COMP540 Xray COVID Detection

Kaichun Luo, Max Yu

Department of Computer Science, Rice University

Problem

COVID-19 has been prevailing since the beginning of 2020. The contagious virus can be diagnosed via polymerise chain reactions. However, it can take a few hours and sometimes days before the molecular test results are back. Chest radiography serves as another method of diagnosis, which enjoys its advantages including rapid triaging, availability and accessibility, and portability.[1] However, in the Chest X-rays (CXR), COVID-19 looks very similar to other viral and bacterial pneumonia, so quick and confident diagnoses are still very difficult.

Datasets

We used a total of 7 datasets to train our model.

First are 4 datasets used to pretrain the encoder/decoders used in this project: Pneumothorax,[2] Pneumonia,[3] CheXpert,[4] and Chest14.[5]

We also used 3 datasets to train our actual model.

1. SIIM-Covid19-detection: The SIIM-Covid19-detection dataset[6] comprises 6,334 de-identified chest scans in DICOM format for training.
2. COVIDx CXR-2: The COVIDx CXR-2 dataset[7] is comprised of a total of 13,975 chest X-ray images across 13,870 patient cases.
3. TCH Dataset: This dataset consists of pediatric X-ray images from Texas Children's Hospital.

Encoder-Decoder Architecture

Our model is built upon the code base of https://github.com/dungnb1333/SIIM-COVID19-Detectiondungnb1333. As shown in Figure 2, an encoder first extracts features from the input image, then the encoder output goes through several layers of linear transforms, activation, and dropout, and finally a classification head (a dense layer) outputs the classification predictions. On the other hand, a decoder takes the encoder output, followed by a segmentation head which outputs a segmentation mask.

The model is pretrained on CheXpert,[4] Chest14[5] and RSNA pneumonia[3] datasets, as suggested by.[8]

Encoder	Decoder
seresnet152d[9]	Unet[10]
efficientnet-b5[11]	DeepLabV3-Plus[12]
efficientnet-b6[11]	Linknet[13]
efficientnet-b7[11]	Unet++[14]

Fig. 1: Four Encoder-Decoder Models

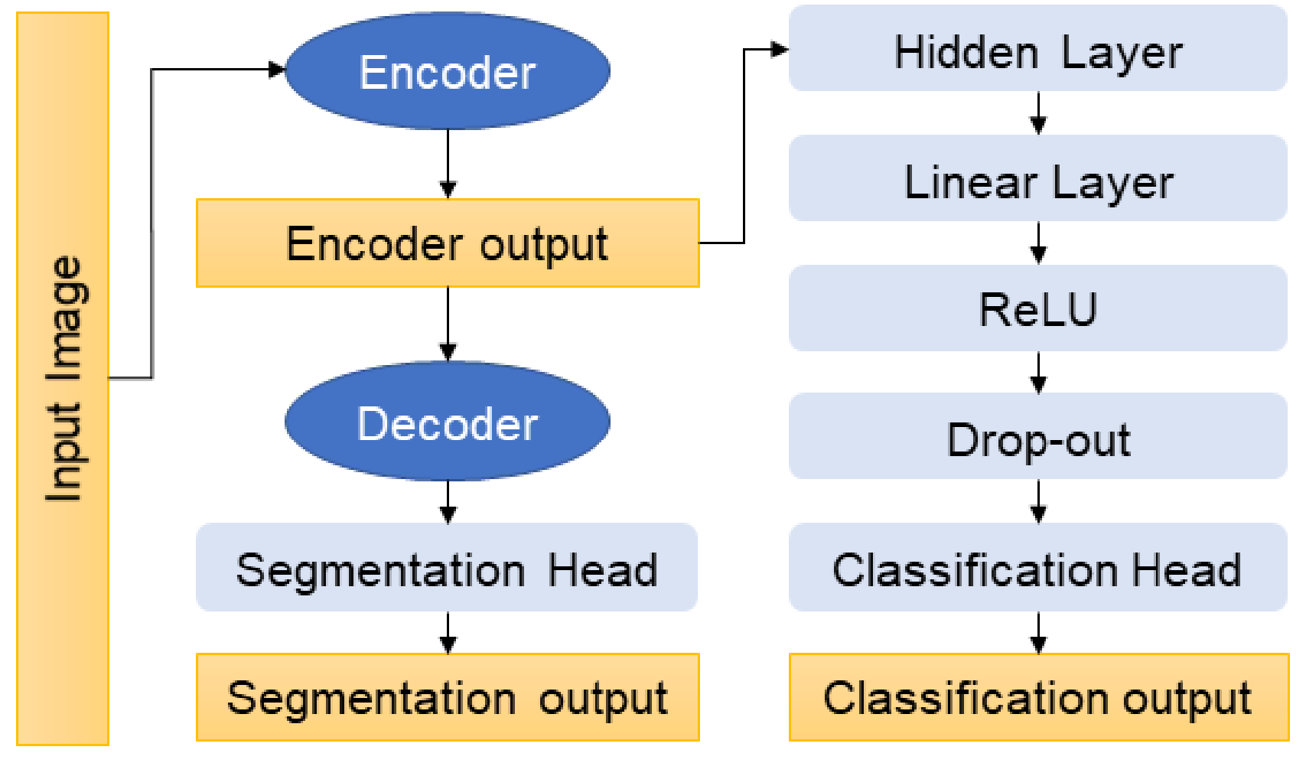


Fig. 2: Architecture of our models

Sementation Masks and Psuedo Segments

In the later stages, we realized that we could use our decoders to generate pseudo segmentations on the TCH train data, which originally didn't have segmentations labeled. We trained the models on TCH data for both segmentation and classification.

Model and Training

Due to large memory limitations of the pretrained models, our training was ran on a University of Houston's server with GPU NVIDIA Tesla V100 32GB.

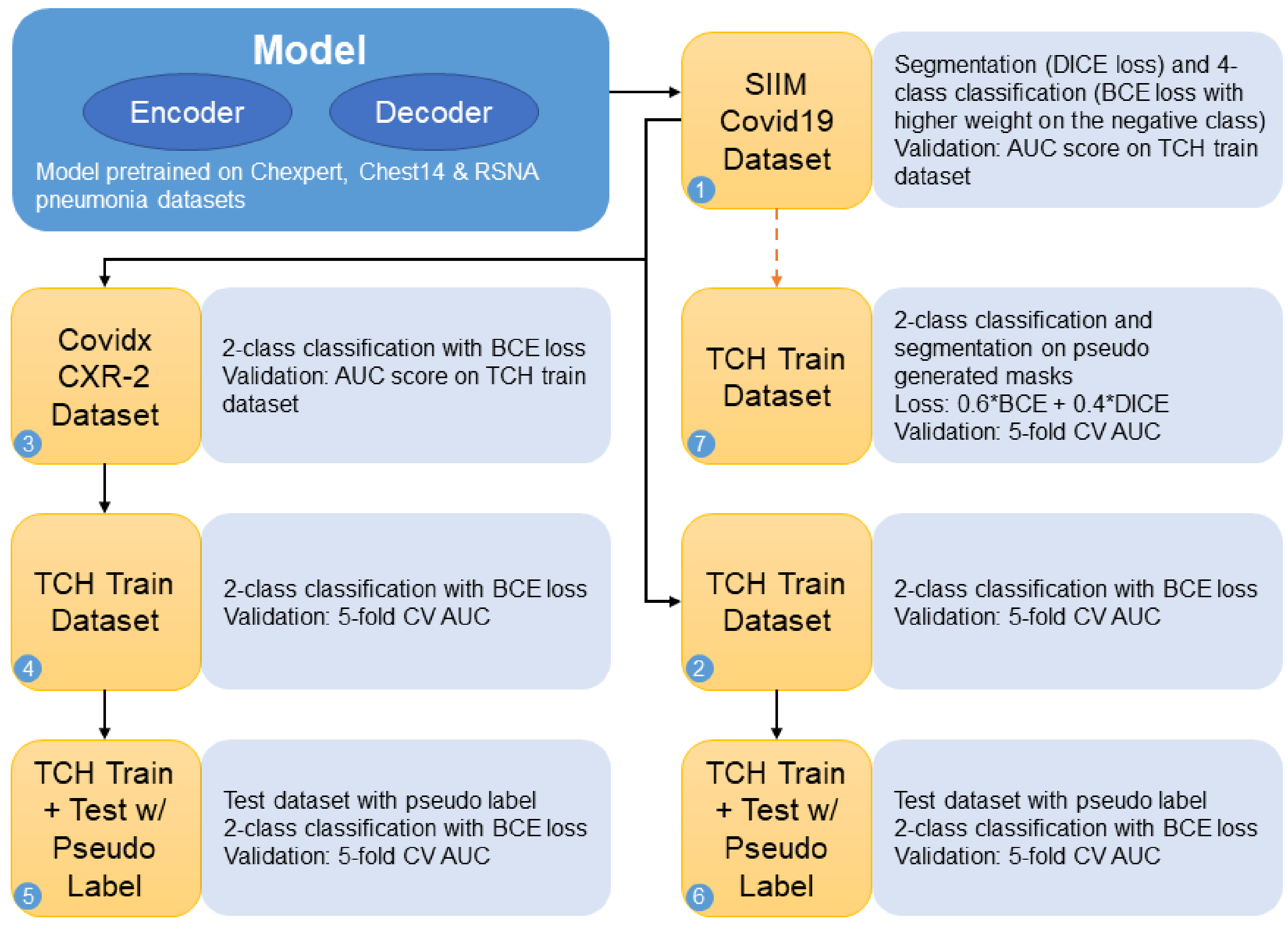
We train the models on the SIIM Covid19 Dataset[6] (Step 1) for both segmentation and classification. We use a combination of Dice loss and weighted Binary Cross Entropy loss for this training step, since there are 4 classes for this dataset.

To prevent overfitting, we use the TCH dataset for validation. For each encoder-decoder pair, weights of the epoch with the highest AUC score on TCH train data are kept.

We then continue training the models with the TCH train data. Since this dataset has only binary labels, we use BCE loss for training and ignore the segmentation output. For validation, we split the train data into 5 folds. We train each model 5 times, each time using one fold for validation and the other four folds are training and keep the epoch with the highest AUC score on validation data.

In order to build more robust models, we have also tried to pretrain our model on the Covidx CXR-2 Dataset[7] (Step 3), using BCE loss for binary classification and validating the model with AUC score on TCH train data. Thereafter, we train the models obtained from Step 3 on TCH Train Dataset (Step 4), with exactly the same settings as Step 2.

At this stage, we have several highly scored submissions on Kaggle. We then generated pseudo labels using the prediction which has the highest AUC score on the public test. Then, we trained the models on the pseudo labelled data using the same 5 fold validation training method as earlier. This process is done for both Covidx-CXR-2-pretrained and non-Covidx-CXR-2-pretrained models (Step 5 and Step 6 respectively).



Model pretrained on Chexpert, Chest14 & RSNA pneumonia datasets

SIIM Covid19 Dataset: Segmentation (DICE loss) and 4-class classification (BCE loss with higher weight on the negative class) Validation: AUC score on TCH train dataset

Covidx CXR-2 Dataset: 2-class classification with BCE loss Validation: AUC score on TCH train dataset

TCH Train Dataset: 2-class classification and segmentation on pseudo generated masks Loss: 0.6*BCE + 0.4*DICE Validation: 5-fold CV AUC

TCH Train Dataset: 2-class classification with BCE loss Validation: 5-fold CV AUC

TCH Train Dataset: 2-class classification with BCE loss Validation: 5-fold CV AUC

TCH Train + Test w/ Pseudo Label: Test dataset with pseudo label 2-class classification with BCE loss Validation: 5-fold CV AUC

TCH Train + Test w/ Pseudo Label: Test dataset with pseudo label 2-class classification with BCE loss Validation: 5-fold CV AUC

Fig. 3: Flowchart of our training process

Predictions

Previous work[8] suggests that test time augmentations (TTAs) would improve prediction results. We have trained a YoloV5 based lung cropper using the SIIM-Covid19-detection dataset.[6]

To classify a CXR image in the TCH test data, we use the model to predict on the original image and the cropped lung image (both resized to the input size required by the model). We pass the two inputs to the 5 instances of each model generated from 5-fold CV and then averages the ten outputs to obtain the prediction. With the predictions by the 4 encoder-decoder models, we use ensemble by calculating the weighted average probability

Results

The following table shows the progression of our model, starting with the pretrained model training on TCH dataset, to more complex methods like pseudo labels and ensembles. We were doing better and better in terms of the public dataset, and for our final submission we chose our best public scoring model, ending at a public score of 0.94070 and private score of 0.91609.

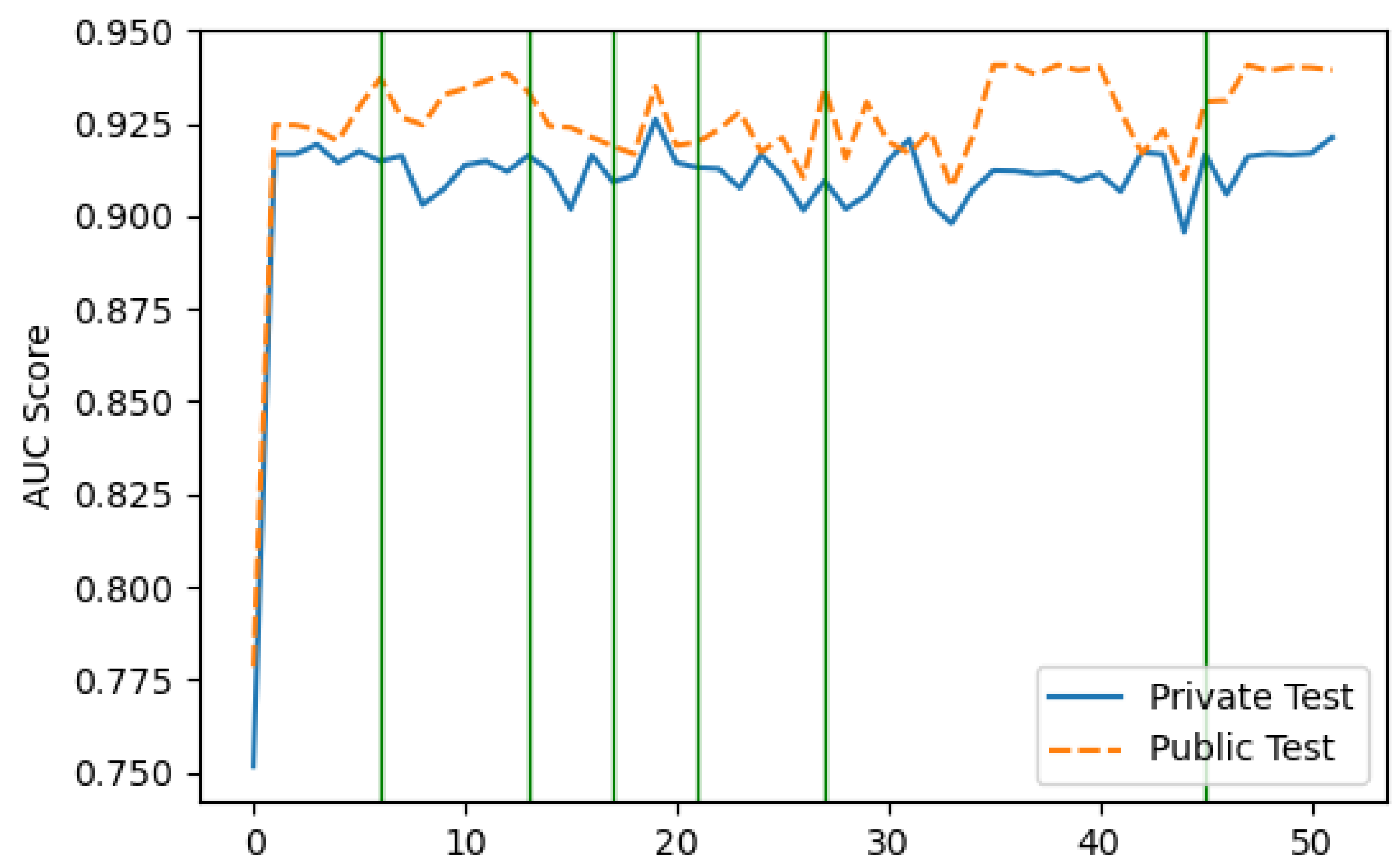


Fig. 4: Kaggle Results

And the final ensemble weights we ended up using are

$$SeR\left<2\right> \times 0.2 + EB7\left<2\right> \times 0.2 + SeR\left<5\right> \times 0.3 + SeR\left<7\right> \times 0.3$$

Discussion and Afterthoughts

The reason why we choose to use sigmoid over softmax is that at the early stage of our training process, sigmoid yields better results on the public test data.

This, however, is one lesson we learn from this project. Throughout the project, we focused too much on improving the public leaderboard score, which doesn't necessarily mean good overall performance. The private leaderboard results has demonstrated this. Our early models achieve an unexpectedly high score on the public leaderboard, which could be sheer luck, giving us a misdirection. We do not have the time and capacity to train with both activation function, and now we recognize that use of softmax may lead to a better result.

References

[1] Linda Wang, Zhong Qiu Lin, and Alexander Wong. Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):19549, Nov 2020.

[2] See. pneumothorax. https://www.kaggle.com/seesee/siim-train-test, 2019.

[3] Radiological Society of North America. pneumonia. https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data, 2018.

[4] Stanford ML Group. Chexpert. https://stanfordmlgroup.github.io/competitions/chexpert/, 2017.

[5] National Institutes of Health. Chestx-ray14. https://nihcc.app.box.com/v/ChestXray-NIHCC, 2017.

[6] Society for Imaging Informatics in Medicine. Siim-covid19-detection. https://www.kaggle.com/c/siim-covid19-detection/data, 2021.

[7] Linda Wang, Zhong Qiu Lin, and Alexander Wong. Covidx cxr-2. https://www.kaggle.com/datasets/andyczhao/covidx-cxr2, 2020.

[8] Nguyen Ba Dung. Siim-covid19-detection. https://github.com/dungnb1333/SIIM-COVID19-Detection, 2021.