Classification of X-ray images using different models

Choi, Moon-ki **Choi, Won Joon** **Kim, Hyunwoo** **Lee, Garim**

1 Data pre-processing

The chest X-ray images contained three parts: train, val, and test. The labels for the X-ray images were either Normal or Pneumonia. Since the dataset was not balanced we wanted to explore the difference between models we obtain from ANN, CNN, and transfer learning by training on both balanced and unbalanced dataset. Originally, the dataset contained 1341 normal and 3875 pneumonia images for the train set, 8 normal and 8 pneumonia images for the validation set, and 234 normal and 390 pneumonia images for the test set. Since there were only eight images for each label in validation set, we moved some of the images from train set to make reasonable validation in each training phase. The following is the number of images in each phase in balanced and unbalanced dataset, respectively:

Number of images (balanced data)		
	Normal	Pneumonia
Train	1241	1241
Val	108	108
Test	234	234

Number of images (unbalanced data)		
	Normal	Pneumonia
Train	1239	3773
Val	110	110
Test	234	390

The following is different data-preprocessing steps taken for different models:

- For SVM, before applying image data to the SVM, the images were converted to a grayscale value and received input, and the images were integrated into an image size of 100x100 pixels. The images were then normalized by 255 and its 3-dimension changed to 2-dimensions.
- For ANN, CNN, and transfer learning, we prepared our data using PyTorch's data transformer.
- For ANN, the data was normalized using a mean of [0.5, 0.5] to use grayscale images instead of RGM images, considering the time and computation complexity of ANN. The batch size was set to 1.
- For CNN and transfer learning, we normalized our data using the mean of [0.485, 0.456, 0.406] and standard deviation of [0.229, 0.224, 0.225] since there are three channels for RGB images. The batch size was set to 16 both CNN and transfer learning.

The following images are original image vs. transformed image of the same picture:

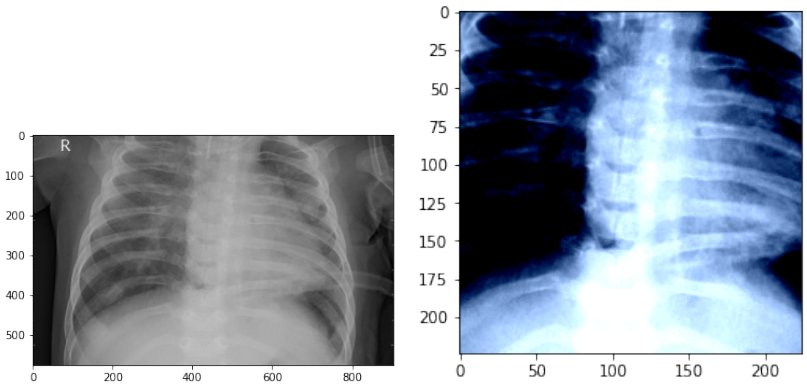

Figure 1: Original x-ray image vs transformed x-ray

2 Training and testing with different models

2.1 SVM

Unlike Pytorch, which can use GPUs, SVM which uses normal CPU image processing takes too long. There are thousands of image files in our data, which took about five or six hours. Above all, the time problem of this image processing is the priority. The complexity arising from nonlinear SVM image mapping; lower-order polynomials fail to represent complex datasets and the higher-order polynomials add too many properties to slow processing. But, the mapping problem can be solved using kernel tricks, such as, Polynomial, Sigmoid, and Gaussian Radial Basis Function (RBF) and the most efficient one of them will apply to the SVM.
Overall, image mining with SVM is successfully projected, but, for a cleaner project code implementation, SVM will be implemented using the Pytorch data loader used by other members. Using SVM kernel functions and implementing loaders created through image data processing for SVM is more challenging than expected.

2.2 ANN

Artificial neural network (ANN) model was created using one input layer, four hidden layers, and one output layer. The number of nodes for each hidden layer was set to 256, 128, 128, and 64, respectively. The number of hidden nodes was set not too large based on the previous literature, which suggests that a large number of hidden nodes may degrade the generalization ability of a network because large hidden nodes may have spurious connections. Responsive activation functions (RELU) were used for activation. After the ANN model was created, the model was trained using training data and validated using validation data with 15 epochs. The model with the highest validation accuracy was used to predict the output for test data. All these experiments were conducted separately for a balanced and unbalanced data set. The below table shows the test accuracy of the balanced and unbalanced data. For future steps, it would be beneficial to try varying sizes of hidden nodes to see if accuracy can be enhanced.

Test Accuracy	
Balanced	Unbalanced
0.759	0.800

2.3 CNN

Convolutional neural network (CNN) model is composed of combination of two 2D convolution layers (input dimension = 3, output dimension = 3, kernel size = 4 and 3) and 2D max-pooling (kerneal size = 4). Then, a linear transformation with the RELU as an activation function result in 2 final outputs. The number of epoch is 5 and the model is evaluated and saved after computing the accuracy through the validation data at the end of each epoch. After training, the test accuracy of model for balanced data set is 77.05 % and accuracy of model for unbalanced set is 80.21%, as shown in the following table.

Test Accuracy	
Balanced	Unbalanced
0.771	0.802

2.4 Transfer learning(GoogleNet, ResNet)

Transfer learning part includes two pre-trained models from PyTorch: GoogleNet and ResNet. The details of GoogleNet and ResNet will be included in the final report of the group project due to the limited length of the interim report. The progress made so far is training on the balanced and unbalanced dataset using GoogleNet and ResNet and choosing the best model from the validation set. We chose 15 epochs for the training phase of pre-trained models. More experiments will be done by changing epochs, hyperparameters and the dimension of images. The test accuracy of the models trained on balanced and unbalanced set are as follows: The following table shows test accuracy of four different scenarios.

Test Accuracy		
	Balanced	Unbalanced
GoogleNet	0.470	0.385
ResNet	0.821	0.886