.19
Baseline ResNet-101 w/ OS	78.04	78.04	78.00	77.90

Table 3.1: Results of the Models on All Tasks

TEST SCORES	**Binary Classification**			
Models	**Accuracy**	**Sensitivity**	**Specificity**	**Weighted F1**
DenseNet-121 w/o OS + CXR	87.12	94.08	78.46	87.12
DenseNet-121 w/ OS + CXR	87.52	85.40	93.02	87.18
ResNet-101 w/o OS + CXR	87.34	93.08	77.78	87.18
ResNet-101 w/ OS + CXR	86.38	93.59	74.36	86.12
	Multi to Binary Classification			
Models	**Accuracy**	**Sensitivity**	**Specificity**	**Weighted F1**
DenseNet-121 w/o OS + CXR	88.12	89.58	86.88	88.12
DenseNet-121 w/ OS + CXR	90.64	92.14	86.46	90.58
ResNet-101 w/o OS + CXR	83.81	94.87	65.38	83.19
ResNet-101 w/ OS + CXR	87.82	98.46	70.09	87.32
	Multi Class Classification			
Models	**Accuracy**	**Weighted Recall**	**Weighted Precision**	**Weighted F1**
DenseNet-121 w/o OS + CXR	83.02	83.14	83.08	83.10
DenseNet-121 w/ OS + CXR	86.02	86.08	85.12	86.12
ResNet-101 w/o OS + CXR	77.08	77.08	80.26	77.78
ResNet-101 w/ OS + CXR	79.00	79.00	80.10	78.07

Table 3.2: Results of the Models on All Tasks

TEST SCORES	Binary Classification			
Models	**Accuracy**	**Sensitivity**	**Specificity**	**Weighted F1**
DenseNet-121w/OS+CXR+SAM	86.86	97.69	68.80	86.31
ResNet-101w/OS+CXR+SAM	**87.66**	98.46	69.66	**87.14**
	Multi to Binary Classification			
Models	**Accuracy**	**Sensitivity**	**Specificity**	**Weighted F1**
DenseNet-121w/OS+CXR+SAM	**91.12**	93.12	89.24	**91.13**
ResNet-101w/OS+CXR+SAM	88.14	95.59	79.06	88.00
	Multi Class Classification			
Models	**Accuracy**	**Weighted Recall**	**Weighted Precision**	**Weighted F1**
DenseNet-121w/OS+CXR+SAM	**79.65**	79.65	79.61	**79.46**
ResNet-101w/OS+CXR+SAM	76.44	76.44	78.10	76.46

Table 3.3: Results of the Models on All Tasks

Related Work

Classification performance can be increased by performing uncertainty analysis of the model. In addition, as stated in [8], the under-sampling method can be applied to increase the performance of the minority class.

As Hong et al. described in [9] the performance of the classification task can be increased by combining the computational power of graph convolutional networks with convolutional neural networks.

Conclusion

At the beginning of the term project, the data to be used has been analyzed. Literature research was conducted considering the obtained information. As a result of the research, the project started to be developed by applying the methods mentioned in sections 2.1, 2.2, 2.3, 2.4, and 2.5. The project was divided into three different classification tasks, and performance analyzes were made for each. As seen in Table 3.1, 3.2, and 3.3, the added novelties contributed to the performance of the model. In the last part of the project, the

performance of the model was increased by finding suitable hyper-parameters using grid search.

Moreover, the team realized that all methods used in the interim project are almost utterly suitable for use on this dataset.

Bibliography

[1] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *CoRR* abs/1608.06993 (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993 (page 3).

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385 (page 3).

[3] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. *Self-supervised Label Augmentation via Input Transformations*. 2019. eprint: arXiv:1910.05872 (page 4).

[4] Eliot Siegel and Dr. Nicola Sverzelatti. "Radiology Decision Tool for Suspected COVID-19". In: *2020 IEEE/International Symposium on Biomedical Imaging (ISBI)* (Apr. 2020) (page 4).

[5] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. *Sharpness-Aware Minimization for Efficiently Improving Generalization*. 2020. eprint: arXiv:2010.01412 (page 5).

[6] David Samuel. *SAM Optimizer*. https://github.com/davda54/sam. 2020 (page 6).

[7] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. *SAM: Sharpness-Aware Minimization for Efficiently Improving Generalization*. https://github.com/google-research/sam. 2020 (page 6).

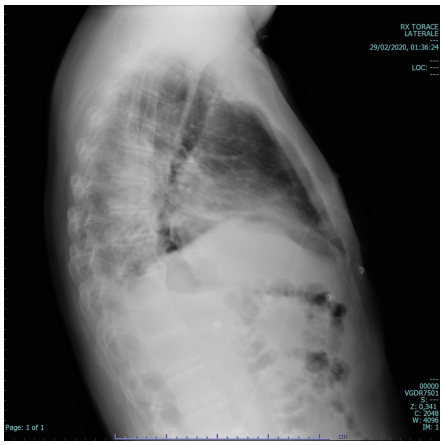
[8] Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. "Clustering-based undersampling in class-imbalanced data". In: *Information Sciences* 409-410 (2017), pp. 17–26. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2017.05.008. URL: http://www.sciencedirect.com/science/article/pii/S0020025517307235 (page 9).

[9] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot. "Graph Convolutional Networks for Hyperspectral Image Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* (2020), pp. 1–13. DOI: 10.1109/TGRS.2020.3015157 (page 9).

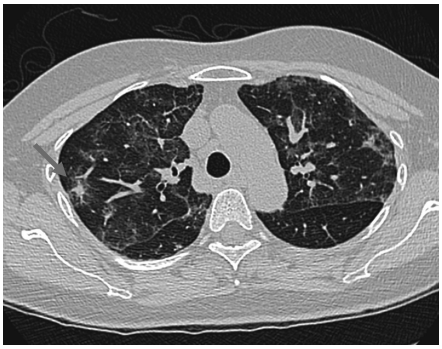
Appendix A

Figures

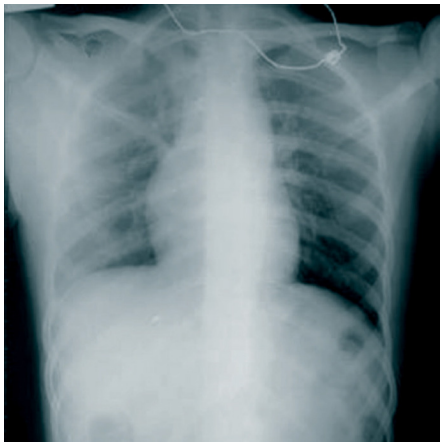
(a) Infant



(b) Side View



(c) Top View (CT Scan)



(d) Inaccurately Colored

Figure A.1: Examples from the Training Set

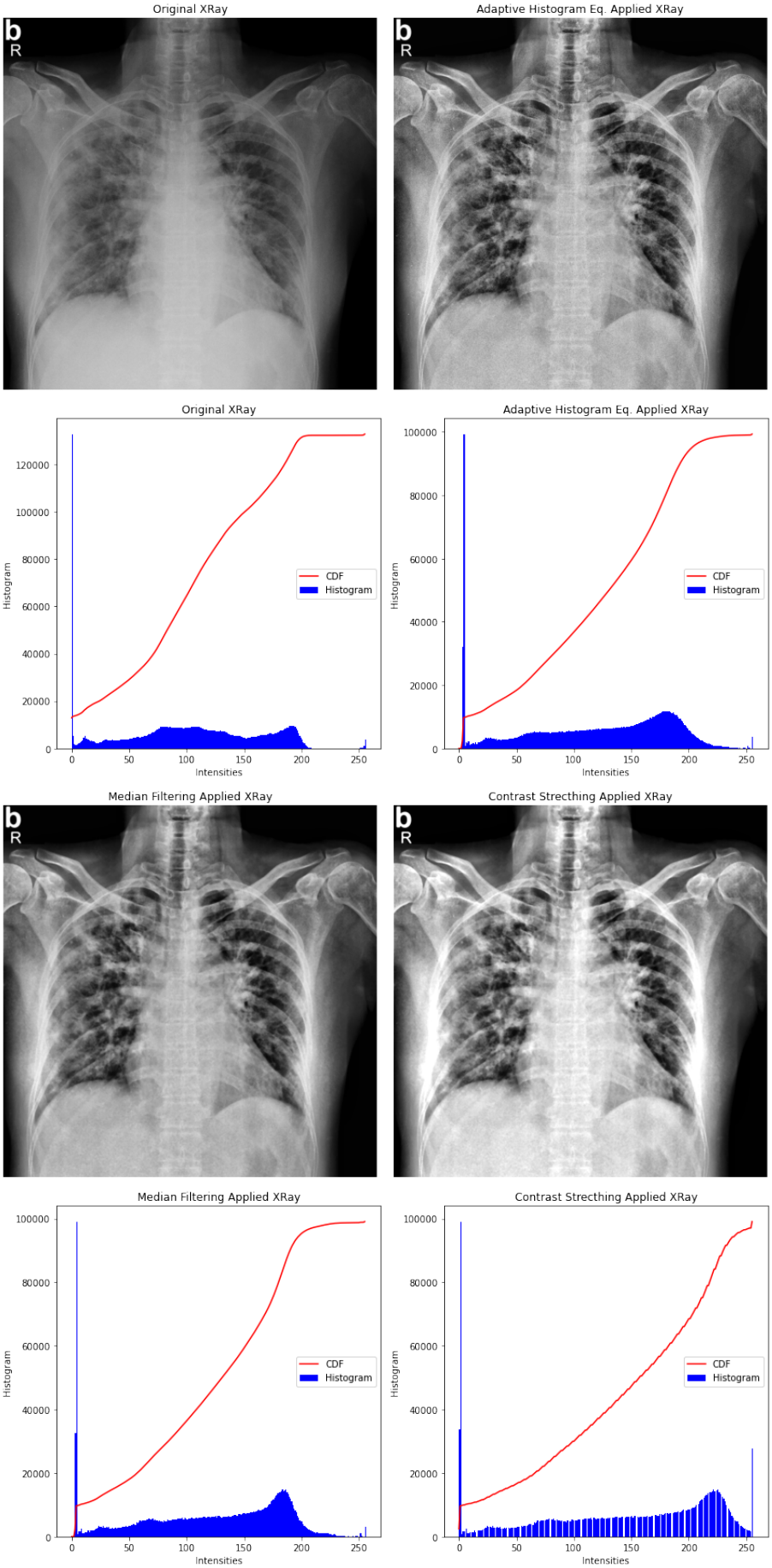
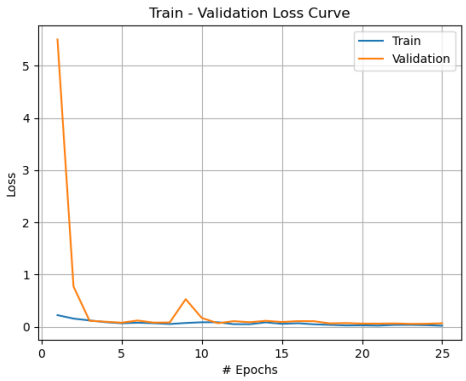
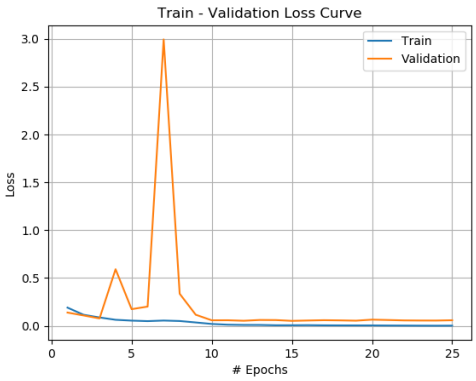


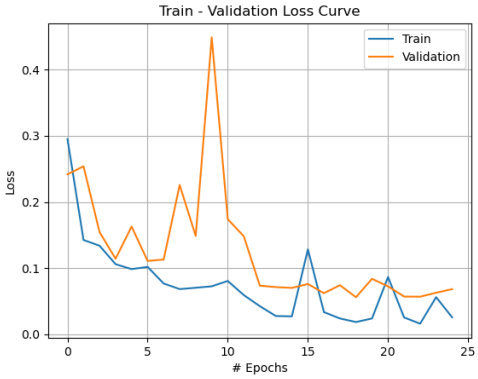
Figure A.2: Chest X-Ray Normalization



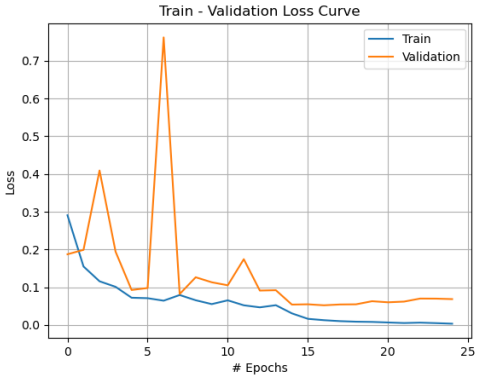
(a) DenseNet-121 Binary w/o Oversample Baseline



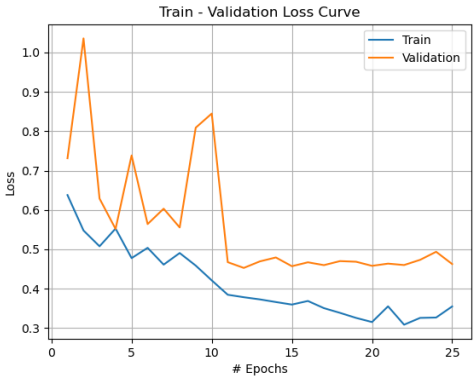
(b) DenseNet-121 Binary w/ Oversample Baseline



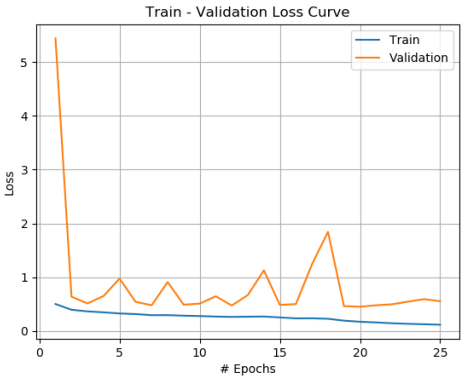
(c) ResNet-101 Binary w/o Oversample Baseline



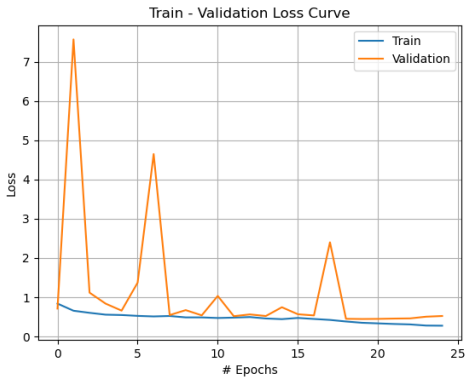
(d) ResNet-101 Binary w/ Oversample Baseline



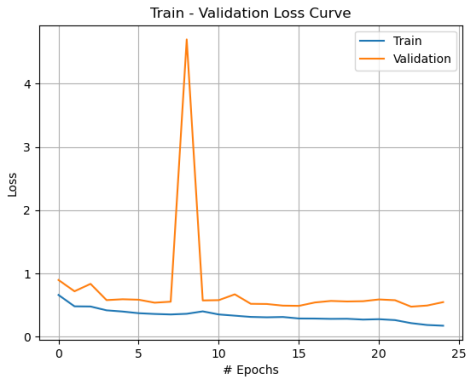
(e) DenseNet-121 Multi-to-Binary w/o Oversample Baseline



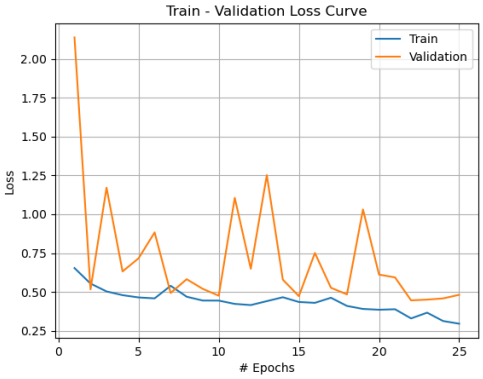
(f) DenseNet-121 Multi-to-Binary w/ Oversample Baseline



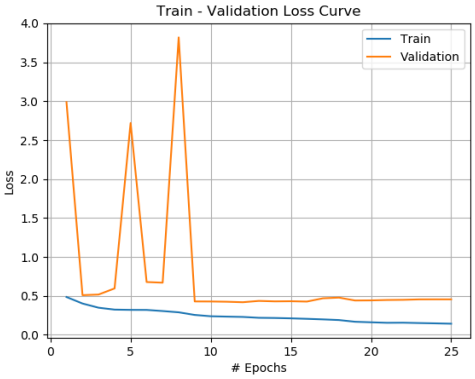
(g) ResNet-101 Multi-to-Binary w/o Oversample Baseline



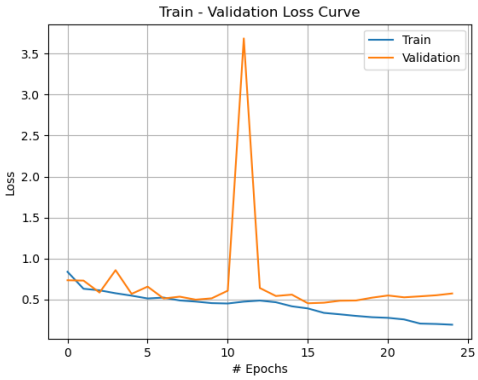
(h) ResNet-101 Multi-to-Binary w/ Oversample Baseline



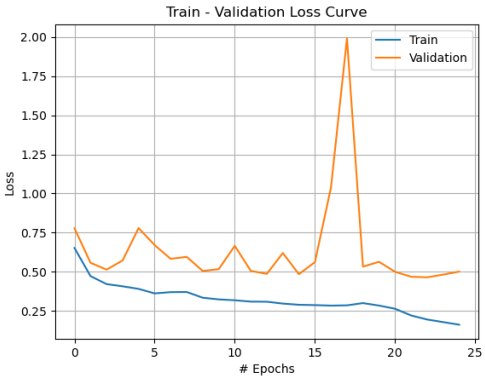
(i) DenseNet-121 Multi w/o Oversample Baseline



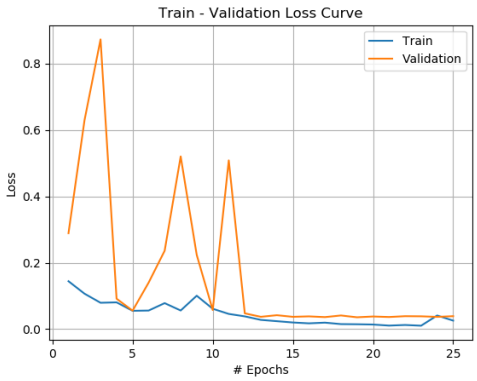
(j) DenseNet-121 Multi w/ Oversample Baseline



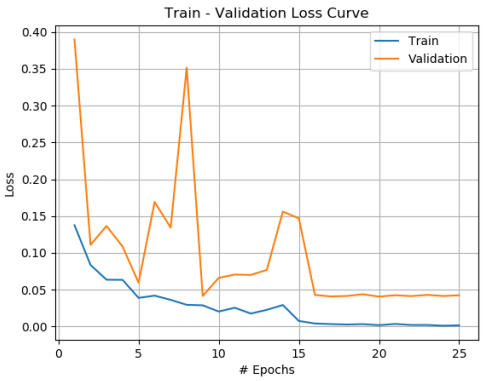
(k) ResNet-101 Multi w/o Oversample Baseline



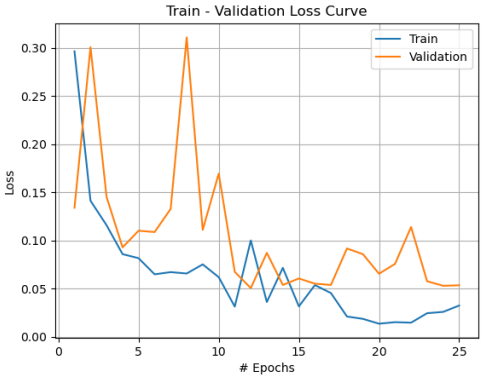
(l) ResNet-101 Multi w/ Oversample Baseline



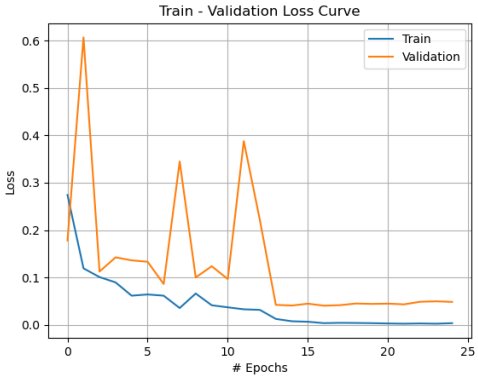
(m) DenseNet-121 Binary w/o Oversample + CXR Norm



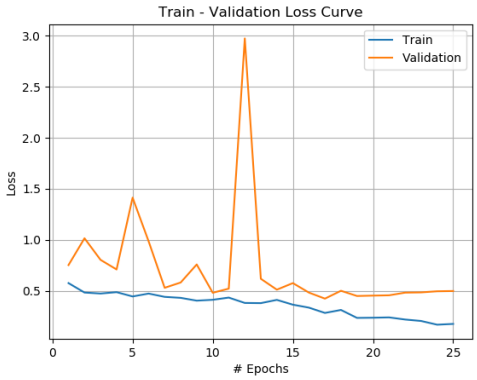
(n) DenseNet-121 Binary w/ Oversample + CXR Norm



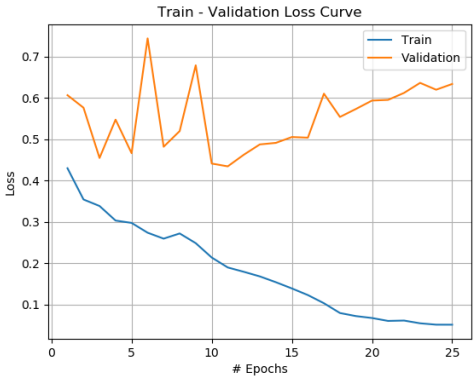
(o) ResNet-101 Binary w/o Oversample + CXR Norm



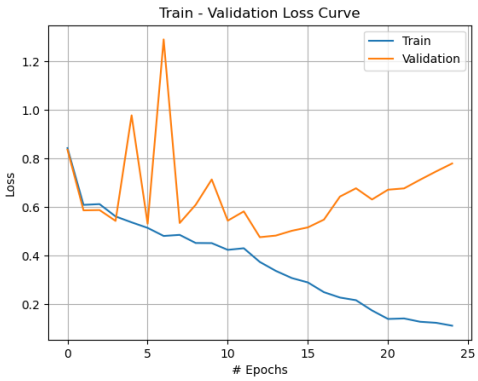
(p) ResNet-101 Binary w/ Oversample + CXR Norm



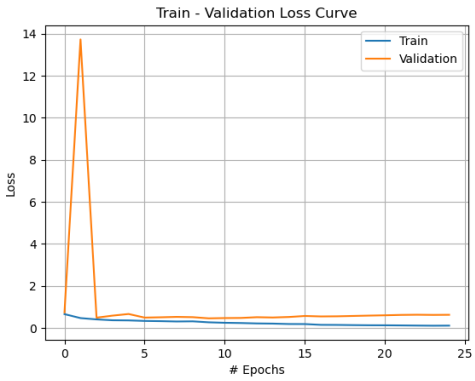
(q) DenseNet-121 Multi-to-Binary w/o Oversample + CXR Norm



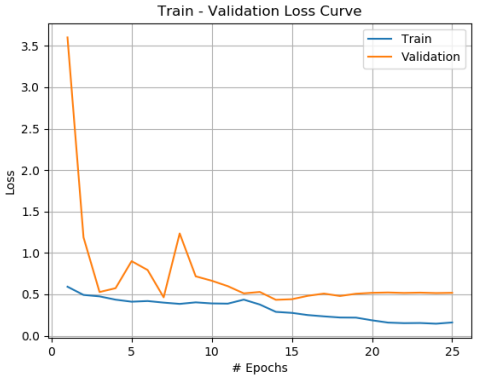
(r) DenseNet-121 Multi-to-Binary w/ Oversample + CXR Norm



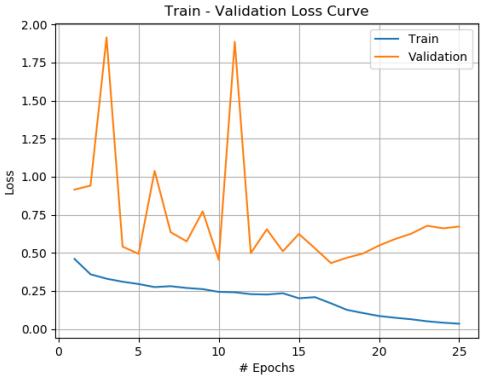
(s) ResNet-101 Multi-to-Binary w/o Oversample + CXR Norm



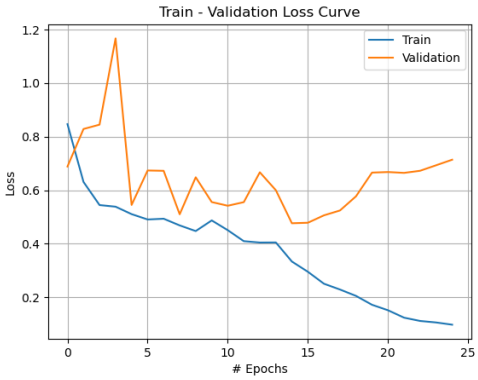
(t) ResNet-101 Multi-to-Binary w/ Oversample + CXR Norm



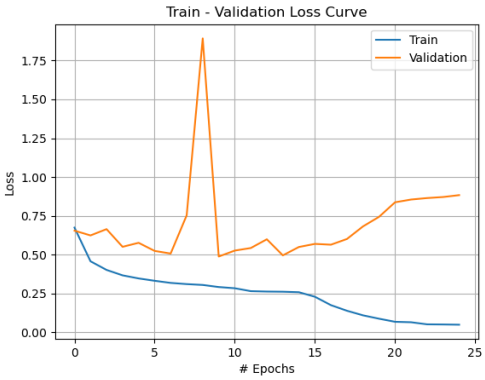
(u) DenseNet-121 Multi w/o Oversample + + CXR Norm



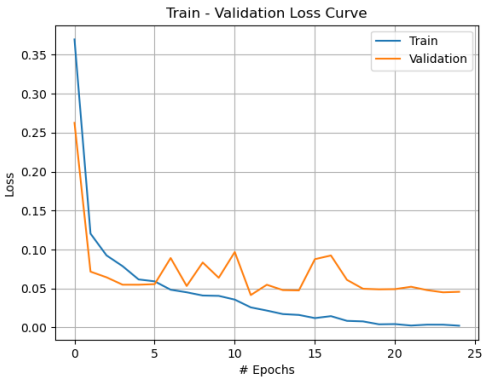
(v) DenseNet-121 Multi w/ Oversample + CXR Norm



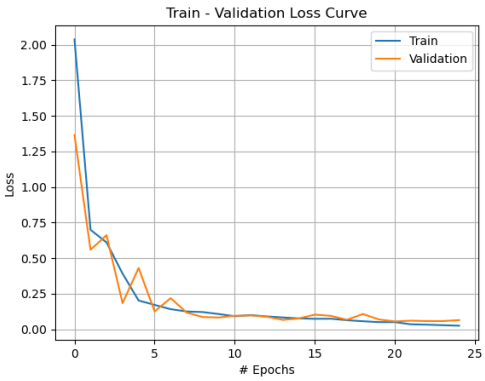
(w) ResNet-101 Multi w/o Oversample + CXR Norm



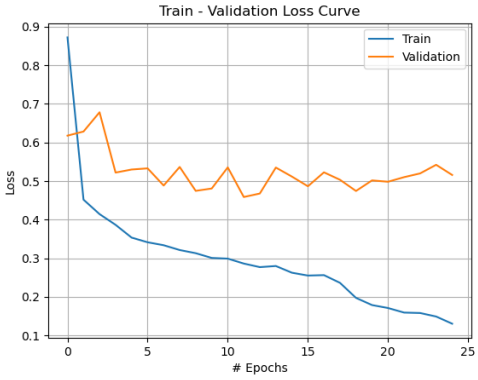
(x) ResNet-101 Multi w/ Oversample + CXR Norm



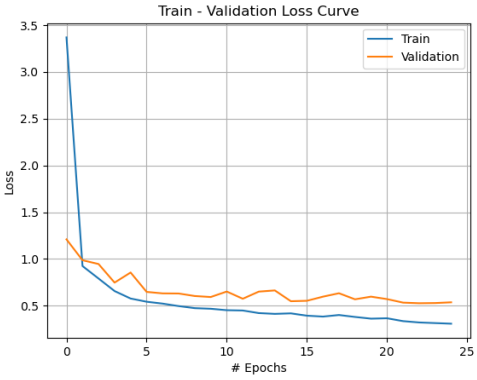
(y) DenseNet-121 Binary w/ Oversample + CXR Norm + SAM



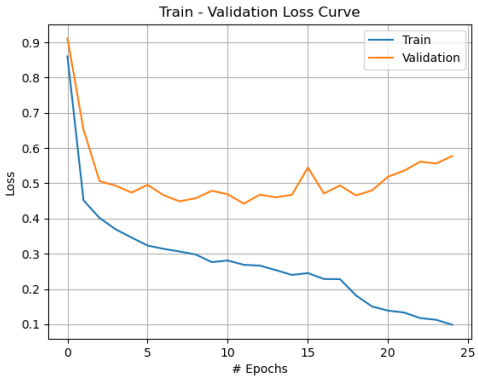
(z) ResNet-101 Binary w/ Oversample + CXR Norm + SAM



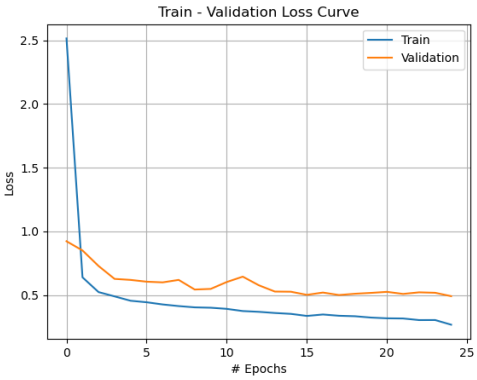
(aa) DenseNet-121 Multi-to-Binary w/ Oversample + CXR Norm + SAM



(ab) ResNet-101 Multi-to-Binary w/ Oversample + CXR Norm + SAM

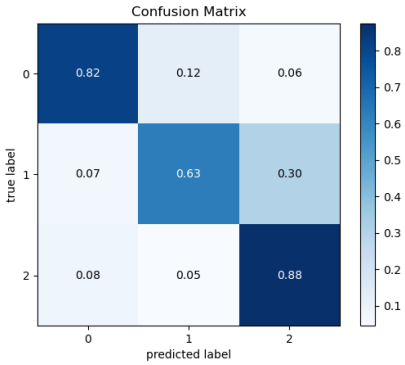


(ac) DenseNet-121 Multi w/ Oversample + CXR Norm + SAM

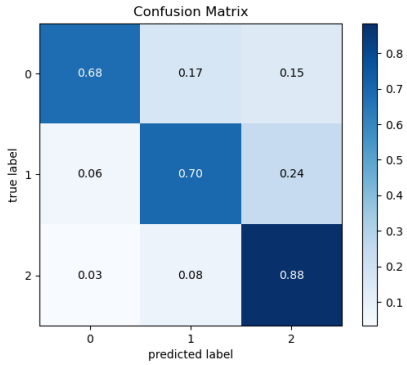


(ad) ResNet-101 Multi w/ Oversample + CXR Norm + SAM

Figure A.3: Loss Curves of All Models up-to 25$^{\text{th}}$ Epoch



(a) Confusion Matrix of DenseNet-121 Multi Class



(b) Confusion Matrix of ResNet-101 Multi Class

Figure A.4: Confusion Matrices of Models Trained with Oversampling, CXR Norm and SAM