# Chest X-ray classification using Keras

Presentation for the data science peers.

Advanced Data Science Capstone Project.

Paolo Cavadini, February 2021

## Architectural Choices/1

Jupyter notebook , local/shared (IBM Watson Studio):
Chest_x_ray_DNN.model_deployment.jupyter.003.ipynb
https://eu-gb.dataplatform.cloud.ibm.com/analytics/notebooks/v2/c6da816d-7403-4d4f-bf55-6c84d674a07e/view?access_token=66b321b8edf6f5b5af346a61757abd4375c698964c38f17d6b9e96a037319d4c

- GitHub (https://github.com/pcavad/capstone_x_rays )

chest_x_ray_DNN.data_exp.jupyter.002.ipynb
chest_x_ray_DNN.data_etl.jupyter.002.ipynb
chest_x_ray_DNN.feature_eng.jupyter.002.ipynb
chest_x_ray_DNN.model_def.jupyter.002.ipynb
chest_x_ray_DNN.model_train.jupyter.002.ipynb
chest_x_ray_DNN.model_evaluate.jupyter.002.ipynb
adsMod.py (methods):
. modeling: add layers to the neural network, compile, save.
. get_path: get image paths.
. plot_validation_curves: plot the validation curves after training.
. plot_confusion_matrix: plot a confusion matrix.

## Architectural Choices/2

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

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

Matplotlib and Seaborn: render the images and the data.

NumPy: manipulate the arrays.

## Data quality assessment, pre-processing, feature engineering

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

5.856 images, 1583 normal, 4273 with pneumonia, size (150,150,3) , class labels (0=normal, 1=pneumonia).

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).

# Model performance

I looped fitting the model for 1 epoch :
- activation functions: 'relu', 'sigmoid', 'tanh'
- optimizers: 'adam', 'rmsprop'
-> best accuracy: 'relu', 'rmsprop'.

Next, I fit the model for 1 epoch using feature weights:
- initial bias: 0.9964 Weight for class 0: 1.85 Weight for class 1: 0.68
-> accuracy improved (90.55% to 92.79%).

I used data augmentation to increase the number of images (flip, rotate). This helps to bring the model closer to reality.

I finally run training using 10 epochs and callback functions, and I found an accuracy exceeding 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:
- 5 blocks with convolution, maxpooling and batch-normalization
- 1 flatten layer followed by two dense layers
- 1 dropout layer in between to reduce over-fitting.

- Input shape (150,150,3)
- batch size of 32 (good compromise between accuracy and response time)
- Fine tuning of the activation function and the optimizer
- binary crossentropy loss function (common choice for classification)
- accuracy metric (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