Chest X-ray classification using Keras

Presentation for the data science peers.

Advanced Data Science Capstone Project.

Paolo Cavadini, February 2021

Architectural choices

Jupyter notebook (it runs locally in my MAC and I shared under IBM Watson Studio).

Keras to extract images, pre-processing, modeling, training, and evaluation.

Sci-kit learn to model, train and evaluate a Logistics Regression which I used as baseline.

Matplotlib and Seaborn to render the images and the data.

NumPy to manipulate the arrays.

The images have been already phased for AI. I rendered the corresponding arrays in 2D and 3D.

I came up with 5.856 images, 1583 normal, 4273 with pneumonia, size (150,150,3) and the corresponding class labels (0,1).

I normalized the features and casted as float.

I used weights to balance the volume of images with and without pneumonia.

I verified the impact of data augmentation (i.e. artificially flip and rotate the images to increase them and present in different positions).

Data quality assessment, pre-processing, feature engineering

Model performance

I looped fitting the model for 1 epoch over a selection of 3 optimizers and 2 activation functions to make the best choice in terms of accuracy.

Next, I fit the model for 1 epoch using feature weights and I evaluated the performance to determine that they improve the accuracy. Indeed feature weights correct the balance between number of images with and without pneumonia.

I used data augmentation to increase the number of images by means of flipping and rotating them. This helps to bring the model closer to reality.

I finally run training using 10 epochs and callback functions, and I found an accuracy above 95%.

I plotted the validation curves, loss and accuracy, and a confusion matrix.

Model algorithm

I have implemented a Deep Learning neural network using Keras, composed of 5 blocks with convolution, maxpooling and batch-normalization. On top of it a flatten layer is followed by two dense layers, and in between a dropout layer to reduce over-fitting.

I used the input shape (150,150,3) and a batch size of 32 which is a good compromise between accuracy and response time. The activation function and optimizer have been searched trough a fine tuning process. I chose the binary crossentropy loss function because it is a common choice for classification, and the accuracy metric because it is easy to interpret.

I run 1 epoch during fine tuning and 10 epochs during the final training.

I used EarlyStopping callback to stop training when the difference between training and validation error starts to increase, instead of decreasing (overfitting).

I used ReduceLROnPlateau to manage the optimization of the learning rate.

Thank you!

Acknowledgements:

Data: https://data.mendeley.com/datasets/rscbjbr9sj/2

Citation: http://www.cell.com/cell/fulltext/S0092-8674(18)30154-5