Using Convolutional Neural Network for Chest X-ray Image classification

Matija Sorić [1,2], Danijela Pongrac [1], Iñaki Inza [2]
[1] Zagreb University of Applied Sciences, Zagreb, Croatia
[2] University of the Basque Country, Donostia-San Sebastián, Spain
msoric@tvz.hr

***Abstract* - Chest X-ray is an imaging technique that plays an important role in pneumonia diagnosis. Owing to the high availability of medically-oriented image datasets, great success can be achieved using convolutional neural networks (CNNs) in the recognition and classification of these images. Since previous research has shown CNNs to perform as well as the best clinicians in diagnostic tasks, they caused great excitement among researchers. In this paper, convolutional neural network (CNN) machine learning (ML) model was built using a supervised dataset. The dataset used contained both pneumonia and non-pneumonia images, which the model had to classify correctly. In the end, the model is demonstrated to have achieved satisfactory results, with the high accuracy of 90.38%, 98.21% recall and 87.84% precision.**

Keywords – convolutional neural network, classification, deep learning, X-ray imaging

I. INTRODUCTION

Due to the rapid development of the X-ray imaging technique, the physiological state of tissues in the human body can now be examined noninvasively. Since every imaging examination should always begin with conventional radiography, X-ray imaging has been the main diagnostic method for diagnosing pneumonia since its inception.

The growing availability of electronic medical data is a great opportunity to discover and develop new healthcare-related technologies. However, as standard systems and programs are incapable of handling such large datasets, new computational techniques are required for data scientists to use this data. This is where artificial intelligence (AI) and machine learning can be of assistance.

Image classification is one of the most challenging problems in healthcare. It essentially aims to classify medical images into different categories which would help physicians with diagnosis and further examinations. Classification is usually performed by extracting certain features from the image and classifying images based on those features.

Until now, physicians have relied on their professional experience to extract important features - usually a laborious and time-consuming task. In that respect, *deep learning* (DL) has become one of the hottest research areas in computer science and computer applications. [1]

Hence, the goal of this study was to evaluate the performance of a *neural network* (NN) using X-ray images, and most importantly, to determine whether a patient has pneumonia or not based on the images fed to the model. For this concept to be tested, 5856 X-ray images were used, previously graded by expert physicians. The results presented in the following sections, demonstrate the proof-of-concept principle using DL method for radiological image feature extraction, which can later be used for x-ray image classification.

II. DATASET ANALYSIS

Images used in this project were taken from the *Kaggle* website. Kaggle is an online community of data scientists, where users can download or publish datasets, as well as build ML models in a web-based environment and collaborate with other data scientists. [2]

Dataset is divided into 3 sets: *'train'*, *'test'* and *'validation'*. Each of these sets contains subfolders Pneumonia/Normal. Dataset structure is illustrated in Figure 1.

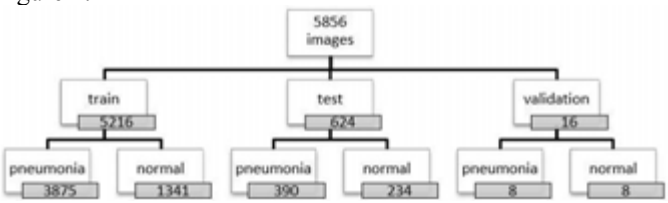

Figure 1. Dataset structure

It needs to be emphasized that patients whose images were used in the dataset are children aged 1-5. Images were taken in the Guangzhou Women and Children's Medical Center in China. Prior to being included in the Kaggle dataset, images were preprocessed, quality screened and all low quality scans removed. Furthermore, they were graded by expert physicians before taking them into account for the Kaggle upload.

Folder 'train' has the size of 1.07 GB, and, as visible in the Figure 1., 1341 and 3875 images in the subfolders 'PNEUMONIA' and 'NORMAL', respectively. Folder 'test' has the size of 75.3 MB, with 234 images in folder 'NORMAL' and 390 images in folder 'PNEUMONIA'. Folder 'val' is the smallest, having the size of 2.88 MB,

containing 8 images in the folder 'NORMAL' and 8 images in the folder 'PNEUMONIA'. As these statistics indicate that the dataset structure did not correspond to the recommended 60%-20%-20% ratio of training-test-validation data, further dataset balancing could presumably improve evaluation metrics. Likewise, the dataset was skewed since there are considerably more pneumonia than normal lung images. The ratio is approximately 72% to 28% in favor of pneumonia images. A 50% to 50% ratio would, in all likelihood, provide a better model.

III. TECHNOLOGY BEHIND MACHINE LEARNING

ML algorithms can be divided into two categories: unsupervised and supervised. Unsupervised learning is known for feature extraction, and supervised learning builds the relationship between input (e.g. patient traits) and output (outcome of interest), and is used in predictive modelling. There are also other types like semi-supervised and weakly-supervised learning. [3]

The model used in this project was built using supervised learning that requires „training" data to be labeled (tagged with the correct answer). E.g. if there is known set of columns with a new column to be predicted, a reference column is needed, for comparing predicted values with the new column values.

Moreover, the model is built using the CNN, since it is the most commonly used algorithm today for computer vision, combining high performance and execution speed. Figure 2 shows the comparison between various ML algorithms used in computer vision.

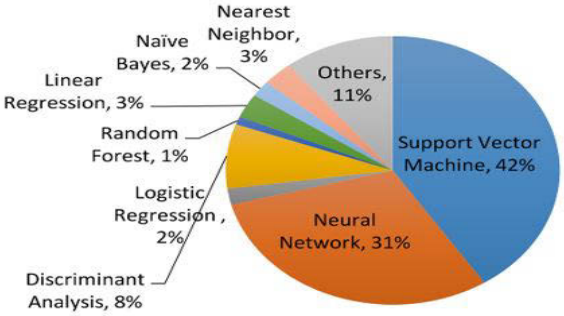

Figure 2. The most popular ML algorithms used for medical purposes [3]

A. Neural Network architecture

NN is actually a set of algorithms, operating similarly to the human brain (emulating brain's cognitive, i.e. pattern-recognition skills). Just like the human brain interprets sensory data, NN functions like a black box, taking inputs like x-ray images and processing them into one of the multiple outputs, in this case, classes. The inputs taken are numerical, contained in vectors, into which all data (e.g. images, sound, text, etc.) entering an NN must be translated. NN has numerous small units called neurons, grouped into layers mutually connected by neurons. NN architecture is depicted in Figure 3, where the first layer is the input layer which takes inputs such as images, hidden layers (all the layers between input and output layer) where computation is done, and the output layer that produces the result for an input:

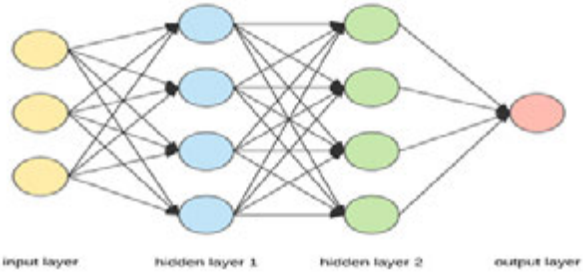

Figure 3. Neural network architecture [4]

Neurons are connected with neurons from other layers through weighted connections. As every weighted connection has a real-valued number attached, the neuron actually takes the value of the connected neuron and multiplies it with the connections weight. That sum of all connected neurons is called neurons' bias value. Bias value is mathematically transformed by putting it through the activation function, which is then assigned to the connected neuron in the connected layer. Activation function 'decides' whether a neuron should be activated or not (based on its weighted sum), thus introducing non-linearity. [5]

Operations performed by neurons are presented in Figure 4:

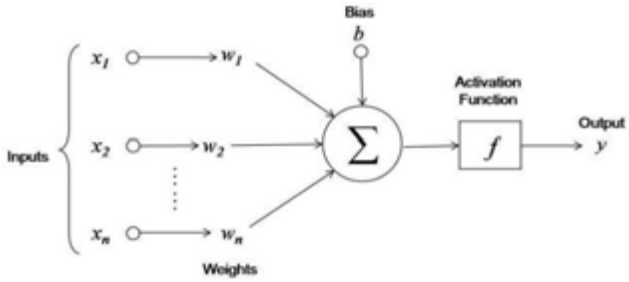

Figure 4. Operation performed by neurons [6]

In this respect, NN can be thought of as a filter which calculates possibilities, and the challenge is to obtain the correct neuron values (weights) that would allow the correct results to be computed. These weights are found by *optimization functions*.

B. Convolutional Neural Network architecture

Deep Learning (DL) is a subfield of ML based on learning multiple levels of representations by making a hierarchy of features where the higher levels are defined from the lower levels and the same lower level features can help in defining many higher level features. [7]

In the same manner, *Deep Neural Network* (DNN) is a variation of an NN with more than one hidden layer. Each of these hidden layers trains on a distinct set of features based on the previous layer's output. The more hidden layers there are in the DNN, the more complex are the features the nodes (neurons) can recognize, thus creating a better network.

One of the best DL algorithms that continuously proves its ability of diagnosing medical images with substantial performance is CNN. Its main characteristic is

its ability to learn features automatically from domain-specific images (unlike the classical ML methods). [8]

For example, a CNN was used to detect patients suffering from COVID-19, using computorized tomography (CT) images. [9]

CNNs are a class of DNNs, and are designed specifically for computer vision. Their name is derived from convolutional layers. The goal of the CNN is to shrink images into a more easily processed form, without losing the essential features critical for making predictions. The aforementioned convolutional layer is the main building block of the CNN. Although its parameters include learnable filters with a small receptive field (similar to the human visual cortex neurons), they span full input volume depth. [10]

This process is best illustrated if one imagines a flashlight illuminating only a part of an image, like in Figure 5. It is important to note that for a computer, an image is nothing more than a pixel of values:

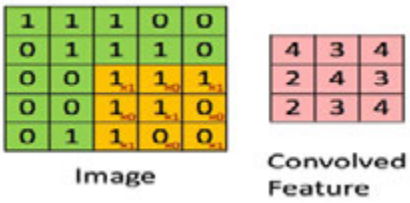

Figure 5. Convolution operation [11]

Therefore, input image is represented by a 5x5 matrix on the left. Our kernel/filter is a 3x3 matrix (flashlight) that shifts along the input matrix (stride value) and produces a new matrix on the right. Convolutional filters can be abstractly described as feature identifiers, with features represented by edges, colors, curves, etc. When an image is too large, a pooling layer is used which essentially reduces parameter number, and is represented in Figure 6:

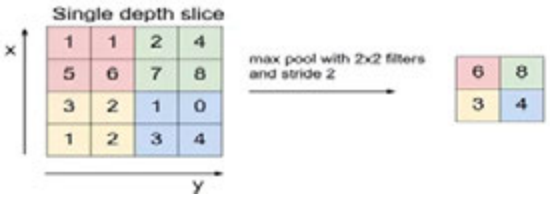

Figure 6. Max pooling [11]

Another important layer is ReLU layer, using an activation map to set negative values to zero (or another number, depending on ReLU type), thus increasing the nonlinear properties of the decision function. [12]

Hence, at first, convolutional layers detect edges and curves, but for the network to be able to detect higher level features (such as eyes or ears - if a human face has to be detected), a *fully-connected* (FC) layer is needed. It takes the inputs and outputs of an N-dimensional vector, where N is the number of classes to be predicted (in this case two). Consequently, the FC layer determines which features correlate the most with a particular class. A classic CNN architecture would look like this:

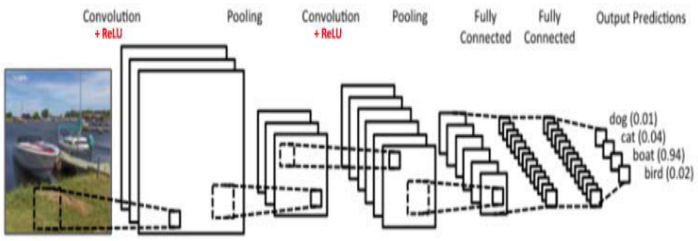

Figure 7. CNN architecture [13]

Finally, there is an FC output layer which makes the final prediction, as depicted in the Figure 7. [13]

IV. BUILDING THE MODEL

Our program code for this project, written in Python using Jupyter Notebook environment, which will be described here, can be found on *Github*. [14]

Model building commences with the libraries importation. Table I describes the standard libraries used.

TABLE I. IMPORTED STANDARD LIBRARIES

Library	Description
os	OS operations: reading files, changing or making a directory, etc.
random	generates random numbers
matplotlib	library for creating visualizations
cv2	used for reading, displaying images, saving
pandas	used for data analysis and modeling
numpy	support for arrays and matrices

Then ML libraries like Keras and Tensorflow were imported.

- TensorFlow - library for DL model production.
- Keras - an API built on TensorFlow, used to build and test NNs.

As for image preprocessing, images are resized and casted to float datatype for normalization, labels (0 and 1) are added subsequently and added to an array. Figure 8. illustrates this process (pseudo-code):

```
test_data   = []
test_labels = []

for cond in ['NORMAL', 'PNEUMONIA']:
  for img in path:
    img = read(path + 'test' + cond + img)
    img = resize(img, dimensions)
    img = cast(float)

    if cond == '/NORMAL/':
      label = 0
    elif cond == '/PNEUMONIA/':
      label = 1
    test_labels.append(label)
    test_data.append(img)

test_data = np.array(test_data)
test_labels = np.array(test_labels)
```

Figure 8. Image preprocessing (pseudocode)

The data is augmented with ImageDataGenerator class, usually used for artificial dataset expansion and the improvement of a model's ability to generalize what it has learned to new images. This is achieved through the use of image rescaling, flipping and viewing at a different magnification scale (zoom).

Following preprocessing and data augmentation, an input function is required to return the tensor, i.e. to instantiate the Keras tensor. In Table II, convolutional layers and pooling attributes are explained.

TABLE II. IMPORTED CONVOLUTIONAL LAYERS

Function	Description
conv2D	creates 2D kernel convolved with input to produce an output. It is a traditional convolution - filter sliding through the image
SeparableConv2D	variation for faster computing
MaxPool2D	used to down-sample an input (image) and reduce dimensionality.

Figure 9. shows the first convolutional layer:

```
model = Sequential()
model.add(Conv2D(16, kernel_size=(3, 3), padding = 'same', input_shape=(150,150,3)))
model.add(Conv2D(16, kernel_size=(3, 3), padding = 'same'))
model.add(MaxPool2D(pool_size=(2, 2)))
```

Figure 9. Instantiating the first convolutional layer

Another four convolutional layers were used, as well as functions that enable the use of higher learning rates; which in return accelerates the learning process. This architecture is in many respects similar to the architecture advocated by Yann LeCun in the 1990s for his image classification model (with the exception of LeakyReLU).

Activation function used was *PReLU* (*Parametric ReLU*). As previously mentioned, activation functions are an important NN feature. They decide whether a neuron should be activated. 'Activation' is used to determine an output (like yes/no). ReLU is the most commonly used activation function, especially in CNNs. Since ReLU is linear for all positive values and zero for all negative values:

- model training is computationally cheaper
- convergence is accelerated

By contrast, Leaky ReLU has a small slope for negative values, instead of zero, which accelerates the training process. Although Parametric ReLU (PReLU) is a variant of LeakyReLu, instead of having a predetermined alpha value of e.g. 0.3, it is learned automatically by NN. The difference between ReLU, LeakyReLU and PReLU is illustrated in Figure 10, where ReLU (presented as a blue line) follows the y=0 axis, LeakyReLU is depicted as a red line, and PreLU as a yellow line.

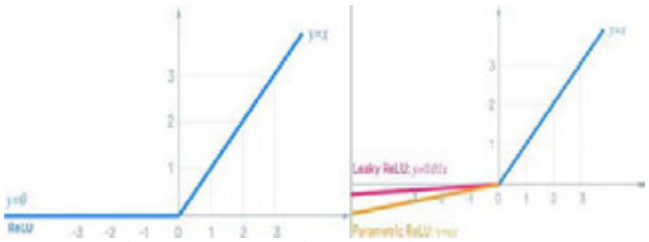

Figure 10. ReLU, LeakyReLU and PreLU [15]

Adam, an adaptive learning rate optimization algorithm designed specifically for training DNNs, was used as an optimization function for identifying the best possible weights. [16]

The Keras library implements numerous functions for creating and performing various operations with the ML model. Core layers like Dense, Flatten and Dropout are imported, and explained in Table III.

TABLE III. IMPORTED CORE LAYERS

Layers	Description
Dense	Used when association exists between features. Feeds all outputs from the previous layer to all of its neurons - each neuron provides one output to the next layer
Flatten	Converts pooled feature map to a single column that is passed to the FC layer. Flattening removes all but one dimension. Later it is passed to the Dense layer.
Dropout	A regularization technique, random neurons are ignored (dropped-out) during training. When a neuron is dropped, another takes its place, reducing the network's sensitivity to neuron weight and improving generalization

BatchNormalization, which accelerates the learning process by maintaining the activation standard deviation close to 1 and the mean activation close to 0, is also used. Pixel intensity in grayscale images (used in the project) varies from 0 to 255. Prior to inputting an image to NN, every image is transformed to an array (1D), hence every pixel enters one neuron from the input layer. If each neuron output is passed to a function, every value other than 0 (i.e. 1 to 255) will be reduced to a number between 0 and 1. Normalizing pixel values before training is common practice. [17]

The FC layer is the last layer in the model. Overfitting was reduced by regularization, i.e. *L2 regularization*, which forces the weights to take smaller values. In the output layer, activation is set to *'sigmoid'*, because the problem is a binary (two-class) classification. Sigmoid activation function exists only in the area between 0 and 1 and is therefore suitable for binary classifications.

Upon its completion, the model is compiled and run using the *model.compile( )* function. Before that, certain functions are applied during the training, called callbacks. *Callback* is a class called at various training stages. It interacts with the model during the training process. Two attributes of this class are *ModelCheckpoint*, which saves the best model obtained during training, and *ReduceLROnPlateau* which monitors a metric and reduces the learning rate if the metric stops improving. Callbacks are used because learning rate reduction is frequently beneficial for ML models.

V. RESULT EVALUATION

Once the model is built, prediction model training takes place, shown in Figure 11, with some additional arguments set, e.g. the number of training epochs, etc.

```
hist = model.fit_generator(train_gen,
                           steps_per_epoch = train_gen.samples // batch_size,
                           epochs = epochs,
                           validation_data = test_gen,
                           validation_steps = test_gen.samples // batch_size,
                           callbacks = [checkpoint, lr_reduce, early_stop, tensor_board])
```

```
Epoch 1/10
163/163 [==============================] - 209s 1s/step - loss: 0.3954 - accuracy: 0.8115 - v
al_loss: 0.7950 - val_accuracy: 0.6234
Epoch 2/10
163/163 [==============================] - 204s 1s/step - loss: 0.2953 - accuracy: 0.8763 - v
al_loss: 1.1236 - val_accuracy: 0.6334
```

Figure 11. Model training

The results were evaluated using two graphs presented in Figure 12. The graphs show the accuracy and loss of the model over a fixed number of epochs.

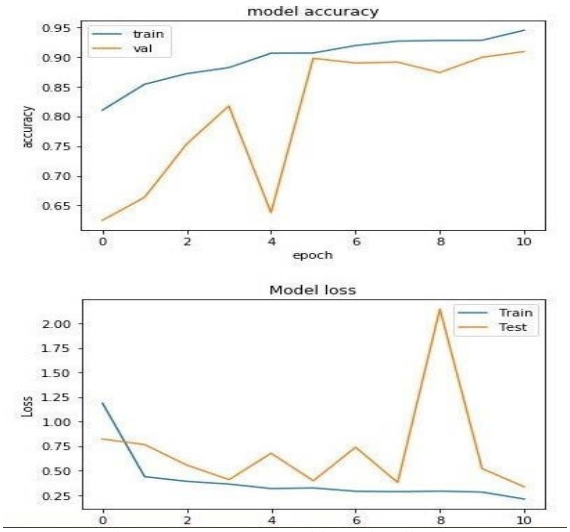

Figure 12. Accuracy and loss visualization

Figure 13 shows model performance evaluation using a series of metrics, like accuracy, precision, recall, etc.

```
Test metrics
----------------
Confusion matrix
 [[181  53]
 [  7 383]]
Accuracy: 90.38%
Precision: 87.84%
Recall: 98.21%
F1-score: 92.74%
ROC AUC: 0.952104
```

Figure 13. Model metrics

In this respect, the model given was concluded to be converging and classify X-ray images with 90.38% accuracy. The model's precision was 87.84%, and a model producing no false positives would have the precision of 1, or 100 %. Therefore, a higher number of false positives is clearly present. There were 53 false positive results, i.e. 53 people diagnosed with a disease were actually healthy. Recall lies at 98.21%, indicating a very low number of false negative diagnoses – only seven. F1 score is 91.68%, which is usually considered an excellent result.

The model could definitely be enhanced by further adjustment of parameters, layers etc. Since train accuracy was 94.45% and test accuracy 90.38%, there was probably some model overfitting. As techniques like ReduceLROnPlateau, dropouts and L2 regularization (which prevent overfitting) were used, other techniques, like training the model with more data or dataset balancing (with respect to the number of images) could be applied.

VI. COMPARING ACTIVATION FUNCTIONS PERFORMANCE

In this section, the difference between applying different rectified activation functions in CNN, namely, between ReLU, Leaky ReLU and PReLU, will be presented. Since ReLU is one of the most frequently used activation functions for ML applications, LeakyReLU can usually be found in the best Kaggle competition projects and PReLU is a variant of LeakyReLU. Performance differences were established using different activation functions. Comparison is given in Table IV.

Table IV. Activation function comparation

Activation function	**Training accuracy**	**Test accuracy**	**Training loss**	**Test loss**
ReLU	0.9460	0.8750	0.1952	0.5904
LeakyReLU ($\alpha = 0.3$)	0.9378	0.8894	0.2106	0.2640
LeakyReLU ($\alpha = 30.0$)	0.9372	0.8702	0.2296	0.2754
PReLU	0.9445	0.9038	0.2088	0.3339

A total of four models implementing different activation functions were built. The same preprocessing, regularization techniques, layers and the same number of epochs were used in all models. Therefore, the only difference between them was the usage of different rectified activations. Their performance was compared and the best model chosen for the final model.

As anticipated, the results were extremely similar, with PReLU achieving the best result. Other types of rectified activations were found to outperform ReLU in the image classification task, achieving similar performance as models created by Xu, Wang et al., described in 'Empirical Evaluation of Rectified Activations in Convolution Network' on CIFAR-100 and CIFAR-10 networks. [18]

Since PReLU outperformed the other 2 functions, achieving the highest accuracy and satisfactory loss, the activation function used in the project was PReLU. Hence, although ReLU is the most commonly used function, better alternatives exist, depending on the dataset.

Since there exist considerably larger datasets than the one used here, the performances of these functions on them are still to be investigated. However, the results suggest that this issue is worth pursuing in the future.

VII. COMPARING RESULTS WITH THE REFERENCE MODEL

To check the performance of the created model, it had to be compared with a reference model. As there are neither ongoing nor completed Kaggle competitions for this dataset, this type of comparison was impossible. The reference model was found online. A Kaggle user used a CNN based on residual NN, or ResNet. A residual NN is an NN, the architecture of which is based on constructs known from pyramidal cells in the cerebral cortex. ResNet is typically implemented with double or triple layer architecture using a ReLu activation function and batch normalizations. In this reference model, the data were augmented in a different manner and trained using a different DL library - *fastai*. The exact same dataset was used.

After preprocessing images in the similar manner as in our project, and conducting a 12-epoch training, the aforementioned reference model achieved the accuracy of 90.86 %, as seen in Figure 14: [19]

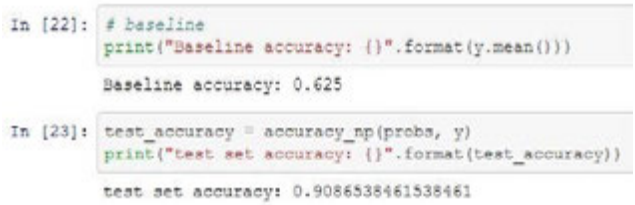

Figure 14. Reference model performance

Since the original model created in the project achieved the accuracy of 90.38%, it was found to be satisfactory and achieve acceptable metrics.

CONCLUSION

In this paper, a medical image classification algorithm based on neural networks was presented. This project shows the true power of machine learning in real-life use. Since the model clearly has satisfying metrics such as accuracy, precision, F1 score etc., it could, with further parameter adjustment (which would resolve the issue of overfitting), be used in a real-life environment. The findings of published papers that neural networks can be of exceptional help in healthcare, have been repeated and confirmed.

REFERENCES

[1] ZhiFei Lai, HuiFang Deng, "Medical Image Classification Based on Deep Features Extracted by Deep Model and Statistic Feature Fusion with Multilayer Perceptron", 12 September 2018.;.

[2] Paul Mooney, "Chest X-Ray Images (Pneumonia)", online, accessed 5/25/2020, March 2018, https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia,

[3] Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, et al., "Artificial intelligence in healthcare: past, present and future, SVN – Stroke and Vacular Neurology", 0:e000101., DOI: 10.1136/svn-2017-000101, June 2017.

[4] Sai Kambampati, "What's New in Core ML 2", online, accessed 5/25/2020, June 2018, https://www.appcoda.com/coreml2/,

[5] Armaan Merchant, "Neural Networks Explained", online, accessed 5/23/2020, December 2018, https://medium.com/datadriveninvestor/neural-networks-explained-6e21c70d7818

[6] Arthur Arnx, "First neural network for beginners explained (with code)", online, accessed 5/25/2020, January 2019, https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf

[7] TM Capital, "The Next Generation of Medicine: Artificial Intelligence and Machine Learning", November 2017.

[8] Asmaa Abbas, Mohammed M. Abdelsamea, and Mohamed Medhat Gaber, "Classification of COVID-19 in chest X-ray images usingDeTraC deep convolutional neural network", Mathematics Department, Faculty of Science, Assiut University, Assiut, Egypt , April 1, 2020.

[9] Wang S, Kang B, Ma J, Zeng X, Xiao M, Guo J, et al. "A deep learning algorithm using CT images to screen for Corona Virus Disease (COVID-19)", medRxiv.2020.

[10] Adit Deshpande, "A Beginner's Guide To Understanding Convolutional Neural Networks", online, accessed 5/23/2020, July 2016, https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/,

[11] Moazzem Hossain , "Easy way to understand Convolutional Neural Network", online, accessed 5/23/2020, April 2019, https://medium.com/@_moazzemhossain/easy-way-to-understand-convolutional-neural-network-its-easy-d9b7c0c5adb3

[12] V.G. Ivancevic. Darryn J Reid, Michael J Pilling, "Mathematics of Autonomy, Mathematical Methods for Cyber-Physical-Cognitive Systems", Defence Science & Technology Group, Australia, 2017

[13] Camron Godbout, "TensorFlow in a Nutshell — Part Three: All the Models", online, accessed 5/25/2020, October 2016, https://medium.com/hackernoon/tensorflow-in-a-nutshell-part-three-all-the-models-be1465993930

[14] Ayperos23, "Github (ayperos23)- pneumonia classification", online, accessed 5/25/2020, May 2020, https://github.com/ayperos23/ML-pneumonia-classification,

[15] Danqing Liu , "A practical guide to ReLU", online, accessed 5/23/2020, November 2017, https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

[16] Algorithmia, "Introduction to Optimizers", online, accessed 5/23/2020, May 2018, https://algorithmia.com/blog/introduction-to-optimizers,

[17] Cory Maklin, "Batch Normalization Tensorflow Keras Example", online, accessed 5/23/2020, June 2019, https://towardsdatascience.com/backpropagation-and-batch-normalization-in-feedforward-neural-networks-explained-901fd6e5393e

[18] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li," Empirical Evaluation of Rectified Activations in Convolution Network", arXiv:1505.00853v2, November 2015

[19] Yuan Tian, "Detecting Pneumonia with Deep Learning ", online, accessed 5/25/2020, June 2018, https://becominghuman.ai/detecting-pneumonia-with-deep-learning-3cf49b640c14