CAP 5516 Medical Image Computing Spring 2022 Assignment 1: Pneumonia Classification from Chest X-Ray

Kyle Beggs
Department of Mechanical and Aerospace Engineering
University of Central Florida
kbeggs07@knights.ucf.edu

Abstract

DOnt forget to fill this out!

1. Problem Definition

The most common diagnostic for patients that are suspected to have pneumonia is a chest x-ray. After the x-ray is taken, a radiologist looks at the image and determines if the patient is positive or negative for pneumonia. Automating this process would save much time for physicians on a routine task. The goal of this project is to build a deep learning model to detect pneumonia in pediatric chest x-rays.

2. Methods

Due to limited compute resource, the ResNet18 network (Figure 1) is chosen to perform the classification task as it has few learnable parameters compared to top performing networks, yet only suffers little in performance. The network is trained both with random initialization of weights and using pretrained weights found from training the network on ImageNet.

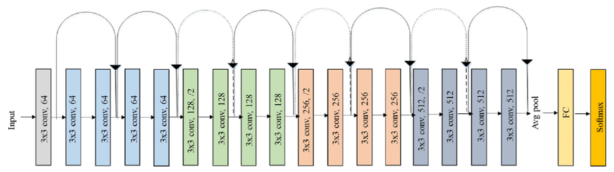

Figure 1. ResNet18 Architecture.

The dataset is taken from Kaggle (https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia) containing 5856 images in total. There are 5216 in the training set, 16 in the validation set, and 624 in the test set. There are only 2 classes - 'normal' which is pneumonia negative and 'pneumonia' which is positive for the condition. the classes are imbalanced, as the 'normal' class makes up 25.7% of the training set while the 'pneumonia' class makes up the rest at 74.3%. Class imbalance is addressed through weighting the loss function according to the inverse of the percent of the datset that class represents. Data augmentation is performed by randomly flipping about the x and y axis, both with 50% probability as well as randomly zooming between a range of 0.9-1.1 with a 50% probablity. An example of the original image and its transform, as well as their histograms are shown in Figure 2.

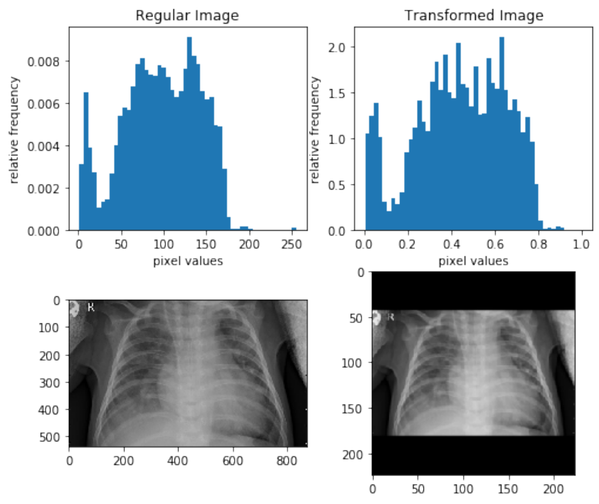

Figure 2. Histogram and display of original image (left) and transformed image (right).

The images are preprocessed by padding with 0's such that they are square as the aspect ratios of the images in the datasets are not constant. Then the image is resized to 224x224 for computational efficiency reasons, as well as this was the original image size ResNet was used for. The image is ensured to be 1 channel (greyscale) and lastly the image is scaled according to its intensity.

The batch size is set to 8 due to GPU memory limitations. The learning rate is initially set to 0.00005 and is set to decay according to a reduction of 30% of the current value every 10 epochs. The model is trained for 100 epochs every time with no early stopping. The weights are saved from the last epoch of each run for evaluation of model performance using the test set.

3. Results

The model is incrementally trained - meaning we begin with a most basic model and keep adding complexities that in theory should increase the performance of the model. Four models are completed, each including everything of the model before:

1. Basic model with random weight initialization.

2. Data augmentation.

3. Pretrained with ImageNet.

4. Class imbalance addressed through loss weighting.

A bonus is also trained where the model is pretrained, balanced, with no augmentation. The training and validation loss is shown in Figures 3 and 4 and the validation accuracy is shown in Figure 5.

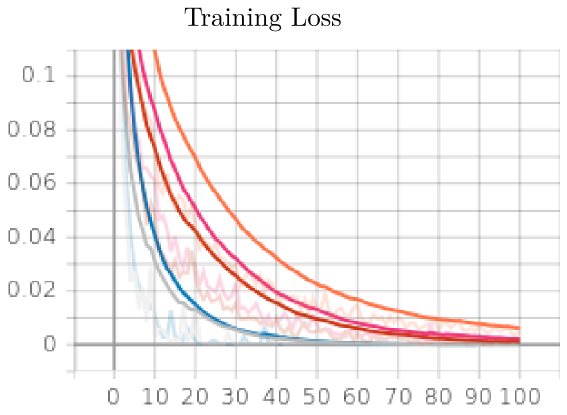

Figure 3. Training loss curves for training set. Curve colors: blue is random initialization with no augmentation, orange is random initialization with augmentation, red is pretrained with augmentation, pink is pretrained with augmentation that is balanced, and grey is pretrained with balancing and no augmentation.

The test set is evaluated as the results are shown in Tables 1 and 2.

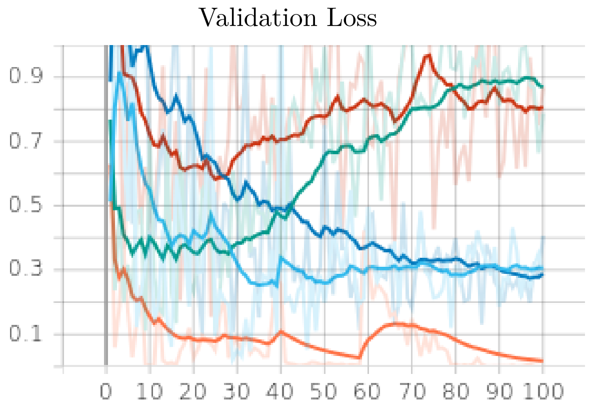

Figure 4. Validation loss curves for training set. Curve colors: red is random initialization with no augmentation, blue is random initialization with augmentation, light blue is pretrained with augmentation, green is pretrained with augmentation that is balanced, and orange is pretrained with balancing and no augmentation.

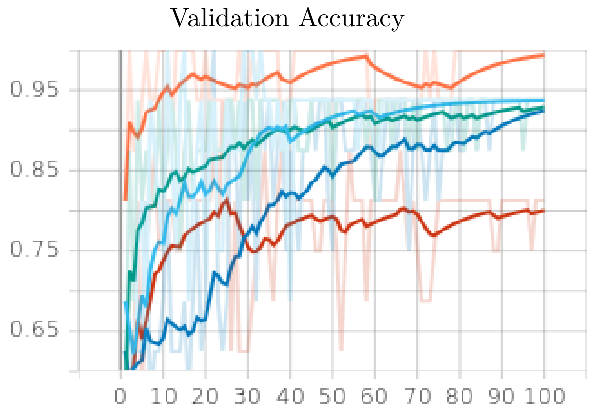

Figure 5. Accuracy curves for validation set. Curve colors: red is random initialization with no augmentation, blue is random initialization with augmentation, light blue is pretrained with augmentation, green is pretrained with augmentation that is balanced, and orange is pretrained with balancing and no augmentation.

4. Discussion

AS shown in Tables 1 and 2, the best performing model (according the F1 score) was pretrained on ImageNet, and class balanced. Conventional wisdom expects these settings with the addition of data augmentation to be the best, however it performs second best implying augmentation to be detrimental to the performance of the model. There is suspicion that augmentation was too aggressive as it included flipping on both axis and zooming. After re-examination of the original images, they are all oriented correctly, so flipping upon axis does not make sense. Slight zoom in or

Model Settings	Accuracy (Overall)	F1
Random	0.8685	0.7297
Random w/ Aug.	0.8741	0.7602
Pre. w/ Aug.	0.8869	0.7752
Pre. w/ Aug. & Bal.	0.8866	0.7853
Pre. & Bal.	0.8974	0.8145

Table 1. Results of test set showing overall accuracy for both classes and F1 score. 'Pre.' is for pretrained with ImageNet, 'Aug.' is for data augmentation, and 'Bal.' is when class imbalance was addressed.

Model Settings	Accuracy (Pneumonia)	Accuracy (Normal)
Random	0.7466	0.9903
Random w/ Aug.	0.7653	0.9829
Pre. w/ Aug.	0.7738	1.0000
Pre. w/ Aug. & Bal.	0.7811	0.9921
Pre. & Bal.	0.8021	0.9928

Table 2. Results of test set showing accuracy for each class. 'Pre.' is for pretrained with ImageNet, 'Aug.' is for data augmentation, and 'Bal.' is when class imbalance was addressed.

out is evident throughout the dataset. It is suggested to slightly rotate the images as well. It is not clear why misclassifications happen when looking at images only (shown in Figure 6) and the use of TorchCAM is suggested.

The code can be found HERE.

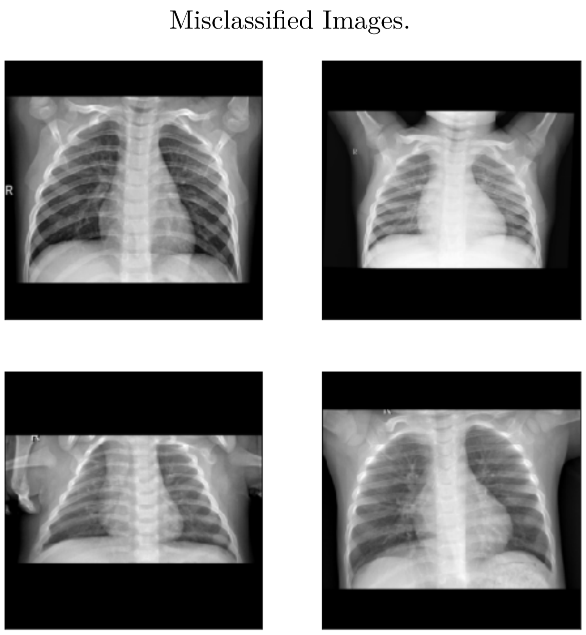

Figure 6. Transformed images that are misclassified by the best-performing model.