Detection Of Chest Opacities From CT Scan Images Using Deep Learning Models

Epuret Moses Obuya

2100702291

2021/HD05/2291

Background

Chest opacities are a major health problem worldwide; failure of early detection and treatment may lead to significant morbidity and mortality. In addition, chest diseases are a great burden to the individual suffering, their attendants and society at large. Chest diseases are among the ten leading causes of death, ranking second in Africa. Chest X-ray is the first line procedure to assess chest conditions but the interpretation of radiologic signs such as vascular opacity redistribution and interstitial oedema are often questionable and subjective. Moreover, X-ray is 2 dimensional hence one may not accurately tell the nature of these opacities and localization of the same is not definitive; even with established guidelines for interpretation, chest X-ray has demonstrated to be an insensitive method with relatively low accuracy. This delays the establishment of the cause of chest opacity hence mismanagement of the patient leads to more costs for hospitalisation and even death. CT scan has been determined as the gold standard to characterise, localise and identify the nature of opacities. However, it is expensive, not always available, uses ionising radiation and transferring critically ill patients to the CT room is complicated.

In this project, we use deep learning to train a model, based on Ultrasound images, that helps in the diagnosis and management of chest opacities while overcoming the limitations of X-ray and CT scan-based methods. The method has the potential to accurately localize opacities, differentiate between opacities and suggest appropriate further investigations. Ultrasound is easily available, uses no ionising radiation, and is relatively cheaper and portable.

Methodology

A. Dataset

The dataset used in this comprised 715 images with 364 collected from patients who had no chest opacities and 351 from patients with chest opacities. It was organised in such a way that the normal images were stored in a separate folder and the sick images were also in a separate folder. This dataset was split into training and validation sets in a ratio of 80 to 20 of the total dataset.

B. Data Preprocessing

Given the training and validation dataset being small and imbalanced and training the model on this could lead to overfitting[1] thus we preprocess the data by augmenting it. Augmenting helps increase the number of images as different techniques are employed to synthesise data from the limited dataset[2]. Techniques such as:

- Rescale which scales the images to a value provided.
- Flip which flips the image vertically and horizontally.
- Rotation which rotates the image to a specified angle between 0 and 360 degrees.

After the different augmentation parameters are performed on the images, they are considered new images thus increasing the dataset size. Augmentation helps expand the training dataset in order to improve the performance and ability of the model to generalise. This augmentation is only performed on the training set while the validation and test data are only rescaled.

C. Deep Learning Models

The VGG16 pre-trained model is used to train the dataset and the include_top parameter of the model is set to false because it consists of classifications already based on the imagenet dataset. Our goal is to train the model on features in our dataset thus the trainable parameter is set to false. While training this model, it's set to not trainable as it has its own parameters but our goal is to train it with parameters from our dataset. Thus to make the model specific to our problem, we add the *GlobalAveragePooling2D* layer which is adaptive and doesn't take a fixed size and offers a better representation of vectors. We also remove the *Flatten* layer since it would result in a larger Dense layer afterwards, which is more expensive and may result in worse overfitting. The training attribute for our model was set as *False* as our goal was to infer from our dataset provided instead of randomly generating from the Fully Convolutional (FC) layers of the network. The goal was for our model to reach the level of the pre-trained model quickly.
Since we compile our model with parameters such as optimisation using the Adam [3] optimiser, the loss function of categorical_crossentropy as the labels are 2 and non-integer.

Model Check Points

We save the weights of our model for use in finetuning it and create a checkpoint to monitor the model weights. The *ModelCheckpoint* function is the callback function that helps us save the model while fitting it. While saving the model, we do save the weights upon improvement in the validation accuracy of the model and only save the best weights.

Early Stopping

While the epochs are running during model fitting, we also made sure the fitting process stops early when the validation loss is minimum and the learning rate is monitored. These callbacks help us not to waste computing resources in case the chosen metric isn't improving during model fitting.

Learning Rate

The rate at which a model converges is influenced by a learning rate [4] and we used a fixed learning rate of 0.001 and also implemented *ReduceLROnPlateau* which reduces the learning rate when certain metrics like validation loss have stopped improving. When model performance stops improving, reduce on the plateau decreases the learning rate of the model enabling the model to take smaller steps towards convergence [1].

Initial Model Training Results

As the model is trained, it can be observed that the training accuracy and validation increase as the number of epochs increases while the training loss and validation loss also decrease as the epochs increase. The model stops learning at a certain number of epochs because of early stopping since there is no improvement on the model thus terminating. The plot below shows the performance of the model during training and validation.

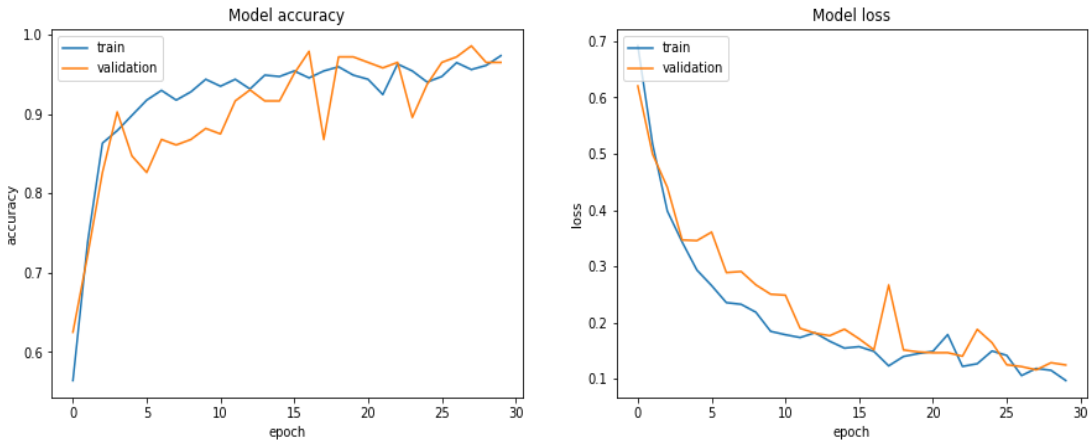

Fig. 1: Training plot for initial model

The plot shows that the model can be improved thus we shall use this model for transfer learning and fine-tuning.

Model Saving

As the model is being fitted, we do save the weights as the validation accuracy improves and this saved model is loaded for fine-tuning and retraining the model. This saved model is what we call and use for testing the detection of unknown datasets.

Transfer Learning and Model Fine-tuning

After training our model on the dataset without the classification layer of the pre-trained VGG16 model on imagenet dataset, we then take our weights that have been saved and retrain the model. Transfer learning has been used in visual recognition, natural language processing (NLP) tasks and medical diagnosis [4]. The goal of transfer learning and model fine-tuning is to take the features learned on the problem and leverage them on a similar problem. We compile our model again then fit it and save the weights again.

Fine-tuned Model Training Results

On fine-tuning the model, it can be observed that the training and validation accuracy improved and eventually the model converged and model training and validation loss also decreased as illustrated in Fig. 2. This is as a result of retraining the model on data its familiar with to enable it to learn it better.

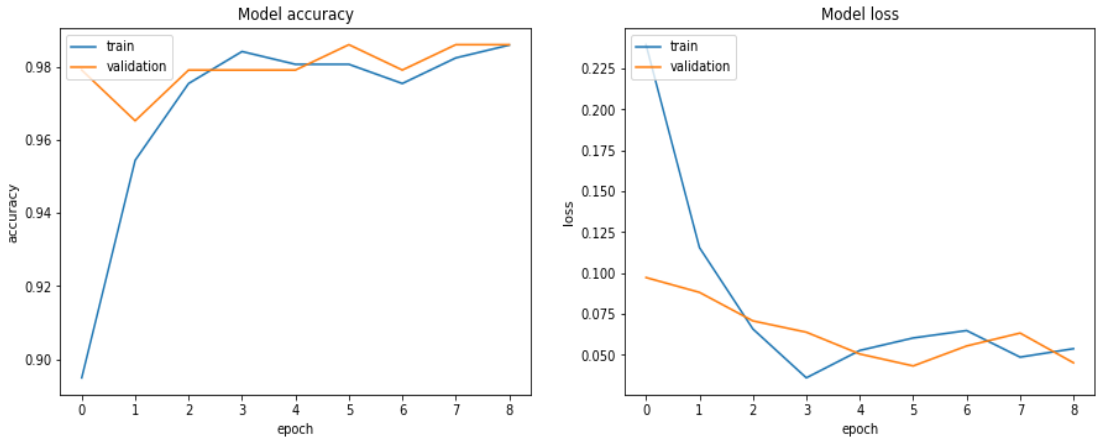

Fig. 2: Training plot for fine-tuned model

D. Results

We presented the model with 2 different datasets with one dataset taken under ideal conditions and the other testing dataset with images from not-so-great equipment and the results were as follows.

Confusion Matrix

We plotted confusion matrices for how the model performed while testing the 2 datasets and the results were as follows

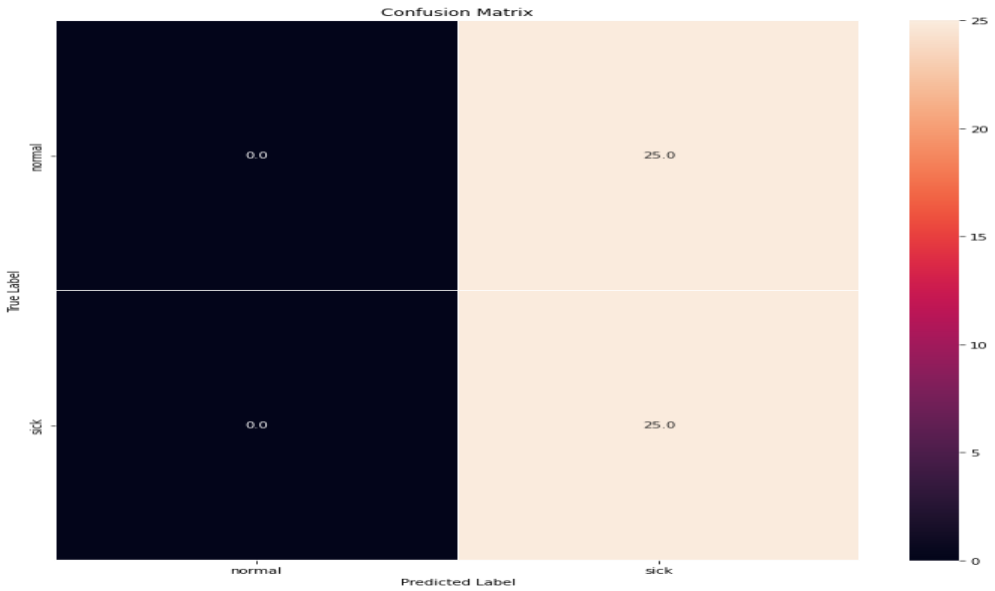

Fig.3: Confusion Matrix for dataset 1

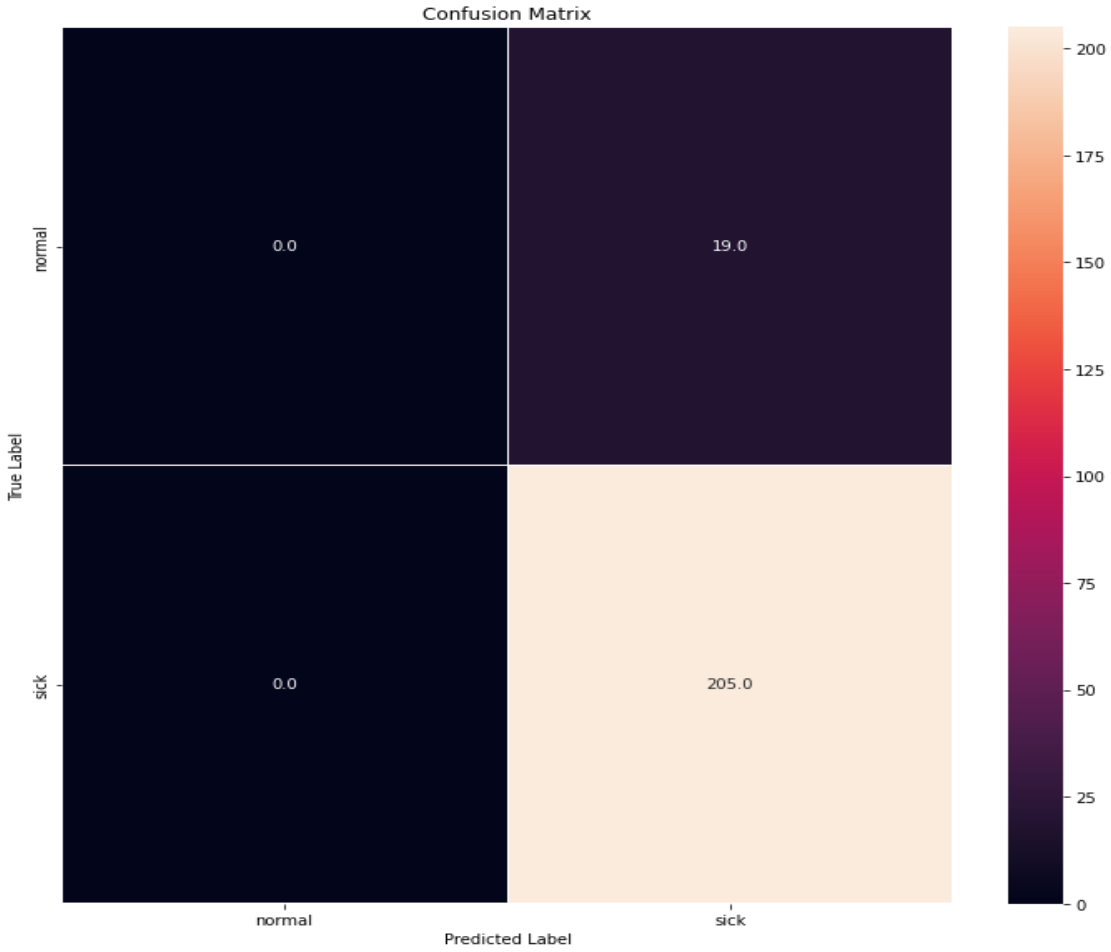

Fig.4: Confusion Matrix for dataset 2

Model Evaluation Criteria

We evaluated the model performance by using accuracy, precision, recall and F1 score. Accuracy is the total number of correct predictions out of all the predictions made. Precision is the out those predicted as sick, how many are really sick? Recall on the other hand the total number of those images which are sick and how many are actually sick. F1-score is the mean of precision and recall.

Table 1: Precision, Recall and F1-Score for dataset1 unknown images

index	precision	recall	f1-score	support
normal	0.0	0.0	0.0	0.0
sick	1.0	0.5	0.6666666666666666	50.0
accuracy	0.5	0.5	0.5	0.5
macro avg	0.5	0.25	0.3333333333333333	50.0
weighted avg	1.0	0.5	0.6666666666666665	50.0

Table 2: Precision, Recall and F1-Score for dataset 2

index	precision	recall	f1-score	support
normal	0.0	0.0	0.0	0.0
sick	1.0	0.9151785714285714	0.9557109557109557	224.0
accuracy	0.9151785714285714	0.9151785714285714	0.9151785714285714	0.9151785714285714
macro avg	0.5	0.4575892857142857	0.47785547785547783	224.0
weighted avg	1.0	0.9151785714285714	0.9557109557109557	224.0

E. Conclusion

Based on the evaluation of the performance of the model, it can be noted its leaning towards the sick class which especially with dataset 2 which was taken in not so ideal conditions. Thus image preprocessing can be key to improving the ability of the model to generalise on unknown images. Techniques to find regions of interest (ROI) is another key to improving the model's ability to generalise. The learning rates can also also be looked into as they play a major role in the convergence of a model which leads to better performance. The loss function can also be looked into to improve the performance and ability of the model to generalise. Instead of cross entropy, functions like focal loss can be employed as they give importanc eto hard exampelsgive importance to hard examples and put less weights for the parts of the data that are easily classified. Will also examine other methods using ResNet and DenseNet to run the experiments and find which method gives better results.

References

[1] T. Anwar and S. Zakir, "Deep learning based diagnosis of COVID-19 using chest CT-scan images," *2020 IEEE 23rd International Multitopic Conference (INMIC)*, Nov. 2020, doi: 10.1109/inmic50486.2020.9318212.

[2] "tf.keras.preprocessing.image.ImageDataGenerator," *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

[3] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv.org*, 2014. https://arxiv.org/abs/1412.6980

[4] X. He *et al.*, "Sample-Efficient Deep Learning for COVID-19 Diagnosis Based on CT Scans," Apr. 2020, doi: 10.1101/2020.04.13.20063941.