Application of Convolutional Neural Networks in Pneumonia Prognosis Prediction

Gustavo Brito (r20170760@novaims.unl.pt), Marta Santos (r20170770@novaims.unl.pt),
Miguel Mateus (r20170752@novaims.unl.pt)

Pneumonia is a pulmonary disease that kills more than any other infectious disease, claiming the lives of over 800,000 children under five every year. In this scope, the sooner a diagnosis is made, the higher the likelihood of being cured. The main objective of this project is to build a model that can identify, from x-ray images, lungs affected by pneumonia. To accomplish this, a model using convolutional neural networks was developed and tested accordingly. Additionally, Grad-CAM was used to assess which parts of the x-rays were more important to the model's final decision. With the results attained, the model can be implemented in efforts to combat the child casualties caused by pneumonia.

1. Introduction

Acute respiratory infection (ARI) [Pneumonia] is a swelling (inflammation) of the tissue in one or both lungs. It's usually caused by a bacterial infection, but it can also be caused by a virus. Symptoms may include feeling tired, difficulty breathing, fever, chest pain and loss of appetite.

The task of identifying Pneumonia has become increasingly important since the beginning of the COVID-19 pandemic, as it is one of the most lethal diseases people can get with this virus.

The greatest incidence of this disease is in under-developed countries, which could highly benefit from having a model that could make a diagnosis, even without the presence of a doctor.

To build and train a model, 5856 chest X-ray images were used, each belonging to different children, including 4273 characterized as Pneumonia. The images had previously been screened for quality control and labelled as Normal, Bacterial Pneumonia and Viral Pneumonia by two experts in the field, to be used in the study Kermany et al, 2018. Unfortunately, access was only granted to the binary classification of Normal or Pneumonia.

The repository that contains the source code can be found on this link.

2. Methodology

2.1 Data Pre-Processing

To prepare the data for the project, first a new train, validation and test split was done, as the one originally specified by the data's source was not suitable, for example, less than 1% of data was left out for validation and the splits were not stratified by the output variable. As such, 70% (4101 images) of data were selected for training, 15% (878 images) for validation and the remaining 15% (877 images) for testing. The classes' unbalance of 73% for the majority class and 27% for the minority were preserved equally on all splits.

Due to the unbalanced classes in the dataset, three extra oversampling train partitions were created, to see if they would improve the final model's performance. To equalize both classes' proportions on the training sample, on one partition random oversampling was used, on another random under-sampling and on the last one, new images were created through data augmentation, on a random set of training images of the minority class.

All images were normalized to have pixel values between [0, 1] and reshaped to have 75x75 pixels and turned in to (75,75,1) tensors, to be able to be fed into the models. The pixel size was chosen due to computational reasons and a grayscale was used since the data are X-ray images.

2.2 Architectures

Before the creation of the models, some exploration was made into already existing architectures.

1. Lenet5: One the first CNN architectures, it is very simple to understand and build. Due to its simplicity, it can yield weak results when applied to more complex problems, although it has a fast training time.

2. Alexnet: Using the ReLu activation function, in combination with the GPU for processing, it can reach good results, both saving time training the network and achieving the same error rate decrease as other activation functions. It also uses data augmentation and dropout to prevent overfitting instances.

3.VGG16: Similarly to the previous network, it is also used in complex image recognition problems. It differentiates itself from AlexNet architecture by using a smaller kernel size in its filters and a greater number of convolutional layers.

Out of the three architectures, only Lenet5 was able to be fully tested, due to its computational simplicity. However, various characteristics of all three networks were taken as inspiration, when creating the first models.

2.3 Models

Approach

The models' creation, training and assessment were made **in two major phases**.

1. Several model architectures were trained and tested on the original train and validation splits. Tensorboard was used to monitor the models' training and to have a clearer view of what architectures worked best.
2. The hyperparameters of the two best model architectures were optimized using a grid search with k-fold cross-validation on the original training partition, using k=3. K-fold cross-validation was used to have more robust results.

This two-step approach was chosen due to time and computational constraints, otherwise, a full grid search, with k-fold cross-validation and larger k, of all architectures and hyperparameters would be preferred.

To better manage computational resources, the main techniques used were:

1. Call-backs (to stop the training process once the validation accuracy stopped improving for more than 3 epochs)
2. Maximum training time of 50 epochs
3. One hyperparameter optimized at a time, in the second phase.

Chosen Evaluation Metrics

Within the scope of our project, the consequences of making a wrong prediction vary greatly on the mistaken class. Wrongly classifying someone as not having pneumonia can have devastating effects. Therefore, accuracy alone as the main comparison metric between models would not be appropriate. Careful attention was given to recall[1], but it also should not be taken as the main evaluation metric.

To evaluate and compare each model, **the metric chosen was the f1-score with a micro average**, not only because it tackles the aforementioned problems, but also due to the imbalance of classes in the dataset.

Model Building

All initial architectures had some commonalities:

- Every layer but the output one used the Relu activation function.
- Pooling layers used Max Pooling with windows sized (2,2) with a stride of 2.
- Convolutional layers had filters sized (3,3) and stride 1 and each following layer had a higher number of filters than the last
- 1 neuron in the dense output layer with the sigmoid activation function
- Loss function: binary cross-entropy
- The optimizer was RMSprop with an LR of 0.0001
- Batch size of 20

These commonalities were chosen due to:

- Computational efficiency: Relu activation function and max pooling layers with a stride of 2.

- Success in the studied classic CNN architectures: convolutional filters with a

[1] It is not a very robust metric and can easily give deceiving expectations of a model's performance.

kernel size of (3,3) and stride of 1 and an increasingly higher number of filters.

- Nature of the problem (binary): sigmoid activation function, single neuron on the last dense layer and binary cross entropy loss function.

Several architectures varying in number of convolution and dense layers were tested. Pooling layers were experimented both after every convolutional layer and skipping every second layer. When overfitting occurred, the combination of techniques used to mitigate it were:

- **Data augmentation**: rotation with a range of 40, width shift with a range of 0.2, height shift with a range of 0.2, shear transformations with a range of 0.2 and zoom with a range of 0.2. No horizontal flips were applied as the x-ray images were all facing the chest area of the patients, making it unwise to train the model with "reversed" images.
- **Dropout** of 0.5 applied to every hidden dense layer.
- L1 and/or L2 **weight regularization** of 0.001.

The results of using these techniques to reduce overfitting will be discussed in section 3.1.

The two best model architectures were chosen for hyperparameter tuning, as they had similar results, f1-score of about 0.975, and were as follows:

Model 1.1	Model 2.1
Conv2D(3,3,16)	Conv2D(3,3,16)
Conv2D(32)	MaxPooling2D
MaxPooling2D	Conv2D(3,3,32)
Conv2D(3,3,64)	MaxPooling2D
Conv2D(3,3,128)	Conv2D(3,3,64)
MaxPooling2D	MaxPooling2D
Dense(256)	Conv2D(3,3,128)
Dropout(0.5)	MaxPooling2D
Dense(1)	Dense(256)
	Dense(128)
	Dense(1)

Table 1 - Initial model architectures

To optimize the hyperparameters of the chosen architectures, as discussed before, several grid searches with k-fold cross-validation (k=3) were performed and the f1-score was used to evaluate all models.

A single grid search was used for each hyperparameter, e.g., first, a grid search with different numbers of filters and neurons was tested, then one with different batch sizes. From one grid search to the other, the hyperparameters were changed according to the previously found as being the best. E.g., the grid search exploring different batch sizes used the number of filters and neurons found to be best by the previous grid search.

Since only one hyperparameter was able to be optimized at a time, the order in which they were optimized was of great importance. As such, they were chosen in order of the perceived impact they would have on the performance of the model.

First, 3 configurations of different numbers of filters and neurons of each architecture were tested.

	Model 1	Model 2
#1		
Conv2D #1	16 filters	16 filters
Conv2D #2	32 filters	32 filters
Conv2D #3	64 filters	64 filters
Conv2D #4	128 filters	128 filters
Dense #1	256 Neurons	256 Neurons
Dense #2	1 Neuron	128 Neurons
Dense#3	-	1 Neuron
#2		
Conv2D #1	16 filters	16 filters
Conv2D #2	32 filters	32 filters
Conv2D #3	32 filters	64 filters
Conv2D #4	64 filters	128 filters
Dense #1	128 Neurons	512 Neurons
Dense #2	1 Neuron	256 Neurons
Dense#3	-	1 Neuron
#3		
Conv2D #1	16 filters	16 filters
Conv2D #2	32 filters	32 filters
Conv2D #3	64 filters	64 filters
Conv2D #4	64 filters	128 filters
Dense #1	256 Neurons	128 Neurons
Dense #2	1 Neuron	64 Neurons
Dense #3	-	1 Neuron

Table 2 - Combinations of filters and neurons tested

Then, for each architecture, 4 models with batch sizes of 20, 32, 100 and 200 were tested, following 2 models with either max pooling or

average pooling and 2 models with either all convolutional windows of size (3,3) or (5,5).

In total, 22 models were tested, and the best results were with the following two:

Model 1.2	Model 2.2
Batch size: 20	Batch size: 100
Conv2D(3,3,16)	Conv2D(3,3,16)
Conv2D(3,3,32)	MaxPooling2D
MaxPooling2D	Conv2D(3,3,32)
Conv2D(3,3,32)	MaxPooling2D
Conv2D(3,3,64)	Conv2D(3,3,64)
MaxPooling2D	MaxPooling2D
Dense(128)	Conv2D(3,3,128)
Dropout(0.5)	MaxPooling2D
Dense(1)	Dense(512)
	Dense(256)
	Dense(1)

Table 3 - Best Hyperparameters found

After the best hyperparameters were found, both models were retrained for 50 epochs, on the whole training partition, to determine the optimal number of epochs. This was done through the analysis of the f1-score, accuracy, and loss of the validation partition. The results of these models can be found in section 3.3.

Even though on the final results the class imbalance wasn't an issue, the best model was still trained on the 3 different generated training samples developed in the beginning of the project. Call-backs were used, in the same structure as before and f1-score was still the metric chosen, as the validation data remained untouched. The results can be found on section 3.4.

3. Results

3.1. Mitigating Overfitting

When it came to mitigating some overfitting in the initial architectures, only dropout provided good results.

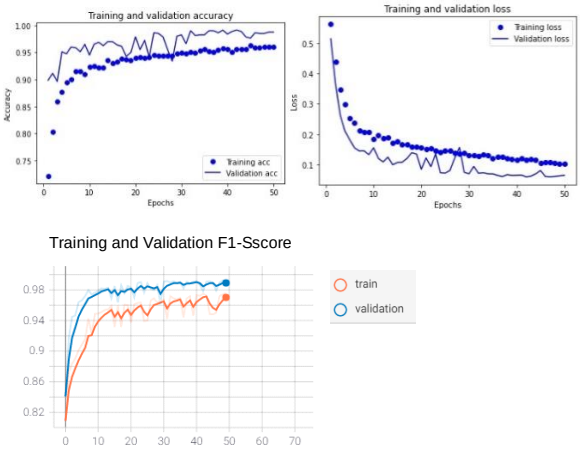
Both weight regularization techniques (alone or in combination with other methods) were not enough to fully prevent overfitting, at least for a considerable number of epochs to justify the added complexity to the model.

The use of data augmentation made the validation metrics very inconsistent during the training process. This erratic behaviour is to be expected, as the x-ray images are standardized so the chest of the patients is centred. As such, any transformation to a training image will likely generate a datapoint very different from the standard.

3.2. Hyperparameter tuning remarks

After several hyperparameters were optimized, the f1-score improved by only approximately 0.05. This relates to the fact that a full grid search of all possible hyperparameter combinations was not done. Instead, a greedy approach was taken, which might have not returned the best possible solution.

3.3. Validation Partition Results



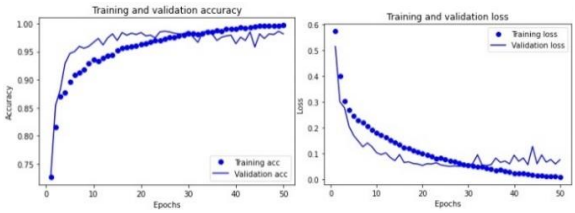
Figures 1,2 and 3 - Model 1.2 learning curves

Model 1.2 could be trained longer, as seen by the f1-score, accuracy and loss of the validation set, where no signs of overfitting are shown. The model hasn't yet converged, which is evidence of underfitting and that more training is needed to fully take advantage of it. Unfortunately, due to time constraints, it was not possible to retrain model 1.2 further and fully test its potential.

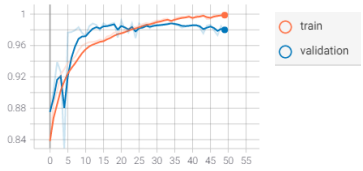
	Precision	Recall	f1-Score
Normal	0.99	0.97	0.98
Pneumonia	0.99	1	0.99

Accuracy	0.99
f1-score	0.998

Table 4 - Model 1.2 validation results (50 epochs of training)



Training and Validation F1-Sscore



Figures 4,5 and 6 - Model 2.2 learning curves

Model 2.2 converged during the 50 epochs of training. The validation scores for all metrics analysed equalled the training ones at about epoch 30, after which some overfitting was present. However, the validation scores didn't get worse during the last 20 epochs They seemed to remain constant or even improve at a very slow rate. This indicates that 30 epochs might not necessarily be the best choice. As such, the f1-score, accuracy, recall and precision will be shown for both 30 and 50 epochs of training.

	Precision	Recall	f1-score
Normal	0.92	0.99	0.95
Pneumonia	1	0.97	0.98

Accuracy	0.97
f1-score	0.973

Table 5 - Model 2.2 validation results (30 epochs of training)

	Precision	Recall	f1-score
Normal	0.97	0.96	0.97
Pneumonia	0.99	0.99	0.99

Accuracy	0.98
f1-score	0.985

Table 6 - Model 2.2 validation results (50 epochs of training)

The extra 20 epochs of model 2.2 training improved the f1-score and accuracy. They also improved the recall of the pneumonia class, however at the cost of the recall of the normal class. This trade-off is considered advantageous, as it shows a model less prone to wrongly predicting as normal an individual who has pneumonia and improves the overall f1-score and accuracy.

Even though model 1.2 shows better scores for all metrics, due to it not converging during training these scores might not be very robust. During training, the model showed that it still hadn't captured all relevant patterns present in the training data. As such**, model 2.2, trained for 50 epochs, was chosen as the fittest to solve the problem**.

3.4. Classes Imbalance

3.4.1. Random Undersampling

Random undersampling produced a f1-score and a training behaviour very similar to that of the original training of model 2.2. Unquestionably, because the only difference with the original training partition is the almost 2000 images of the majority class that were randomly removed. Validation loss behaviour was slightly less stable than with the original training split, and the f1-score lower.

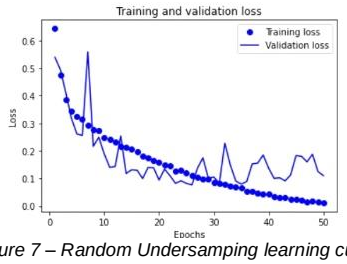


Figure 7 – Random Undersamping learning curve

3.4.2. Random Oversampling

Random oversampling training was stopped at 10 epochs, the consequences of duplicating datapoints were shown and the model overfitted from that point onwards. Just like random undersampling, the validation loss during training is unstable and an even lower f1-score was presented.

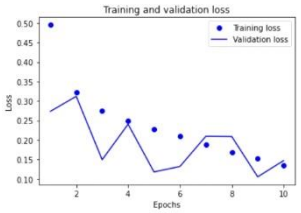


Figure 8 – Random Oversampling learning curve

3.4.3. Data Augmentation to balance classes

The training was stopped at 15 epochs, since after that the model starts to overfit. During

training, the validation loss wasn't erratic, as with the previous methods, but still, no further improvement was made, likely due to the same constraints pointed out before about data augmentation.

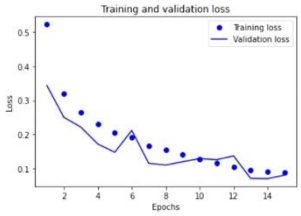


Figure 9 – Data Augmentation to balance classes learning curve

3.5. Classes Imbalance Remarks

No oversampling technique tested improved the prediction capabilities of the model, which is not surprising, seeing as already very good results were obtained with an unbalanced dataset and the techniques tried were simplistic and had some known disadvantages.

Technique	Validation F1-Score
Random Undersampling	0.974
Random Oversampling	0.943
Data Augmentation to balance classes	0.969

Table 7 – Results of techniques to battle class imbalance

3.6. Test Partition Results

Model 2.2, trained on the original splits, had the following results on the test partition:

	Precision	Recall	f1-score
Normal	0.93	0.86	0.89
Pneumonia	0.95	0.98	0.96

Accuracy	0.95
f1-score	0.945

Table 8 and 9 – Model 2.2 test results

Both the f1-score and the accuracy were lower than the ones of the validation partition, 4 and 3 percentage points, respectively. Recall for the minority class was 10 percentage points lower than what it was with the validation partition.

Nonetheless, good results were achieved on unseen data. F1-score and accuracy were approximately 0.94 for both, recall and precision for the pneumonia class were 0.98 and 0.95 respectively and an AUC of 0.98 was achieved.

In Kermany et al. 2018, the authors achieved a test set accuracy of 0.928, sensitivity of 0.932, specificity of 0.901 and AUC of 0.968. However, the data was partitioned differently and therefore direct comparisons between the scores are not advisable.

3.1 Grad-CAM

To have a better understanding of the model's behaviour, Grad-CAM was applied to a few images of the test set. In most cases, when the model's prediction was correct, the model is very specific on what part of the image it is basing its prediction on. In wrong predictions however, the model seems to be "looking" everywhere, not focusing on key areas of the image, or focusing on a meaningless part (as shown in the bottom-left image, where the bottom-left part of the image is highlighted as red). Perhaps, in future work with this data, some transformations, like cropping, might be done to better isolate the lung area of the images during training.

		Predicted Class	
		Normal	Pneumonia
Actual Class	Normal		
	Pneumonia		

Table 10 – Confusion Matrix of Grad-CAM images

3.2 Classification with non-MLP models

Inspired by the work of Janke et al, 2019, some classification algorithms, other than MLP's, were tested on top of the pre-trained

convolutional layers of model 2.2. The models received as training and validation input high-level image features of the training and validation data respectively, produced from the last convolutional layer of model 2.2. The models' tested were AdaBoost, Random Forest and K-Nearest Neighbour classifiers. To optimize each models' hyperparameters, a grid search with k-fold cross-validation (k=10) of the training data was performed. The model with the highest f1-score was chosen.

- 9 AdaBoost classifiers were tested, using 10, 50 or 100 estimators and a learning rate of 1, 0.1 or 0.001.
- 3 K-Nearest Neighbour classifiers were tested, using 1, 5 or 10 neighbours. All of them used the Euclidean distance.
- 4 Random Forest classifiers were tested, using 10, 25, 50 or 100 estimators. All having the square root as the maximum number of features and Gini as the diversion metric.

Model	Validation F1-Score
Adaboost	0.970
KNN	0.970
Random Forest	0.973

Table 11 – Summary F1-Scores of Non-MLP Models

The models had similar scores, random forest outperforming the other ones. Although they were not able to reach the f1-score provided by model 2.2, they were close.

4. Conclusions

The main objective of this project was to train a model that would be able to classify chest x-ray images.

The predictions of the final assessed model allowed us to correctly identify around 95% of the x-rays.

For many of us pneumonia has not been an issue for many years now, however in underdeveloped countries it is still an ongoing battle. We believe we have created a robust model with real-life application that could potentially be used around the world to increase children's lifespan.

Even though the hardest part of this process would be to provide the technological resources (x-ray machines and high-end computers), we believe that when the technology is present this model will be of great interest to help diagnose an easily treatable disease.

5. References

Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification”, Mendeley Data, V2, doi: 10.17632/rscbjbr9sj.2

http://dx.doi.org/10.17632/rscbjbr9sj.2

Jonathan Janke, Mauro Castelli, Aleš Popovič,

Analysis of the proficiency of fully connected neural networks in the process of classifying digital images. Benchmark of different classification algorithms on high-level image features from convolutional layers,
Expert Systems with Applications,
Volume 135,
2019,
Pages 12-38,
ISSN 0957-4174,
https://doi.org/10.1016/j.eswa.2019.05.058

Chest X-Ray Images (Pneumonia). (n.d.). Retrieved 3 April 2021, from https://kaggle.com/paultimothymooney/chest-xray-pneumonia

Goel, A. (2020, June 4). How to add user defined function (Get F1-score) in Keras metrics ? Medium. https://aakashgoel12.medium.com/how-to-add-user-defined-function-get-f1-score-in-keras-metrics-3013f979ce0d

Grad-CAM class activation visualization. (n.d.). Retrieved 3 April 2021, from https://keras.io/examples/vision/grad_cam/

Keras: Feature extraction on large datasets with Deep Learning. (2019, May 27). PyImageSearch.

https://www.pyimagesearch.com/2019/05/27/keras-feature-extraction-on-large-datasets-with-deep-learning/

Posted by Muhammad Rizwan on October 16, 2018 at 4:33pm, & Blog, V. (n.d.). LeNet-5—A Classic CNN Architecture. Retrieved 3 April 2021, from https://www.datasciencecentral.com/profiles/blogs/lenet-5-a-classic-cnn-architecture

Sakib, S. (2021). SadmanSakib93/Stratified-k-fold-cross-validation-Image-classification-keras [Python]. https://github.com/SadmanSakib93/Stratified-k-fold-cross-validation-Image-classification-keras
(Original work published 2020)

Thakur, R. (2020, November 24). Step by step VGG16 implementation in Keras for beginners. Medium. https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

Wei, J. (2020, September 25). AlexNet: The Architecture that Challenged CNNs. Medium. https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951