University of Osnabrück

Final Project

Course

Implementing ANN with TensorFlow

Winter Term 2022/2023

Comparison of CheXNet and ResNet binary classification performance on chest X-ray images

written by

Marie Spreen	Marta Sokol	Lena Tegge
991996	1006570	977643
mspreen@uni-osnabrueck.de	marta.katarzyna.sokol@gmail.com	ltegge@uni-osnabrueck.de
Cognitive Science	*Cognitive Science*	*Cognitive Science*

Handed in on April 1, 2023

Contents

List of Figures

List of Tables

1 Introduction

Pneumonia is a medical term describing the infection of the lungs that is caused either by bacteria or viruses [1]. Despite being seemingly treatable, the long-term consequences of the disease are, in fact, grave [2]. As such, it can be claimed that early diagnosis increased the likelihood of improved long-term health outcomes, and prevents early mortality.

Recent advances in deep learning have provided evidence that neural networks could be successfully used as an alternative to a diagnosis made by medical professionals, which would alleviate the burden on healthcare specialists. While a variety of different model architectures apply to this task, it is unclear which one delivers superior performance on different metrics.

One of the state-of-the-art models, that have been tested for the pneumonia classification task on chest X-ray data, is CheXNet - the model proposed by Rajpurkar et al. [3]. Despite its supposedly good performance on several metrics, its performance has never been compared with another architecture in a model trained on the same task and the same dataset, which would serve as an important possibility to evaluate the factual effectiveness of the model.

Therefore, in this project, we aim to compare the performance of CheXNet with an adapted residual neural network (ResNet). For that, we intend to reimplement the model proposed by Rajpurkar et al. [3], and compare it with ResNet, which will both be trained on the Chest X-ray 2017 dataset [4] on the binary classification task, differentiating between pneumonia-ridden and healthy individuals. We will evaluate the models based on the following metrics: accuracy, recall, F1, AUROC and precision.

The implications of this research include the possibility to improve the healthcare system and support the people in the medical sector by analysing large amounts of data and aiding them in providing faster and more accurate diagnoses. Thereby, patients will receive their treatment quicker and more people can be treated without missing out on personal care and accuracy.

2 Related literature

As depicted in the previous section our project is based on the paper “CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning” published in 2017 [3]. Rajpurkar et al. worked on an algorithm that can detect pneumonia, as well as other diseases, from frontal chest X-ray images [3]. They used the model CheXNet, which is a dense convolutional network with 121 layers and adapted the network such that they get a multidimensional output instead of the initial binary one [3].

Over the years, the approaches to classify medical image data to detect and identify diseases, have consistently been updated and re-evaluated. A decade ago in 2012, shallow networks with only "a single layer of kernels with a large receptive field" [5, p.63] were common, but the advantages of deeper networks, for example, "a lower memory footprint during inference" [5, p.63] soon gained popularity. In 2014 the model VGG19, which has 19 layers won the ImageNet challenge [5]. Unfortunately, deep networks like VGG19 with their lower memory footprint are not so relevant for medical applications [5]. It is shown that network architectures are evolving fast, but are not necessarily relevant in the medical field. Only a year later, He et al. designed residual networks comparing different amounts of layers [6]. The 152-layered ResNet outperforms most other models and also has lower complexity than VGG19 [6], making them win the ImageNet challenge in 2015 [5]. In the following year, deep networks for candidate detection performed almost as well as the standard rule-based image processing, which indicates a steady growth in the usability of deeper networks in comparison to shallower ones when it comes to problem-solving in the medical field [5]. To sum up Litjens et al.'s work from 2017 in their own words "exam classification CNNs are the current standard techniques. Especially CNNs pre-trained on natural images have shown surprisingly strong results, challenging the accuracy of human experts in some tasks." [5, p.66].

Also in 2017, Islam et al. used a deep convolutional neural network for feature detection in chest X-ray images, combining the proposed CNN structures with the advantages of deep neural networks [7]. In their study, the ResNet-152 achieved the highest accuracy among other ResNets as well as VGG models and AlexNet [7]. This again suggests that deeper networks are useful to detect features in medical image data, especially X-ray images. They also stated that Ashraf et al. showed features of shallower deep convolutional networks are useful to detect small objects in images [7]. Furthermore in their study, Islam et al. showed that all deep convolutional network approaches outperformed the rule-based ones [7], hinging that the supposedly standard method is not as efficient as previously assumed.

This journey suggests that convolutional neural networks, shallow as well as deeper ones, are a very common and effective procedure for medical image recognition, but considering current studies on the topic of identifying COVID symptoms from 2022, a deep neural network was used to detect features of X-ray images [8] as well as a Transfer learning approach including ResNet variants to detect pneumonia [9]. This again shows that neural networks are constantly evolving and that there is no fixed standard procedure for image classification tasks, at least not in medical image classification.

3 Methodology

We aim to compare the performance of two different models for binary classification of chest X-ray images into those with and without pneumonia.

The first model we use is the original CheXNet model [3], which is a Dense CNN with 121 layers that have been pre-trained on the ImageNet dataset. In the original paper, the model was initially designed to predict a binary distinction between pneumonia being present or not present in the presented chest X-ray image, before being extended to perform categorical classification across all 14 different conditions present in the ChestX-ray14 database. For our purposes, we only reimplement the binary classification version of the model, as our focus is on comparing this model to our own implementation.

In addition to the CheXNet model, we also attempt our own implementation of a binary classification model for pneumonia detection. This model is implemented and trained in the same way as the CheXNet model, but instead of the DenseNet121 architecture, we use the ResNet101 architecture from the Keras library [10] as a base model. We choose this model because it has shown good performance in various computer vision tasks, including image classification, and has also been pre-trained on the ImageNet dataset [11].

We train both models on the same chest x-ray data. The training process follows the implementation of the original paper.

We then evaluate both models performances on the test data using metrics such as accuracy, precision, recall, F1-score and AUROC and compare the results.

4 Implementation

For our implementation we test two approaches. First, we reimplement the CheXNet model in TensorFlow with the same steps Rajkapur et al. did, second we reimplement the CheXNet model but utilised a ResNet architecture as our base model. We want to compare the performance of these models to analyse whether a DenseNet or a ResNet is better suited to classify medical image data. The entire code is published on Github, accessible with the following link: Final Project.

4.1 CheXNet

Following the methods of the original study [3], we reimplement CheXNet – a modified 121-layer dense convolutional neural network, otherwise known as DenseNet [12], on a binary classification task. We use the CheXNet given its state-of-the-art status in the domain of pneumonia detection and its performance on the F1 metric that is superior to the radiologist average [3].

The reimplemented model based on the pre-trained DenseNet121 is loaded using the ImageNet-weights. This ensures better generalizability and quick training with smaller amounts of data. The pre-trained base model performs categorical classification, so to adapt our model to do binary classification we add a custom output layer. The output layer is a Dense layer of size 1 with a sigmoid activation. We define the model within a ChexNet class, which inherits methods from the base class tf.keras.Model. In the constructor method of this class, we instantiate empty zero and one weights attributes, as well as an empty model attribute, which is to be computed with successive methods. While the first of the aforementioned methods, get_weights(), simply compute the class distribution and uses it to update the weight attributes, the second method, get_model(), is more complex.

With the get_model() method, we firstly use the DenseNet121 as the backbone of our model that we pre-train on ImageNet while using global average pooling and removing the final output layer. We then add a custom output layer of size 1 with a sigmoid activation. Having done that, we update the empty self.model attribute by setting the modified DenseNet121 as an instance of tf.keras.models.Model class.

Next, we make use of a separate module that we import. In this module, we define a class WeightedCrossEntropyBinaryLoss, which we use in order to implement the weighted cross-entropy binary loss suggested in the original paper.
$L(X, y) = -[w_+ y \log(p(Y = 1|X)) + w_-(1 - y) \log(p(Y = 0|X))]$,
where:
- X is the input data
- y is the binary target variable (equal to 0 or 1)
- p(Y=1 | X) is the predicted probability of y=1 given X
- p(Y=0 | X) is the predicted probability of y=0 given X
- w_+ and w_- are the weights assigned to the positive and the negative classes, respectively.

This implementation can be seen in Figure 1. We compute these weights in the CheXNet class by counting the number of images in each class and then computing their distribution compared to the sum of the images in the two classes.

Having imported the loss module, we instantiate a WeightedCrossEntropyBinaryLoss object with the weights computed by the get_weights() method.

Finally, we compile the model, using Adam with default beta values as the optimizer, weighted cross-entropy binary loss as the loss function, and accuracy as our chosen metric.

In the main_chexNet.py file we instantiate the model, split and load the data and train our model. In the training, we introduce a reduction of

Figure 1: Implemented Weighted Cross Entropy Binary Loss

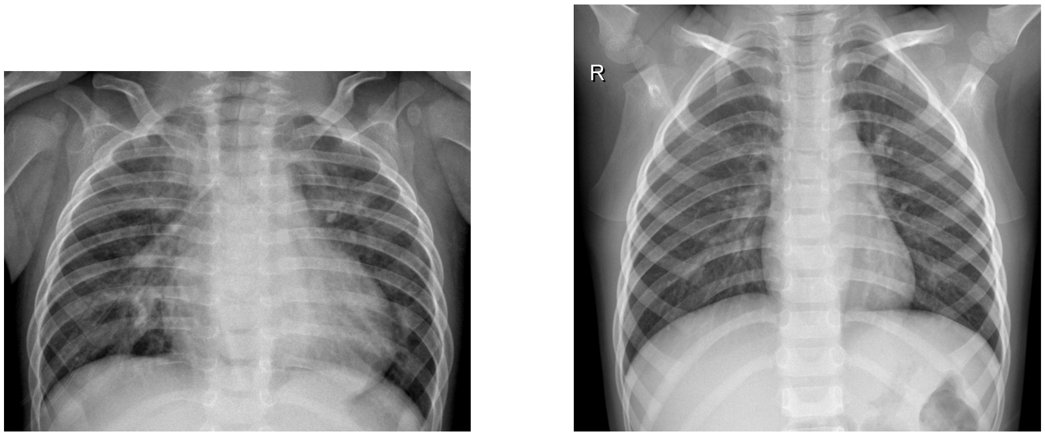
```python
import tensorflow as tf
from keras.backend import epsilon
import tensorflow.python.keras.backend as K

class WeightedCrossEntropyBinaryLoss:
    def __init__(self, zero_weight, one_weight):
        self.zero_weight = zero_weight
        self.one_weight = one_weight
    def weighted_binary_crossentropy(self, y_true, y_pred):
        y_true = K.cast(y_true, dtype=tf.float32)
        epsilon = tf.keras.backend.epsilon()
        y_pred = tf.clip_by_value(y_pred,
                                  epsilon, 1. - epsilon)
        # Compute cross entropy from probabilities.
        bce = y_true * tf.math.log(y_pred + epsilon)
        bce += (1 - y_true) * tf.math.log(1 - y_pred +
                                          epsilon)
        bce = -bce
        # Apply the weights to each class individually
        weight_vector = y_true * self.one_weight +
                        (1. - y_true)   * self.zero_weight
        weighted_bce = weight_vector * bce
        # Return the mean error
        return tf.reduce_mean(weighted_bce)
```

the learning rate as it was implemented in the original code. This means that we reduce the learning rate by a factor of 0.1, whenever our validation loss plateaus after an epoch. This prevents the likelihood of reaching only local optima. We also use early stopping as a protective measure against overfitting. Before we initiate the training, we define two model checkpoints - one that will save and store the best weights obtained as a result of the training, and another one that is used for the visualization of the training with the usage of TensorBoard.

4.2 ResNet

For the implementation of the ResNet, we follow the CheXNet structure. Instead of loading the pre-trained DenseNet121, we use the pre-trained Keras model ResNet101. This pre-trained base model also performs categorical classification. As done previously in the CheXNet model reimplementation we construct a custom output layer to enable binary classification. The output layer is a Dense layer of size 1 with a sigmoid activation. We define the model and its two functions in the ResNet class, which inherits methods of the base class tf.keras.Model and follows the same structure as the CheXNet class. In the constructor method of this class, we instantiate empty zero and one weights attributes, as well as an empty model attribute, which is to be computed with successive methods. We define a function get_weights(), which computes the class distribution and a function get_model(), which sets the ResNet101 as the base model. Just as done in the CheXNet model this base model is also loaded using the pre-trained ImageNet-weights while using global average pooling, and removing the final output layer. We then add the custom output layer of size 1 with a sigmoid activation and connect everything within the model.

Next, we make use of the same previously defined loss class: WeightedCrossEntropyBinaryLoss. In this class, we implement the weighted cross-entropy binary loss as suggested in the original paper. Using the



(a) X-ray with Pneumonia (b) X-ray without Pneumonia

Figure 2: Example Images from Chest X-ray Dataset

get_weights() class of the ResNet, we count the occurrences of both classes and use this as an input for the weights of the loss.

Finally, we again compile the model, using Adam with default beta values as the optimizer, weighted cross-entropy binary loss as the loss function, and accuracy as our chosen metric.

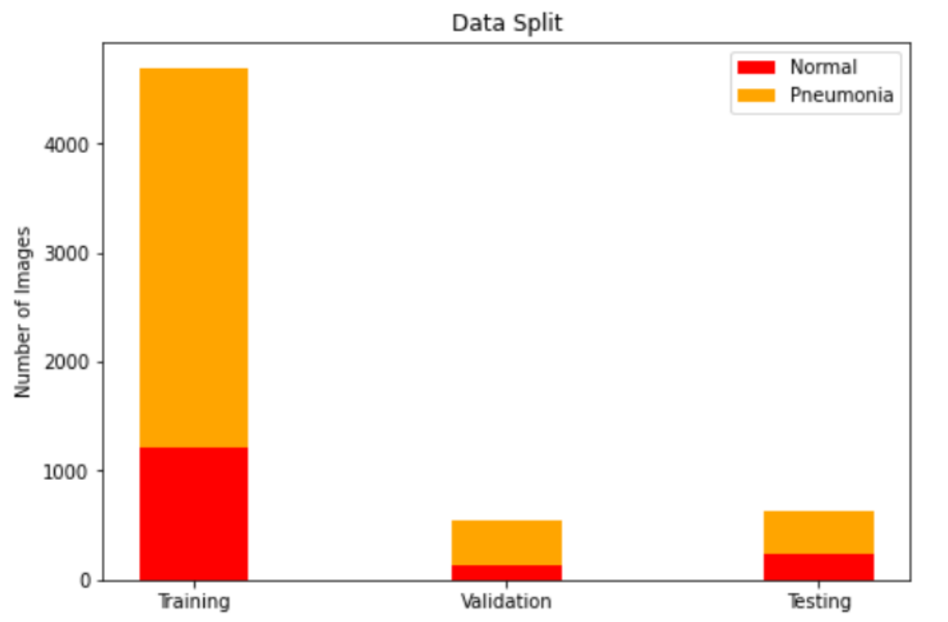
To perform the training we implement a main file which follows the same structure as previously defined for the CheXNet model.

4.3 Dataset

Instead of using the original ChestX-ray14 data set, we opt for a more practical, smaller chest x-ray data set [4]. The data set contains 5,863 images separated into three categories: training, validation, and testing. Each category is further divided into “normal” images (images without pneumonia) and “pneumonia-positive” images (images classified as having either bacterial or viral pneumonia), shown in Figure 2. In Figure 3 you can see the data split and distribution across categories.

We train both models on the same dataset, to ensure a fair comparison

Figure 3: Data Split



between the two. To utilize the data and obtain a TensorFlow dataset for training we define two separate functions. The split_data() function takes in training- and validation- filepaths and a validation-training-split and splits the data along the new ratio.

The prepare_data() function takes in the data filepaths, an image dimension and the batch size. It utilizes the ImageDataGenerator functionality to generate batches of tensor image data with real-time data augmentation from the provided image data. The function outputs two separate lists of batched image tensors for training- and validation data and two numpy arrays which contain the test-data images and labels respectively. The training- and validation data can then be used in the training of our models, whereas the two test-data arrays can be fed into our evaluation method.

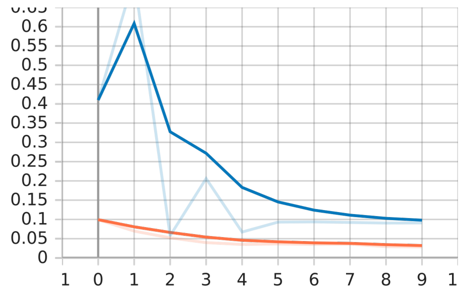
Figure 4: Evaluation Function Code

```python
import numpy as np
import tensorflow as tf
from sklearn.metrics import precision_recall_fscore_support, confusion_matrix, accuracy_score, roc_auc_score

def evaluate(model, test_data, test_labels, batch_size):

    # Compute predictions and round to obtain binary predictions
    yhat = np.round(model.predict(test_data, batch_size=batch_size))

    # Compute accuracy, confusion matrix, and metrics from confusion matrix
    acc = accuracy_score(test_labels, yhat) * 100
    tn, fp, fn, tp = confusion_matrix(test_labels, yhat).ravel()
    precision, recall, f1_score, _ = precision_recall_fscore_support(test_labels, yhat, average='binary')
    auroc = roc_auc_score(test_labels, yhat)

    # Print the results
    print("\nModel evaluation\n")
    print(f"Confusion Matrix:\n{confusion_matrix(test_labels, yhat)}\n")
    print(f"Accuracy: {acc}%")
    print(f"Precision: {precision * 100}%")
    print(f"Recall: {recall * 100}%")
    print(f"F1 Score: {f1_score * 100}")
    print(f"AUROC: {auroc * 100}")
```

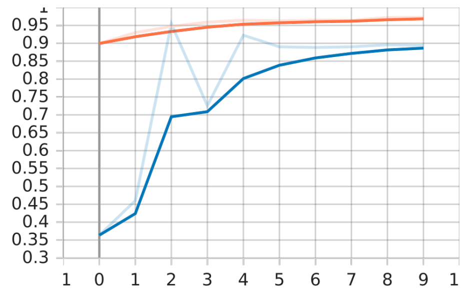
4.4 Evaluation

For the evaluation of both models, we utilize the sklearn library and its functions for determining the accuracy, confusion matrix, precision, recall, AUROC and f1 score of a model, given some test data. We define the function evaluate(). The function takes in a model, the test-images, the test-labels and the batch-size. It then generates the model's predictions for the test-data. Using the sklearn.metrics functions, it determines the accuracy, precision, recall, f1 score, AUROC and confusion matrix based on the predictions of the model and the true labels of the test-data. It prints the results to the screen. The code of the evaluation function can be found in Figure 4.

Additionally, we also observe and visualize the training progression using the TensorBoard, as illustrated in Figures 5 and 6.

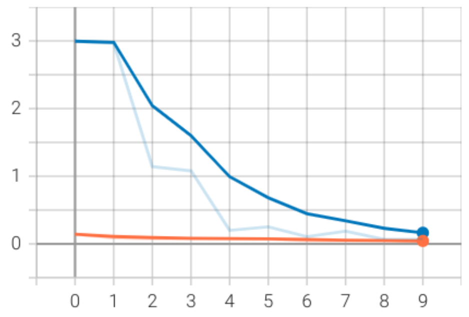


(a) Loss

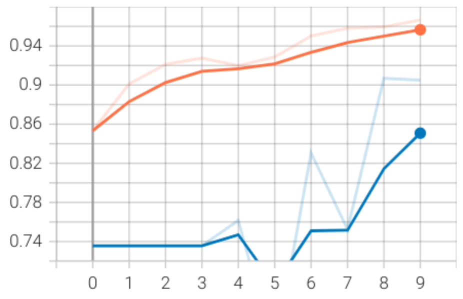


(b) Accuracy

Figure 5: CheXNet performance across 10 epochs of training
(in orange: training; in blue: validation)



(a) Loss



(b) Accuracy

Figure 6: ResNet performance across 10 epochs of training
(in orange: training; in blue: validation)

5 Results

As shown in Table 1, both models perform well on the test-data. Our reimplementation of the CheXNet achieves an AUROC of 0.8774 and our ResNet scores even higher with an AUROC of 0.9201. In comparison, the original paper describes an AUROC score of only 0.7680 for the task of detecting pneumonia.

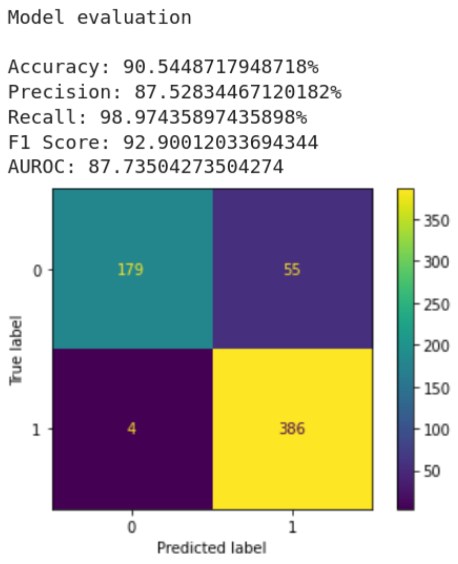
When comparing our reimplementation of CheXNet and our implementation of the ResNet, the ResNet performs better concerning Accuracy, Precision, F1-score and AUROC. The only metric within which the CheXNet outperforms the ResNet is the Recall.

The visualizations of the training process in Figure 5 and Figure 6 show that in both the training of the CheXNet and ResNet validation loss and accuracy steadily approach training loss and accuracy until both flatten out. This shows that neither model seems to be heavily overfitting since the validation loss and accuracy (performance on "unseen" data) steadily improve. There is one downward spike in the validation accuracy of the ResNet training progression, which suggests that some overfitting might have occurred, but it seems as though the problem was resolved shortly after.

In the Model Evaluation (Figure 7) we can see the previously mentioned metrics, across which we compared both models, as well as the confusion matrix. The confusion matrix displays the true negatives in the left upper corner, the false positives in the right upper corner, the false negatives in the left lower corner and the true positives in the right lower corner. A model is best if it has as few false positives (upper right) and false negatives (lower left) as possible. In our visualisation, you can see that both of those quadrants for both models are coloured dark blue to purple, which signifies low numbers, as can be read on the colour bar on the right. Additionally, it becomes obvious that the ResNet performs better in precision, as it has more true positives and fewer false positives (right side) than the CheXNet. The CheXNet on the other hand performs better in recall since it has significantly fewer false negatives (lower left corner) and therefore a higher fraction of true positives.

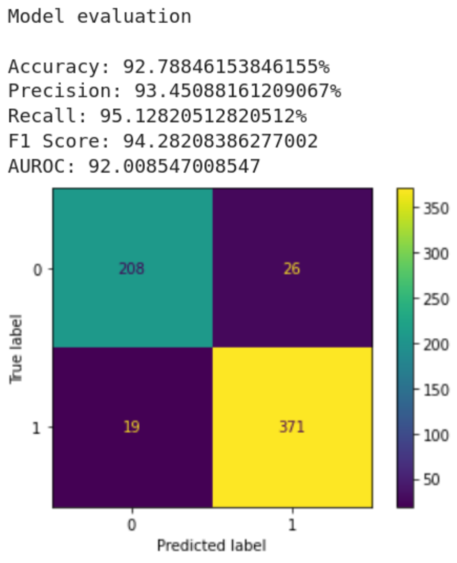
	CheXNet	ResNet
Accuracy	90.54%	92.79%
Precision	87.53%	93.45%
Recall	98.97%	95.13%
F1	92.90%	94.28%
AUROC	87.74%	92.01%

Table 1: Comparison of model evaluations between ResNet and CheXNet on chosen metrics.



```
Model evaluation

Accuracy: 90.5448717948718%
Precision: 87.52834467120182%
Recall: 98.97435897435898%
F1 Score: 92.90012033694344
AUROC: 87.73504273504274
```

(a) CheXNet (DenseNet)



```
Model evaluation

Accuracy: 92.78846153846155%
Precision: 93.45088161209067%
Recall: 95.12820512820512%
F1 Score: 94.28208386277002
AUROC: 92.008547008547
```

(b) ResNet

Figure 7: Model Evaluation

6 Discussion

In this paper, we describe the process of re-implementing the CheXNet model and also developing a ResNet101 model to detect pneumonia from chest X-rays. Both models achieve high performance on the test data, with the ResNet101 outperforming the CheXNet in terms of accuracy, precision, F1 score, and AUROC, while the CheXNet has a higher recall.

One potential reason for the superior performance of the ResNet101 model is the deeper architecture of the ResNet model and its use of residual connections. ResNets are designed to address the problem of vanishing gradients in very deep neural networks by allowing gradients to flow directly through shortcut connections, thus facilitating the training of very deep networks. In contrast, DenseNets are designed to encourage feature reuse by densely connecting each layer to every other layer in a feedforward fashion, which can lead to a reduction in the redundancy of features and more efficient use of parameters.

We can also consider that recall is a very important metric in a medical context since it measures the fraction of positive cases that were successfully detected. We could argue that it is more important for a medical model to detect truly positive cases than it is for the model to avoid falsely classifying negative cases. In that sense, the CheXNet model performs better regarding a very important metric.

Some of the problems we encountered during the implementation process might have added to the results or even distorted them.

In both the implementation of the CheXNet and ResNet, our efforts were limited by the relatively high computational cost given our technological resources, which impacted our ability to fine-tune hyperparameters adequately. Therefore, it would be advisable to further experiment with adjusting the hyperparameters. Additionally, the results might be improved by adding more training epochs, although, given our smaller dataset, we might run into problems of overfitting.

Initially, we were aiming to replicate the original paper as closely as possible, which included using the original ChestX-Ray14 dataset. Unfortunately, the massive amount of images and redundant labels made data processing both time-costly and relatively complex. We implemented a method to handle the amount of data and create TensorFlow data sets for training, validation and testing but discarded it after we realised it would take too long to preprocess the data and also prolong the training. Therefore, we decided to use another data set, that is the ChestXRay 2017 dataset [4]. The main advantage of this dataset is that its relatively smaller size allows for faster training, which is particularly beneficial due to our limited computational resources. Its second advantage is that it is already split into two classes, that is the "normal" category (containing X-ray chest images of healthy individuals) and the "pneumonia" category (containing X-ray images of infected individuals). Furthermore, the data set is also already split into the training, testing, and validation sets. This led to some complications of its own. Since the ratio of training to validation data did not fit our needs, we had to adjust the split. For this, we wrote a function that takes in a preferred training-validation split and transfers images from the training set to the validation set accordingly.

Since we did not end up utilizing the original data set, it is difficult to compare our results to the results in the original paper. The data set we used for training and evaluating our models is smaller than the original data set. In addition, the published results only contain an evaluation of the classification across all 14 chest x-ray categories, while both of our models are evaluated on their binary classification scores. This explains why the AUROC results are considerably better for both of our models (CheXNet: 0.8774, ResNet: 0.9201 vs. Original CheXNet: 0.7680). Since the original model has to apply itself more generally across 13 additional conditions, its performance on the specific condition of pneumonia goes down slightly. Meanwhile, our two implementations are specifically finetuned towards the condition of pneumonia.

Another problem we encountered is that of implementing appropriate visualization and storage of our results. We ended up using the Keras callback objects to not only implement useful features for our training, like early stopping and reducing the learning rate upon reaching a plateau, but also for visualization and checkpoint storage. After each epoch, if we achieve a lower validation loss than previously, we save our weights in a loadable weight file. Once training is finished we can then retrieve the best model by loading the saved weights. In addition, we save the respective training and validation loss and accuracy in log files, which can later be retrieved and visualized through the TensorBoard (Figure 5 and Figure 6).

In CheXNet, the function tf.keras.Model.predict() had to be used since tf.keras.Model.evaluate() returned an error caused by the weighted binary cross-entropy loss, despite having it specified as a custom object.

References

[1] "Pneumonia - What Is Pneumonia? | NHLBI, NIH." (Mar. 24, 2022), [Online]. Available: `https://www.nhlbi.nih.gov/health/pneumonia` (visited on 04/01/2023).

[2] C. Stotts, V. F. Corrales-Medina, and K. J. Rayner, "Pneumonia-Induced Inflammation, Resolution and Cardiovascular Disease: Causes, Consequences and Clinical Opportunities," *Circulation Research*, vol. 132, no. 6, pp. 751–774, Mar. 17, 2023. DOI: `10.1161/CIRCRESAHA.122.321636`. [Online]. Available: `https://www.ahajournals.org/doi/full/10.1161/CIRCRESAHA.122.321636` (visited on 04/01/2023).

[3] P. Rajpurkar, J. Irvin, K. Zhu, *et al.* "CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning." arXiv: `arXiv:1711.05225`. (Dec. 25, 2017), [Online]. Available: `http://arxiv.org/abs/1711.05225` (visited on 03/14/2023), preprint.

[4] D. Kermany, K. Zhang, and M. Goldbaum, "Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification," vol. 2, Jan. 6, 2018. DOI: `10.17632/rscbjbr9sj.2`. [Online]. Available: `https://data.mendeley.com/datasets/rscbjbr9sj/2` (visited on 03/31/2023).

[5] G. Litjens, T. Kooi, B. E. Bejnordi, *et al.*, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, Dec. 2017, ISSN: 1361-8423. DOI: `10.1016/j.media.2017.07.005`. pmid: 28778026.

[6] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition." arXiv: `arXiv:1512.03385`. (Dec. 10, 2015), [On-

line]. Available: http://arxiv.org/abs/1512.03385 (visited on 03/27/2023), preprint.

[7] M. T. Islam, M. A. Aowal, A. T. Minhaz, and K. Ashraf. "Abnormality Detection and Localization in Chest X-Rays using Deep Convolutional Neural Networks." arXiv: arXiv:1705.09850. (Sep. 27, 2017), [Online]. Available: http://arxiv.org/abs/1705.09850 (visited on 03/27/2023), preprint.

[8] H. Nasiri, G. Kheyroddin, M. Dorrigiv, *et al.* "Classification of COVID-19 in Chest X-ray Images Using Fusion of Deep Features and LightGBM." arXiv: arXiv:2206.04548. (Jun. 27, 2022), [Online]. Available: http://arxiv.org/abs/2206.04548 (visited on 03/14/2023), preprint.

[9] S. Showkat and S. Qureshi, "Efficacy of Transfer Learning-based ResNet models in Chest X-ray image classification for detecting COVID-19 Pneumonia," *Chemometrics and Intelligent Laboratory Systems*, vol. 224, p. 104 534, May 15, 2022, ISSN: 0169-7439. DOI: 10.1016/j.chemolab.2022.104534. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743922000454 (visited on 03/27/2023).

[10] K. Team. "Keras documentation: ResNet and ResNetV2." (), [Online]. Available: https://keras.io/api/applications/resnet/ (visited on 03/31/2023).

[11] O. Russakovsky, J. Deng, H. Su, *et al.* "ImageNet Large Scale Visual Recognition Challenge." arXiv: arXiv:1409.0575. (Jan. 29, 2015), [Online]. Available: http://arxiv.org/abs/1409.0575 (visited on 04/01/2023), preprint.

[12] G. Huang, Z. Liu, p. d. u. family=Maaten given=Laurens, and K. Q. Weinberger. "Densely Connected Convolutional Networks." arXiv:

`arXiv:1608.06993`. (Jan. 28, 2018), [Online]. Available: `http://arxiv.org/abs/1608.06993` (visited on 03/31/2023), preprint.