Pneumonia Detection using a CNN

Bram Pulles – S1015194

June 3, 2024

1 Introduction

Pneumonia is a serious lung condition responsible for the deaths of millions each year. Diagnoses are often performed through the analysis of chest X-ray images, due to its wide spread availability and relative low cost. AI can be used to help professionals get to a diagnosis. In this report we outline a computer vision method using convolutional neural networks (CNN) to detect pneumonia.

Much research has gone into computer aided pneumonia detection. We take the existing research done by Szepesi, et al (2022) and reimplement their architecture. In contrast to other approaches using transfer learning, they focus on a simple CNN that was not pre-trained. Their research is unique, because they add a dropout layer to a convolutional block, which is unconventional. We will use their results to set the optimal parameters of our network.

In section 2 we dive into the inner workings of our CNN, as well as the dataset, learning procedure and evaluation criteria of our model. We will discuss the architecture in detail and outline the parameter settings that we use. In section 3 we discuss the performance of the CNN on the training data and relate these to various measures. In section 4 we conclude with a critical review to our study and findings.

2 Methods

2.1 Dataset

The dataset on which we train the CNN is provided through a Kaggle competition[1]. It contains a total of 5863 chest X-ray images, labeled as either normal or pneumonia. The images are taken of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center. All images are taken as part of routine patient clinical care. The images have been filtered before being admitted to the dataset by removing low quality images. The labels for the images are determined by two expert physicians. In addition, a third expert has checked the labels for errors.

Szepesi, et al (2022) uses the same dataset. However, they use additional images for training made by generative AI. The images are generated for the pneumonia labeled cases as these are less common in the data. We do not generate extra images in this way. Instead we use a simple data generator which performs tranformations on the images to generate extra data. In particular, we randomly rotate the images by at most 30 degrees, zoom in by at most 20%, shift in height and width by at most 10% and randomly flip the images left-right. Of course, we also normalize the data to value between 0 and 1 by dividing through 255. At last, we resize the images to a size of 224 by 224 to be in line with Szepesi, et al (2022) and other literature.

Originally, the dataset is split into a train, test and validation set, where the latter only contains a total of 16 samples. We resample the data such that it is a 80%/10%/10% split. This way we can actually use the validation set and test the generalisation behaviour of our model with statistical significance. As this problem applied to more people on Kaggle we use the snippet provided by

[1] `https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia`

Mahdi Ravaghi to perform this reshuffling of the data[2]. The distribution of samples after the resampling can be seen in figure 1.

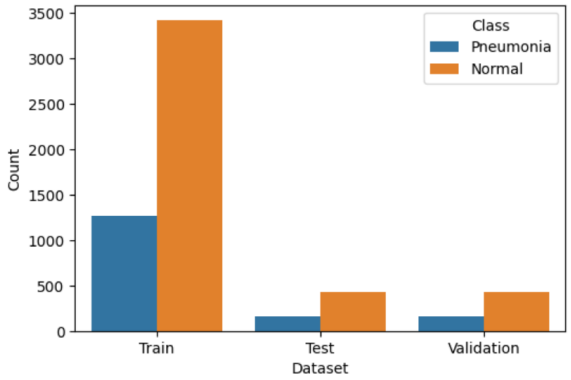

Figure 1: The dataset sample count after the 80%/10%/10% split.

2.2 Model

An impression of a similar model architecture can be seen in figure 2. As visible, it consists of several chained convolutional blocks. In our model there are five blocks each consisting of two convolutions both followed by a batch normalisation and ReLU activation layer. The batch normalisation layers normalize the inputs for the next layer helping to stabilize learning, reaching convergence more quickly and introducing a bit of noise which can have a slightly regularizing effect. The convolution kernels are of size 3 by 3, use zero padding to maintain spatial dimensions and do not use biases. Each convolutional block is ended with a max pooling of 2 by 2 or 3 by 3 to summarize the feature maps provided by the convolutional layers.

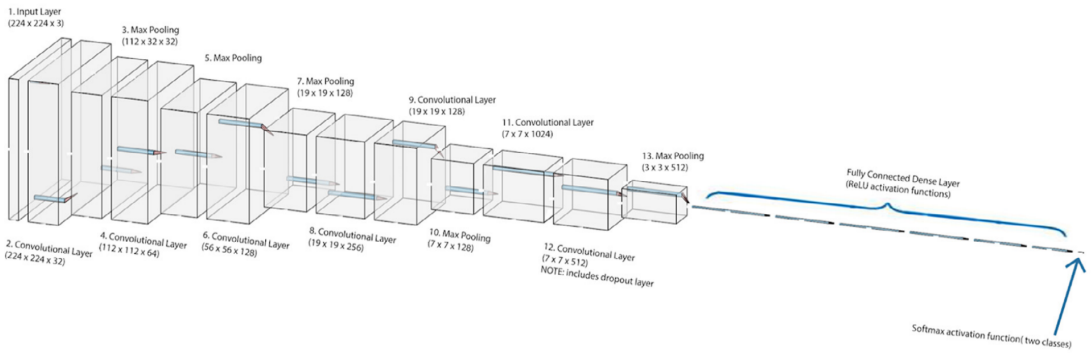

Figure 2: Structure of the model from Szepesi, et al (2022).

Before the chained convolutional blocks there is the input layer. This input layer takes the three channeled RGB encoded images and passes it straight to the first block. After the five blocks there is a flattening layer followed by a series of densely connected layers with ReLU activation functions in between. This fully connected dense set of layers operates on the feature maps provided by the convolutional blocks before and determines whether there is pneumonia or not. In the final output layer we have two output nodes on which we perform the softmax operation to get a probability for each label. In appendix A a full specification of all layers is given.

[2] `https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia/discussion/485689`

Importantly, Szepesi, et al (2022) introduces the unique integration of a dropout layer in the fifth convolutional block. The dropout occurs after the second convolutional layer, but before the batch normalisation step. They found that a dropout rate of 0.4 performs best, which is what we use as well. The dropout layer regularizes the network and prevents it from becoming over dependent on a small number of variables.

2.3 Learning

The loss which we optimize for is defined as the categorical cross entropy on the validation set, an obvious choice for our problem. We use the Adam optimizer with a learning rate of 1e-4. However, this learning rate is not static. As the model improves it can be beneficial to lower the learning rate to make smaller more accurate steps and avoid overshooting. To this end, we use a learning rate reduction, on par with Szepesi, et al (2022). If the validation loss does not improve for 5 epochs, we lower the learning rate by a factor of 0.8 to a maximum of 1e-6. If the validation loss does not improve over 15 epochs we stop and call it done. When we do not stop early, we perform a total of 50 epochs.

2.4 Evaluation

After training we evaluate the model performance using a few metrics. We evaluate the accuracy and loss over time (for each epoch) on both the train and validation datasets. Using the test set, we also compute a confusion matrix, describing the rate of true positive, true negative, false positive and false negative predictions. Especially, false positive and false negative evaluations are important in the medical realm, as you do not want to do a medical procedure on a healthy patient, or, not do a procedure on a sick patient. Precision and recall on the test set allow us to get an even better image of these properties.

3 Results

The training and validation accuracy and loss over time are visualised in figure 3. From these figures we can see that the training accuracy and loss follow a very stable trajectory. This is good, it suggests we chose good values for the learning rate over time. We also see that the validation accuracy and loss is less stable, but seems to get more stable over time. The validation set measures the generalisable performance of the model and it makes sense that this is not as stable at the start. It is promising that it stabilises over time, this is desired behaviour. Overall, it would be good to run the training for more epochs, however, this already takes about 3 hours.

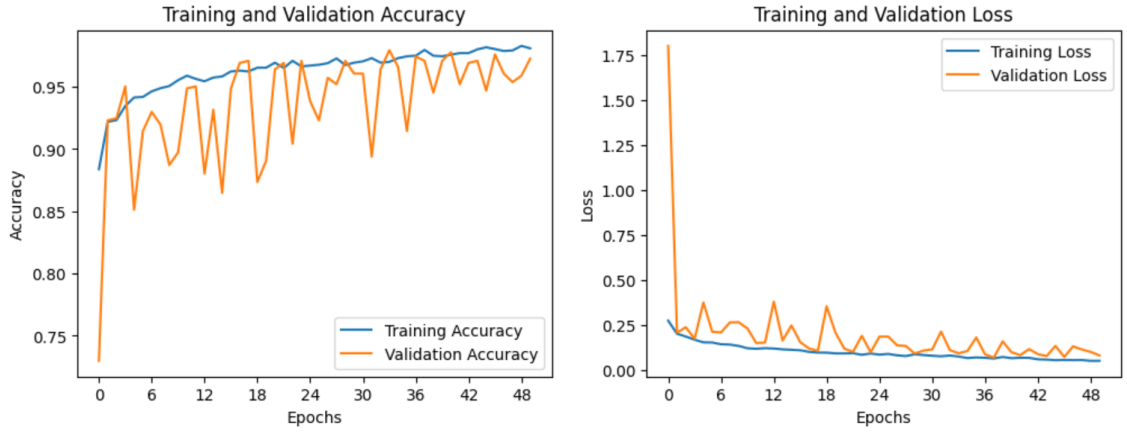

Figure 3: Accuracy and loss over time for the training and validation sets.

The confusion matrix is visualised in figure 4. From this confusion matrix we can derive all our metrics as seen in table 1. Additionally, we achieved an accuracy of 95.91% on the test set. Overall, these results are very good and close to the results in Szepesi, et al (2022). The precision on the pneumonia class is a bit lower, this can probably be explained by the imbalance in the number of test cases available for each class. The original paper solved this issue with using generative AI to create an even amount of images, we did not use this approach.

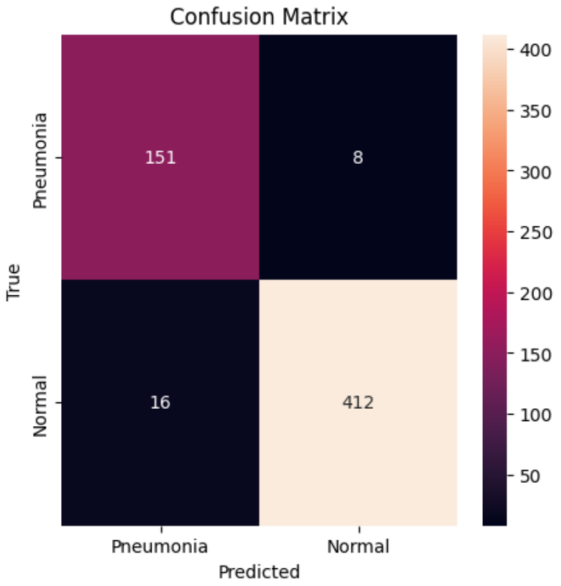

Figure 4: Confusion Matrix for the final model on the test set.

	precision	recall	support
Pneumonia	0.90	0.95	159
Normal	0.98	0.96	428

Table 1: Metrics of the final model on the test set.

4 Discussion

We have seen in this report how a CNN model as proposed by Szepesi, et al (2022) can be implemented and how it performs under slightly different circumstances. The CNN model accurately distinguishes X-ray images of pneumonia and normal lungs. The model achieves high precision and recall, most metrics being close to other state of the art solutions.

For the future, we can likely improve the performance further by increasing the number of epochs. We can also work on the data imbalance by generating or retrieving more pneumonia lung images. It would also be interesting to do a hyperparameter search on the model architecture, but this would be very computationally heavy. Overall, the performance of the current model is already excellent, but there are ideas to make it even better.

References

- Patrik Szepesi, László Szilágyi, Detection of pneumonia using convolutional neural networks and deep learning, Biocybernetics and Biomedical Engineering, Volume 42, Issue 3, 2022, Pages 1012-1022, ISSN 0208-5216, https://doi.org/10.1016/j.bbe.2022.08.001.

A Model

Layer (type)	Output shape	Parameters
(InputLayer)	[(224, 224, 3)]	0
(Conv2D)	(224, 224, 32)	864
(Batch Normalization)	(224, 224, 32)	128
(Activation)	(224, 224, 32)	0
(Conv2D)	(224, 224, 32)	9216
(Batch Normalization)	(224, 224, 32)	128
(Activation)	(224, 224, 32)	0
(MaxPooling2D)	(112, 112, 32)	0
(Conv2D)	(112, 112, 64)	18432
(Batch Normalization)	(112, 112, 64)	256
(Activation)	(112, 112, 64)	0
(Conv2D)	(112, 112, 64)	36864
(Batch Normalization)	(112, 112, 64)	256
(Activation)	(112, 112, 64)	0
(MaxPooling2D)	(56, 56, 64)	0
(Conv2D)	(56, 56, 128)	73728
(Batch Normalization)	(56, 56, 128)	512
(Activation)	(56, 56, 128)	0
(Conv2D)	(56, 56, 128)	147456
(Batch Normalization)	(56, 56, 128)	512
(Activation)	(56, 56, 128)	0
(MaxPooling2D)	(19, 19, 128)	0
(Conv2D)	(19, 19, 256)	294912
(Batch Normalization)	(19, 19, 256)	1024
(Activation)	(19, 19, 256)	0
(Conv2D)	(19, 19, 128)	294912
(Batch Normalization)	(19, 19, 128)	512
(Activation)	(19, 19, 128)	0
(MaxPooling2D)	(7, 7, 128)	0
(Conv2D)	(7, 7, 1024)	1179648
(Batch Normalization)	(7, 7, 1024)	4096
(Activation)	(7, 7, 1024)	0
(Conv2D)	(7, 7, 512)	4718592
(Dropout)	(7, 7, 512)	0
(Batch Normalization)	(7, 7, 512)	2048
(Activation)	(7, 7, 512)	0
(MaxPooling2D)	(3, 3, 512)	0
(Flatten)	(1, 4608)	0
(Dense)	(1, 512)	2359808
(Dense)	(1, 1024)	525312
(Dense)	(1, 512)	524800
(Dense)	(1, 512)	262656
(Dense)	(1, 256)	131328
(Dense)	(1, 64)	16448
predictions (Dense)	(1, 2)	130

Figure 5: Model specification details, same as in Szepesi, et al (2022).