DL Project - Pneumonia Detection with Chest X-RayImages*

Thomas Schmidt, Chonratid Pangdee, Måns Oskarsson, Simon Pontin

Abstract— **The aim of this paper was to train a convolutional neural network on chest X-ray images of healthy people and people suffering from pneumonia in order to find the hyper parameters that most efficiently detected pneumonia. The goal was to achieve an accuracy of at least 70-80%. This study was conducted by experimenting with different hyper parameters and documenting the performance of each agent where the documented metrics consisted of plots of accuracy, loss, confusion matrices, and F1-score and finally an accuracy on the test-set. It was deduced that a simpler network with ReLU, Adam and MSE yields the better results in terms of accuracy and generalization. If another network would be designed, the amount of trainable parameters would lie in the range 1,925,889 - 31,665,153 with the aforementioned activation function, optimizer and loss function as such a network could potentially yield a higher accuracy than 90%.**

I. INTRODUCTION

In today's current pandemic one must know that Pneumonia is a dangerous complication regarding the COVID-19 infection. The biggest reason for casualties is Pneumonia (COVID-19 related) and there are many methods in order to detect it. One is using a feed forward Convolutional Neural Network, trained on a set of x-ray images from patients either being affected by pneumonia or not. The model will be trained on a binary classification series of input data, where either a patient has pneumonia or not depending on the picture being evaluated by the network. The input layer will take normalized pictures (x-ray pictures are gray scaled already), meaning the input layer will take three dimensional pictures. Spatial, positional and color (0-1 gray scale) features at given pixels are evaluated. The output layer will either be '1', given the patient has pneumonia. Or '0' patient is symptom free. The given result from the output layer will be compared to a series of test data, where the network (output) probability of guessing on symptom free or not is evaluated. Experiments will be done in order to achieve the best possible performance of the network. Accuracy, F1 score and confusion matrix are the major indicators for measuring performance.

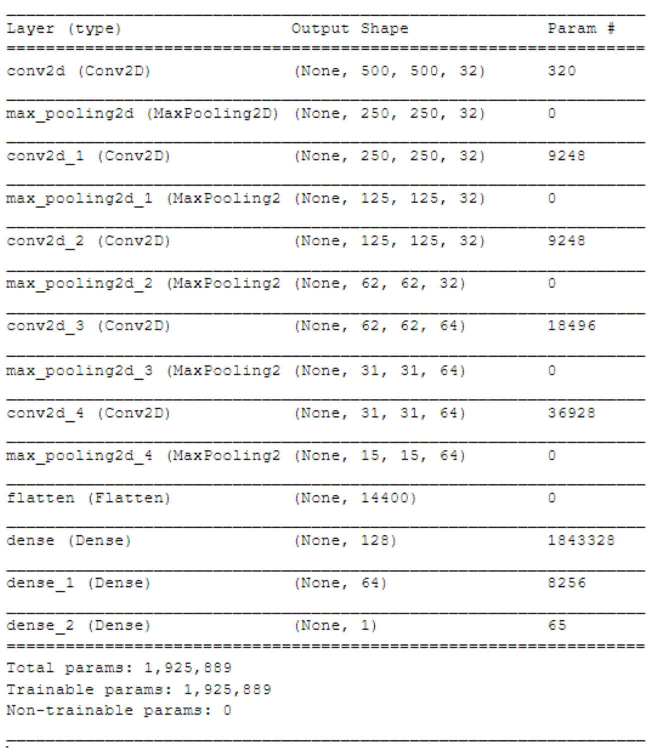
II. METHOD

In order to be able to find optimal parameters for the convolutional neural network, it was concluded the performance of different network architectures must be easily compared with respect to one another. There are however many different metrics to consider regarding the performance of a neural agent which is why it was decided to use the metrics mentioned in the *introduction*. Even though the accuracy and loss gives some insight into how well the agent is learning, it was necessary to understand if the network was somehow biased toward one of the two classes which is why confusion matrices and F1-score was used.

Based on what could be interpreted from the different metrics it could be decided how to improve the model and how to move forward with the next experiment. That is, the metrics gives vital information of what modifications that increased performance (and therefore should be kept) and what modifications that decreased performance. It was also decided to implement t-SNE and PCA visualizations in order to determine how well the convolutional part of the network performed as it directly affects the performance of the fully connected layers.

III. RESULT

A. Prototype 1



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 500, 500, 32)      320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 250, 250, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 250, 250, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 125, 125, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 125, 125, 32)      9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 62, 62, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 31, 31, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 15, 15, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 14400)             0
_________________________________________________________________
dense (Dense)                (None, 128)               1843328
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 1,925,889
Trainable params: 1,925,889
Non-trainable params: 0
_________________________________________________________________
```

Fig. 1: Shows the model layer architecture of prototype 1.

Hyperparameters	
Parameters	Values
Learning rate	0.001
Patience	2
Factor	0.3
Min learning rate	0.000001
Activation function	ReLU
Optimizer	Adam
Loss function	Binary Crossentropy
Epoch	25

TABLE I: Hyperparameters of prototype 1.

Performance metrics	
Accuracy	91.827 %
F1 score Normal	0.86
F1 score Pneumonia	0.92

TABLE II: Performance metrics of prototype 1.

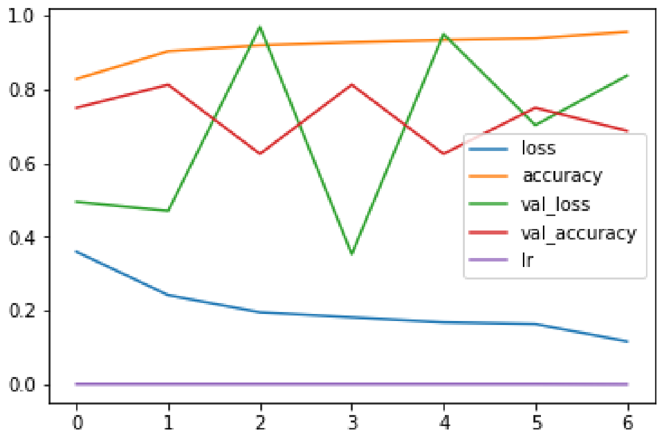


Fig. 2: Displays loss, accuracy, validation loss, validation accuracy and learning rate from training of prototype 1.

B. Prototype 2

Prototype 2 used the same network as shown in figure 1.

Hyperparameters	
Parameters	Values
Learning rate	0.001
Patience	2
Factor	0.3
Min learning rate	0.000001
Activation function	ReLU
Optimizer	SGD
Loss function	Binary Crossentropy
Epoch	25

TABLE III: Hyperparameters of prototype 2.

Performance metrics	
Accuracy	89.423 %
F1 score Normal	0.85
F1 score Pneumonia	0.92

TABLE IV: Performance metrics of prototype 2.

C. Prototype 3

Prototype 3 used the same network as shown in figure 1.

Hyperparameters	
Parameters	Values
Learning rate	0.001
Patience	2
Factor	0.3
Min learning rate	0.000001
Activation function	ReLU
Optimizer	SGD
Loss function	Mean squared error
Epoch	25

TABLE V: Hyperparameters of prototype 3.

Performance metrics	
Accuracy	88.622 %
F1 score Normal	0.84
F1 score Pneumonia	0.91

TABLE VI: Performance metrics of prototype 3.

D. Prototype 4

Prototype 4 used the same network as shown in figure 1.

Hyperparameters	
Parameters	Values
Learning rate	0.001
Patience	2
Factor	0.3
Min learning rate	0.000001
Activation function	ReLU
Optimizer	Adam
Loss function	Mean squared error
Epoch	25

TABLE VII: Hyperparameters of prototype 4.

Performance metrics	
Accuracy	91.346 %
F1 score Normal	0.87
F1 score Pneumonia	0.93

TABLE VIII: Performance metrics of prototype 4.

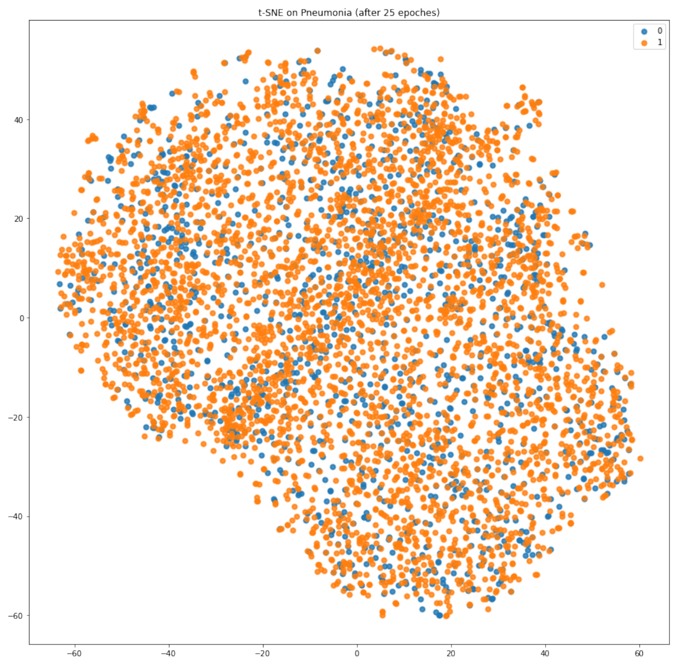


Fig. 3: Shows a t-SNE plot from the convolutional flattened output after 25 epochs

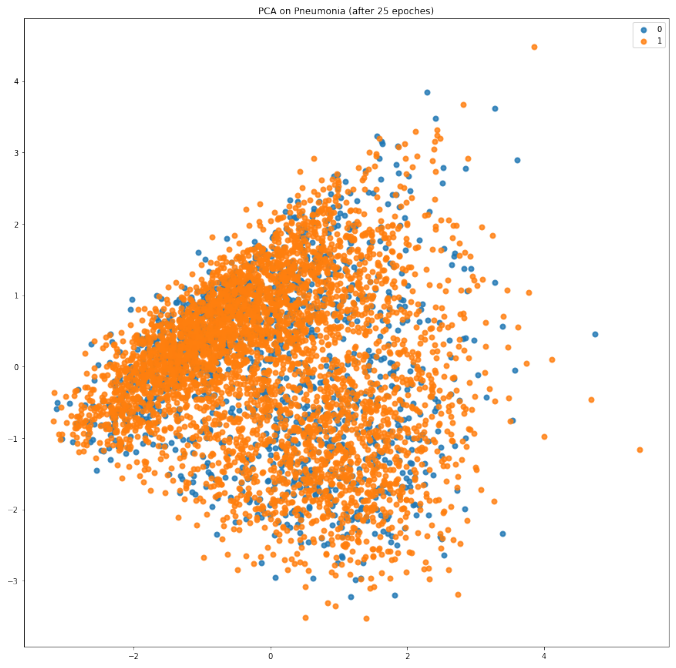


Fig. 4: Shows a PCA plot from the convolutional flattened output after 25 epochs

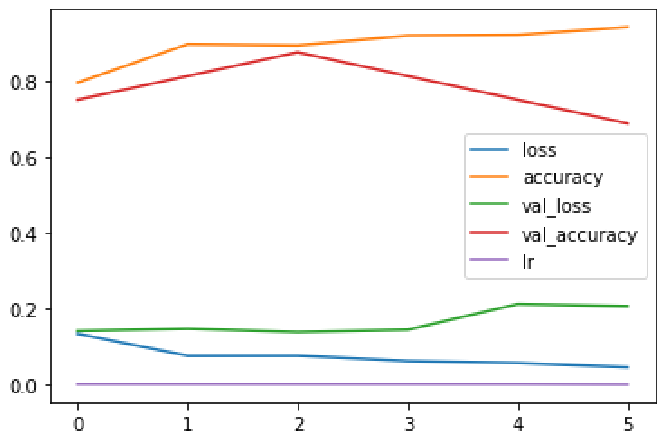
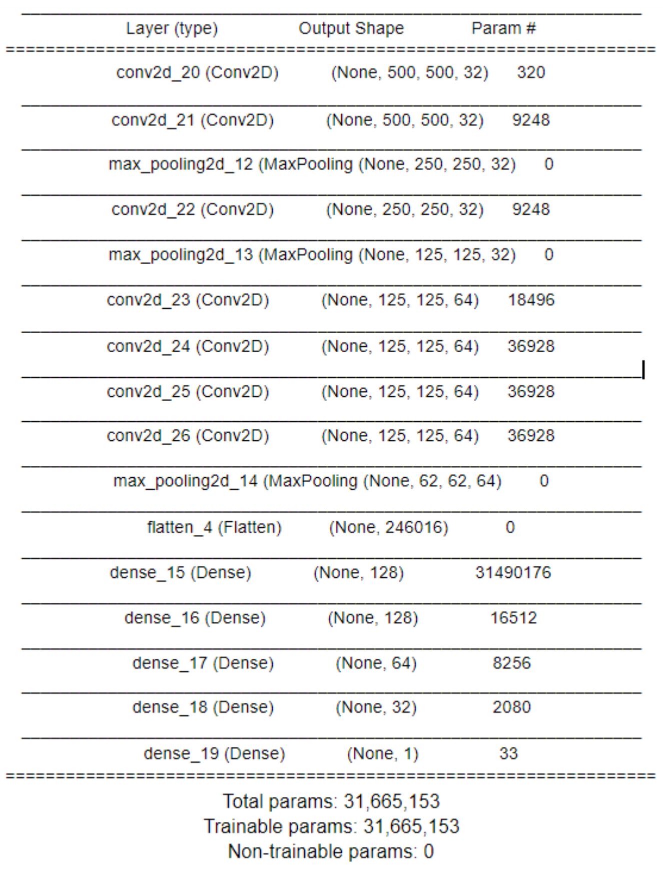


Fig. 5: Displays loss, accuracy, validation loss, validation accuracy and learning rate from training of prototype 4.

E. Prototype 5



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_20 (Conv2D)           (None, 500, 500, 32)      320
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 500, 500, 32)      9248
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 250, 250, 32)      0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 250, 250, 32)      9248
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 125, 125, 32)      0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 125, 125, 64)      18496
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 125, 125, 64)      36928
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 125, 125, 64)      36928
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 125, 125, 64)      36928
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 62, 62, 64)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 246016)            0
_________________________________________________________________
dense_15 (Dense)             (None, 128)               31490176
_________________________________________________________________
dense_16 (Dense)             (None, 128)               16512
_________________________________________________________________
dense_17 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_18 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 33
=================================================================
Total params: 31,665,153
Trainable params: 31,665,153
Non-trainable params: 0
```

Fig. 6: Shows the model layer architecture of prototype 5.

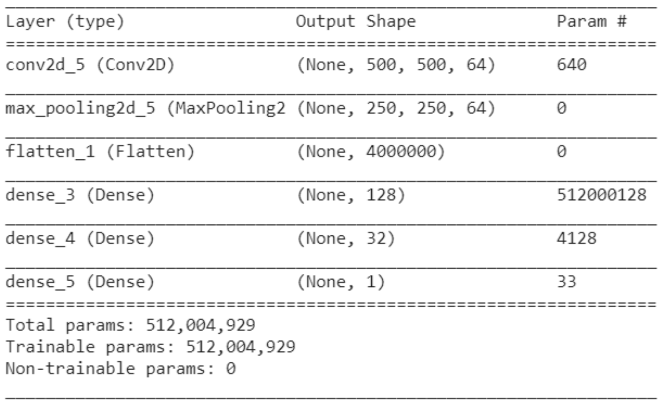
Hyperparameters	
Parameters	Values
Learning rate	0.001
Patience	2
Factor	0.3
Min learning rate	0.000001
Activation function	ReLU
Optimizer	Adam
Loss function	Binary Crossentropy
Epoch	25

TABLE IX: Hyperparameters of prototype 5.

Performance metrics	
Accuracy	88.785 %
F1 score Normal	0.86
F1 score Pneumonia	0.92

TABLE X: Performance metrics of prototype 5.

F. Prototype 6



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 500, 500, 64)      640
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 250, 250, 64)      0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4000000)           0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               512000128
_________________________________________________________________
dense_4 (Dense)              (None, 32)                4128
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 33
=================================================================
Total params: 512,004,929
Trainable params: 512,004,929
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7: Shows the model layer architecture of prototype 6.

Hyperparameters	
Parameters	Values
Learning rate	0.001
Patience	2
Factor	0.3
Min learning rate	0.000001
Activation function	Sigmoid
Optimizer	Adam
Loss function	Mean squared error
Epoch	25

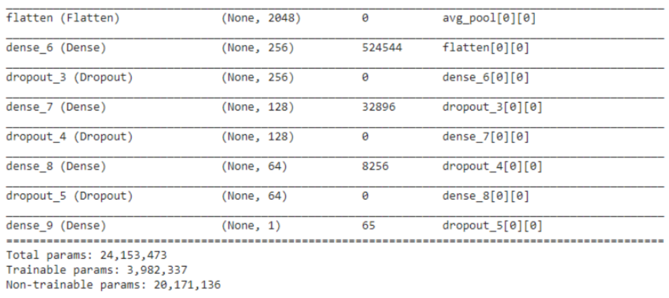
TABLE XI: Hyperparameters of prototype 6.

Performance metrics	
Accuracy	62.500 %
F1 score Normal	0.86
F1 score Pneumonia	0.92

TABLE XII: Performance metrics of prototype 6.

G. Prototype 7

This prototype used transfer learning, where feature extraction on ResNet-50 was implemented. The model was implemented without using learning rate reduction.



```
__________________________________________________________________________________________________
flatten (Flatten)               (None, 2048)         0           avg_pool[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 256)          524544      flatten[0][0]
__________________________________________________________________________________________________
dropout_3 (Dropout)             (None, 256)          0           dense_6[0][0]
__________________________________________________________________________________________________
dense_7 (Dense)                 (None, 128)          32896       dropout_3[0][0]
__________________________________________________________________________________________________
dropout_4 (Dropout)             (None, 128)          0           dense_7[0][0]
__________________________________________________________________________________________________
dense_8 (Dense)                 (None, 64)           8256        dropout_4[0][0]
__________________________________________________________________________________________________
dropout_5 (Dropout)             (None, 64)           0           dense_8[0][0]
__________________________________________________________________________________________________
dense_9 (Dense)                 (None, 1)            65          dropout_5[0][0]
==================================================================================================
Total params: 24,153,473
Trainable params: 3,982,337
Non-trainable params: 20,171,136
```

Fig. 8: Shows the model layer architecture of prototype 7.

Hyperparameters	
Parameters	Values
Learning rate	0.003
Activation function	ReLU
Optimizer	Adam
Loss function	Binary Crossentropy
Epoch	25

TABLE XIII: Hyperparameters of prototype 7.

Performance metrics	
Accuracy	62.607 %
F1 score Normal	0
F1 score Pneumonia	0.77

TABLE XIV: Performance metrics of prototype 7.

IV. DISCUSSION

As can be observed in the *results* many of the networks were able to achieve an accuracy of around 90%. An issue that was encountered however was to determine what exactly inhibited the different networks from achieving a higher accuracy. Both Fig 4 and Fig 3 indicates that the convolutional layers are not able to distinguish the two classes since no clusters are immediately evident. The fully connected layers are however able to consistently output correct results in just over 90% of the time.

An attempt to improve the network was to remove one max-pooling layer to preserve the features in the convolutional layers in prototype 5 in Fig. 6. This drastically increased the trainable parameters in the model and was more costly computationally but did not improve on the performance as can be seen in table X. Comparing the result of prototype 5 and 7, we can see that deeper convolutional networks with more trainable parameters does not yield a better performance than a simpler architecture.

It could be speculated that there exist a better model with an amount of trainable parameters in the range 1,925,889 - 31,665,153 which is the amount of trainable parameters for prototype 4 and 5 respectively. Such a model could eventually differentiate between the classes more efficiently and therefore aid the fully connected layers in making the correct predictions. If yet another model would be designed it could potentially benefit from having the same activation function, optimizer and loss function as prototype 4. That is due to the fact that Fig 5 indicates a less sporadic validation loss compared to Fig 2 and therefore maybe have a better potential for generalization.