American Academic Scientific Research Journal for Engineering, Technology, and Sciences

ISSN (Print) 2313-4410, ISSN (Online) 2313-4402

https://asrjetsjournal.org/index.php/American_Scientific_Journal/index

Detection and Interpretation of X-Ray Scans for the Presence of Pneumonia Using Convolutional Neural Network

Peter Oluwasayo Adigun[a]*, Ayodeji Adedotun Adeniyi[b], Tobi Titus Oyekanmi[c]

[a]*Department of Computer Science, New Mexico Highlands University, 1005 Diamond St, Las Vegas, New Mexico, USA*

[b,c]*Department of Media, Art, and Technology, New Mexico Highlands University, 1005 Diamond St, Las Vegas, New Mexico, USA*

[a]*Email: poadigun@nmhu.edu*

[b]*Email: aadeniyi1@live.nmhu.edu*

[c]*Email: toyekanmi@live.nmhu.edu*

Abstract

Convolutional neural network's application is essentially an impactful technology to proffering solutions in medical diagnostics. This research carried out a design and implementation of a medical imaging analysis and classification of X-ray scans of pneumonia images using a convolutional neural network. The CNN system was designed using an algorithm of a convolutional neural network. The designed CNN system was processed by uploading 5,216 data which comprised normal and pneumonic image scans. The CNN system was trained with 5,000 datasets and tested. The findings from the study established that the implemented system based on a convolutional neural network algorithm is 76% accurate. This study is subjected to further studies.

Keywords: Machine learning (ML); Convolutional Neural Network (CNN); Pneumonia Disease (PD).

Received: 11/23/2024
Accepted: 1/23/2025
Published: 2/3/2025

----

* Corresponding author.

1. Introduction

The world of machine learning (ML) has unending applications in the universe. Through the design and perception of machines, artificial intelligence (AI), and robotics; physical problems, technical challenges, and most especially medical contingencies and healthcare complications have been mitigated and perused. This study has found the pertinence of artificial neural networks by employing the convolutional neural network (CNN) in detecting and interpreting the X-ray scans for the presence of pneumonia. Artificial neural networks (ANNs), simply called neural networks (NNs), are computing systems inspired by the biological neural networks that constitute animal brains [1]. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain [2]. An artificial neural network is known as one of the common mathematical-computational models that are employed to solve unanticipated dynamic problems in developed behavioral systems. By learning to recognize patterns from data in which other computational and statistical methods failed to solve, artificial neural networks can solve the problems [3].

Convolutional neural network (CNN) is a type of ANN in deep learning (DL) that is mostly applied to analyze visual imagery [4]. CNN also uses a mathematical operation called convolution in place of general matrix multiplication in at least one of its layers [5]. CNNs are specifically designed to process pixel data, interpret it, and use it for image recognition and processing. CNN applications include image and video recognition, recommender systems [6], image classification, image segmentation, medical image analysis, and many more as indicated by scholars [7, 8]. CNN has gained prominence in various applications like pattern recognition, which this study will employ in designing a system of CNN that detects pneumonia from X-ray scan images.

This study aims to model and design a CNN system that reads X-ray scans and interprets the results. The objectives of the research are:

- To code a CNN algorithm that will read and interpret the pneumonia clinical data.
- To train the system for reading and detecting the dataset using the CNN algorithm.
- To test and verify the system for simulation of different pneumonia clinical data.

2. Methods

The study involved designing a convolutional neural network (CNN) system to detect, identify, and interpret X-ray clinical data of patients with pneumonia. The CNN, utilizing deep learning, was trained on secondary data sourced from Kaggle's "Chest X-Ray Images (Pneumonia)" dataset by Paul Mooney, updated five years ago, and rated 7.50 in usability as of June 15, 2023. The dataset, originally from Guangzhou Women and Children’s Medical Center, includes pediatric patient X-rays, screened for quality control, depicting clear lungs, bacterial pneumonia with focal lobar consolidation, and viral pneumonia with a diffuse interstitial pattern.

The classification of chest X-ray images (pneumonia) was performed on Python. Essential Python libraries such as NumPy, Pandas, Matplotlib, Seaborn, and TensorFlow were imported and initialized. Detailed information and descriptions of the dataset were computed, followed by visualizations to ensure quality control by identifying and removing any low-quality or unreadable scans. Data preparation involved recoding, categorizing, and manipulating the dataset to make it suitable for preprocessing and training. The TensorFlow library was employed to apply Convolutional Neural Network (CNN) algorithms. The trained models were used to simulate the diagnosis of PD data, with each model tested and evaluated, and performance metrics computed to assess accuracy and efficacy.

3. Results

3.1 Data Training

The X-ray image scanning model was developed using supervised learning (SL) and CNN. The output of the data modeling is presented below.

The generators yielded batches of image data along with their corresponding labels (binary class labels in this case) during the model training and evaluation process. This function helped in visualizing the X-ray scan associated with a specific index, which enabled the inspection of the data and gain insights into the characteristics of the X-ray images in the dataset.

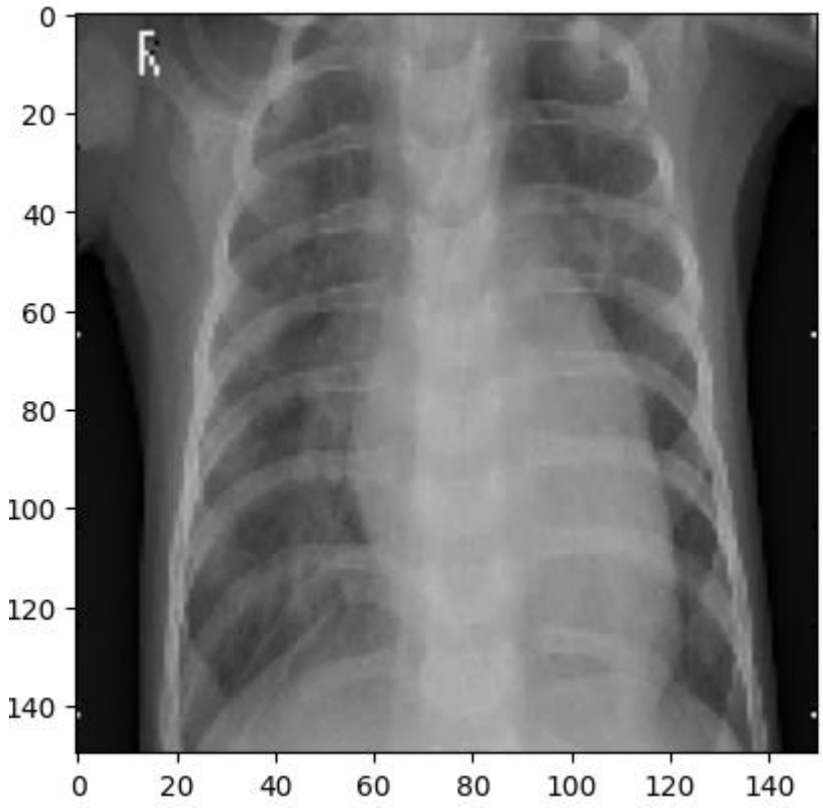

Figure 1: X-ray image from training data

A grid of subplots to display a random selection of X-ray images from the training data was created. The result of this operation allows for a quick visual inspection of multiple X-ray scans, aiding in the exploration and analysis of the dataset.

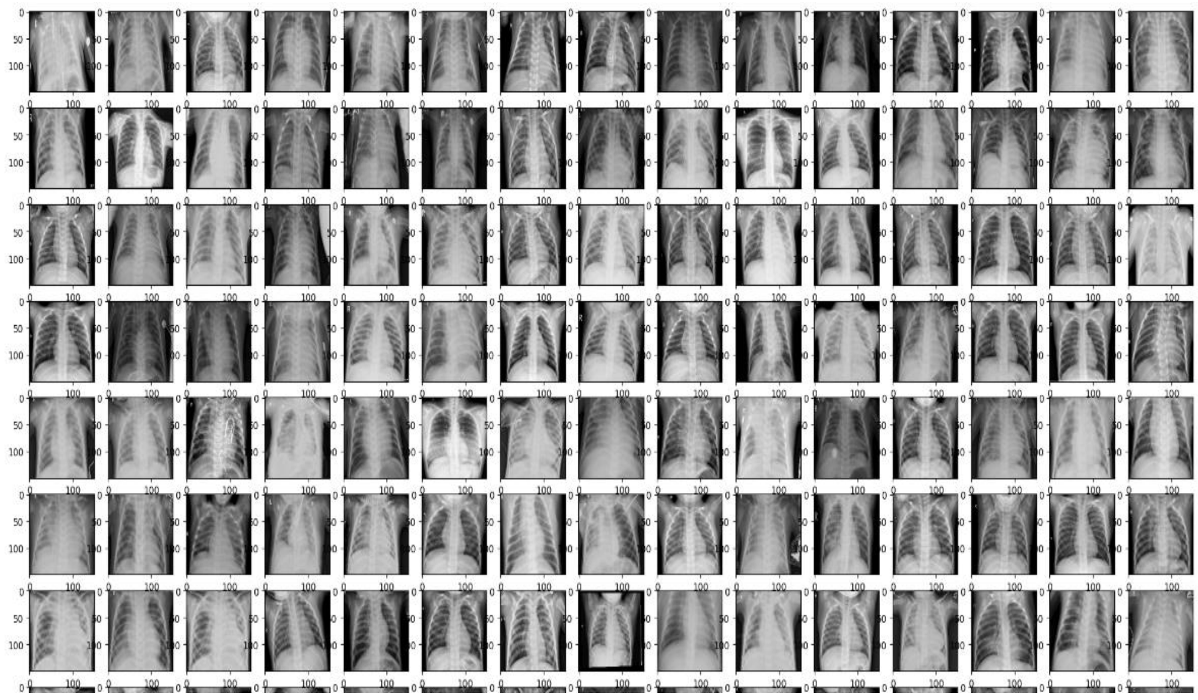

Figure 2: Random X-ray images from training data

Also, a grid of subplots to display a random selection of X-ray images from the testing data was created. This enables to visualize multiple X-ray images at once, providing a representative sample from the testing dataset.

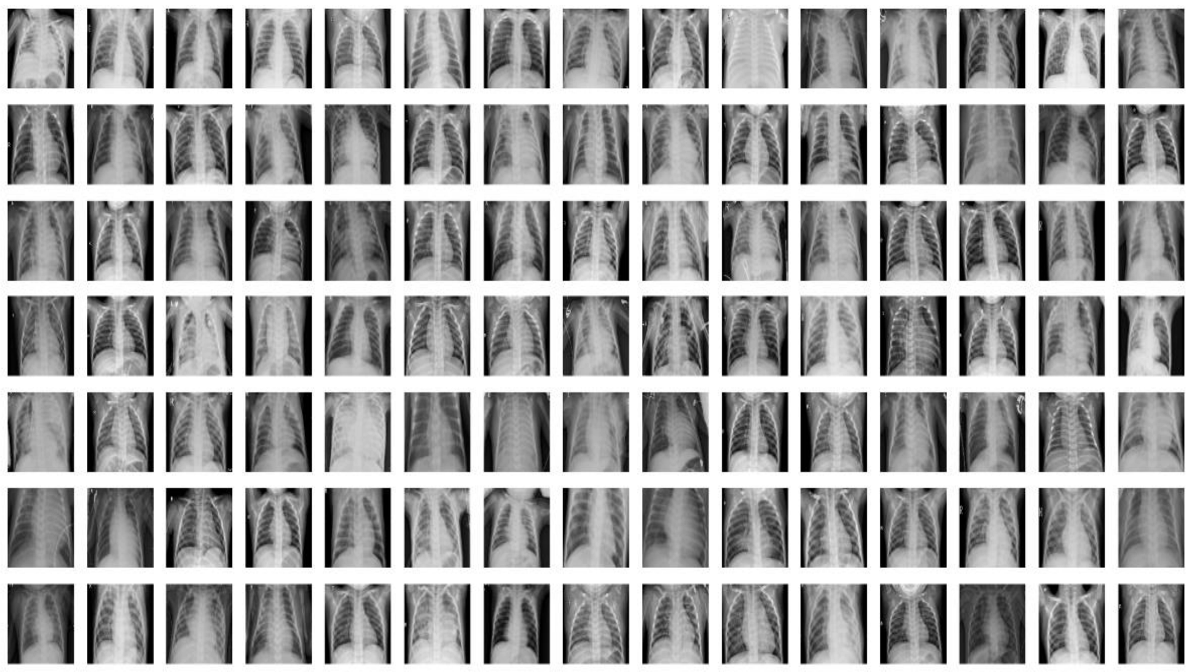

Figure 3: Random X-ray images from testing data

Additionally, an enhanced grid of subplots that displays the X-ray images of the training data, with their titles, and omits the axes was created for a concise and informative visualization.

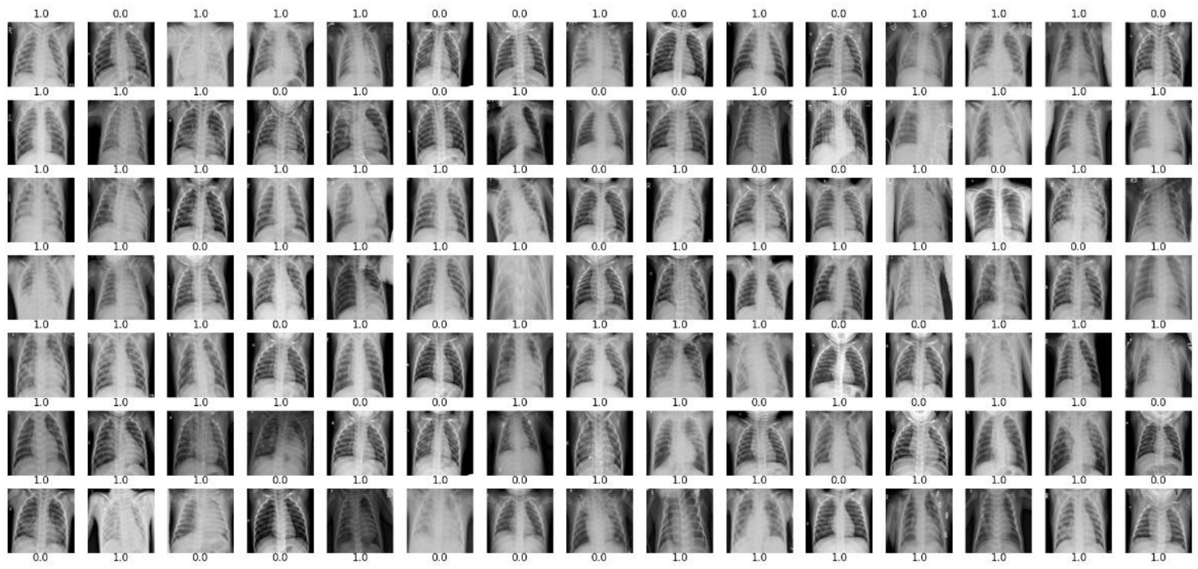

Figure 4: Random X-ray images from training data

The pixel values of the trained data were generated. Each value in the array represented the RGB color values of a pixel in the images of the training dataset.

Output 1: *X_train* pixel values

```
array([[[[0.20392159, 0.20392159, 0.20392159],
         [0.2509804 , 0.2509804 , 0.2509804 ],
         [0.36862746, 0.36862746, 0.36862746],
         ...,
         [0.29411766, 0.29411766, 0.29411766],
         [0.29411766, 0.29411766, 0.29411766],
         [0.2392157 , 0.2392157 , 0.2392157 ]],

        [[0.3019608 , 0.3019608 , 0.3019608 ],
         [0.35686275, 0.35686275, 0.35686275],
         [0.3921569 , 0.3921569 , 0.3921569 ],
         ...,
         [0.3137255 , 0.3137255 , 0.3137255 ],
         [0.29411766, 0.29411766, 0.29411766],
         [0.2627451 , 0.2627451 , 0.2627451 ]],

        ....................................,

        [[0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ],
         ...,
         [0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        ]]]], dtype=float32)
```

Figure 8

The pixel values were generated for the tested data. Each value in the array represented the RGB color values of a pixel in the images of the testing dataset shown before.

Output 2: *X_test* pixel values

```
array([[[[0.22352943, 0.22352943, 0.22352943],
         [0.0509804 , 0.0509804 , 0.0509804 ],
         [0.05882353, 0.05882353, 0.05882353],
         ...,
         [0.8352942 , 0.8352942 , 0.8352942 ],
         [0.85098046, 0.85098046, 0.85098046],
         [0.86274517, 0.86274517, 0.86274517]],

        [[0.0627451 , 0.0627451 , 0.0627451 ],
         [0.04313726, 0.04313726, 0.04313726],
         [0.05882353, 0.05882353, 0.05882353],
         ...,
         [0.8705883 , 0.8705883 , 0.8705883 ],
         [0.8588236 , 0.8588236 , 0.8588236 ],
         [0.89019614, 0.89019614, 0.89019614]],

        ..................................,

        [[0.08235294, 0.08235294, 0.08235294],
         [0.08235294, 0.08235294, 0.08235294],
         [0.07058824, 0.07058824, 0.07058824],
         ...,
         [0.08235294, 0.08235294, 0.08235294],
         [0.08235294, 0.08235294, 0.08235294],
         [0.08235294, 0.08235294, 0.08235294]]]], dtype=float32)
```

Figure 9

Output was generated using a *numpy array* representing an image in the form of pixel values. Each value in the array represented the RGB color values of a pixel in the images of the training dataset. *"y_train"* typically represented the ground truth labels or target values associated with the X-ray scan images in the training dataset. These labels indicate whether each X-ray scan belongs to a specific class, such as pneumonia or normal. Another output was generated: *numpy array* representing an image in pixel values. Each value in the array represented the RGB color values of a pixel in the images of the testing dataset. *"y_test"* typically represented the ground truth labels or target values associated with the X-ray scan images in the training dataset. These labels indicate whether each X-ray scan belongs to a specific class, such as pneumonia or normal.

Output 3: *y_test* pixel values

```
array([1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0., 1., 1., 0.,
       1., 1., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1., 1., 1., 1.,
       1., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 1., 1.,
       1., 0., 0., 1., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 0., 0.,
       0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 0., 1., 0., 1.,
       1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 0., 1., 0., 0., 1., 1., 0., 1., 1., 1., 0., 1., 1.,
       1., 1., 0., 0., 0., 1., 0., 1., 1., 0., 1., 1., 1., 0., 0., 1., 0.,
       ..................................................................
       0., 1., 1., 1., 0., 1., 1., 0., 0., 0., 0., 1., 1., 1., 0., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 1.,
       1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 0., 1., 0., 0.,
       1., 1., 0., 1., 1., 0., 0., 0., 1., 1., 1., 0., 1., 1., 0., 1., 1.,
       1., 0., 0., 1., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 1., 1., 1.,
       1., 1., 1., 0., 1.], dtype=float32)
```

Figure 10

3.2 Data Simulation

The convolutional neural network (CNN) model architecture for predicting pneumonia through X-ray scans was initialized using lines of code in Inpit 1. The CNN model architecture defined by this code snippet consists of multiple convolutional and pooling layers followed by fully connected layers, culminating in an output layer. The model takes X-ray scan images as input and aims to predict whether each image belongs to the pneumonia or non-pneumonia class.

Input 1: CNN model architecture for pneumonia prediction

```
cnn_model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

Figure 11

Input 1 was used for compiling the CNN model for pneumonia prediction through X-ray scans. It specified the optimizer, loss function, and evaluation metrics to be used during the training process. The training parameters is depicted below whch consist of the batch size, number of epochs, and shuffling of data during training. The *'fit'* function trained the model using the provided training data, batch size, and number of epochs. During training, the model learns to make predictions based on the input data and adjusts its internal parameters (weights and biases) to minimize the specified loss function. The training progress and metrics are stored in the *‘history’* object, allowing for further analysis and visualization of the training process.

Output 4: Model training output

```
Epoch 1/5
157/157 [==============================] - 257s 2s/step - loss: 0.2228 - accuracy: 0.9
104
Epoch 2/5
157/157 [==============================] - 257s 2s/step - loss: 0.1009 - accuracy: 0.9
620
Epoch 3/5
157/157 [==============================] - 264s 2s/step - loss: 0.0795 - accuracy: 0.9
720
Epoch 4/5
157/157 [==============================] - 265s 2s/step - loss: 0.0613 - accuracy: 0.9
782
Epoch 5/5
157/157 [==============================] - 260s 2s/step - loss: 0.0529 - accuracy: 0.9
806
```

Figure 12

The provided code snippet in Input 2 is responsible for evaluating the trained CNN model using the test data. It is called the *‘evaluate’* function on the *‘cnn_model’* with the test data and assigns the resulting loss and accuracy values to the variables *‘loss’* and *‘accuracy’*, respectively. It then printed the test accuracy. By evaluating the model on the test data, this code snippet provided an assessment of the model's performance on unseen data. The test accuracy indicated how well the model generalizes to new X-ray scan images that were not used during training.

Input 2: Model evaluation

```
loss, accuracy = cnn_model.evaluate(test_data)
print(f'Test accuracy: [9]')
```

Figure 13

Output 5: Model evaluation output

```
2/2 [==============================] - 23s 312ms/step - loss: 1.4473 - accuracy: 0.759
6
Test accuracy: 0.7596153616905212
```

Figure 14

Input 2 generated two plots to visualize the training and validation metrics (accuracy and loss) over the training epochs. The first plot displays the model accuracy, while the second plot showed the model loss. These plots provided insights into the model's learning progress and performance during training. The accuracy plot showed how the model's accuracy improves or stabilizes over the training epochs, while the loss plot illustrated the decrease in the model's loss (indicating improved predictions) as training progresses. The comparison between the training and test curves helps assess potential overfitting or underfitting of the model.

Output 6: Training and validation metrics visualization output

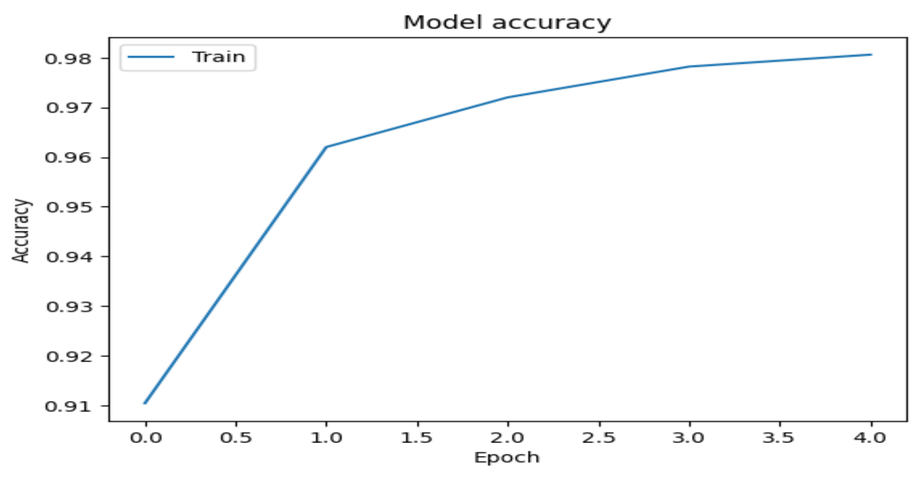

Figure 5: Training and validation metrics 1

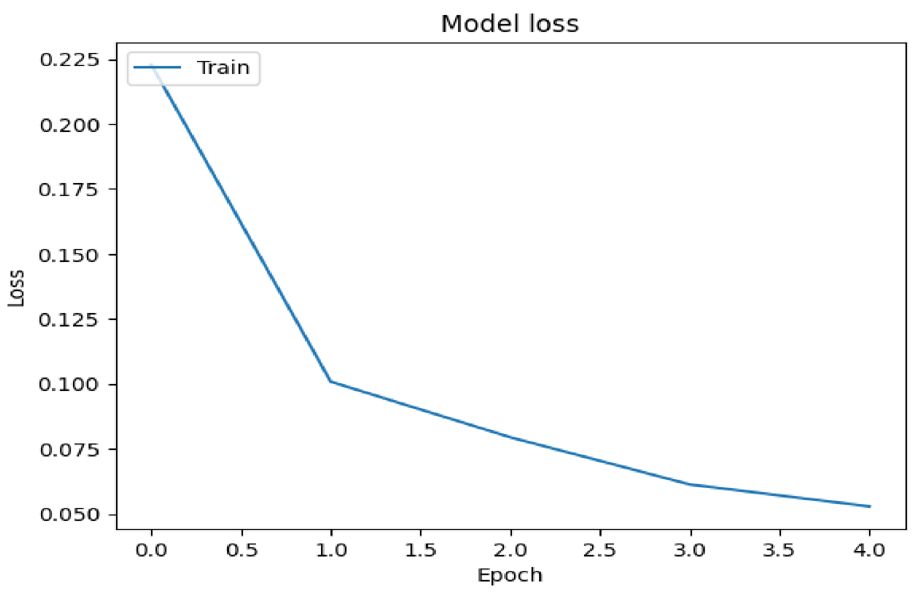

Figure 6: Training and validation metrics 2

Input 3 is responsible for predicting the classes of the test data using a trained CNN model. The class probabilities are calculated using the *'predict'* function, and then these probabilities are converted into binary predictions using a threshold of 0.5. By applying the threshold of 0.5, the code snippet converted the predicted class probabilities into binary predictions, enabling the classification of the test data into the respective classes (in this case, for pneumonia prediction through X-ray scans).

Input 3: Class prediction using thresholding

```
# Predict the class probabilities of the test data
predicted_classes_prob = cnn_model.predict(X_test)

# Convert the class probabilities to binary predictions using a threshold of 0.5
predicted_classes = (predicted_classes_prob > 0.5).astype(int)
```

Figure 15

Output 7: Class prediction using thresholding output

```
19/19 [==============================] - 10s 528ms/step
```

Figure 16

Input 4 presented the binary predictions obtained from the predicted class probabilities. It is an array-like object that contains the predicted classes for each sample in the test dataset. Each element in the array is either a *'0'* or *'1'*, where *'0'* indicates a negative prediction (no pneumonia) and *'1'* indicates a positive prediction (pneumonia detected).

Input 4: Predicted classes of pneumonia

```
predicted_classes
```

Output 8: Predicted classes of pneumonia output

```
array([[1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [0],
       [1],
       [1],
       [0],
            .
            .
            .
       [1],
       [1],
       [1],
       [1],
       [1]])
```

Figure 17

The provided code snippet in input 4 is responsible for displaying the predictions for the test images in a grid format. It used the *matplotlib library* to create subplots and visualize the images along with their corresponding predicted classes and true labels.

Output 27 displayed the test images along with their predicted classes and true labels. It provided a visual representation of the model's predictions and allows for comparison with the ground truth labels.

Output 9: Displaying predictions for test images

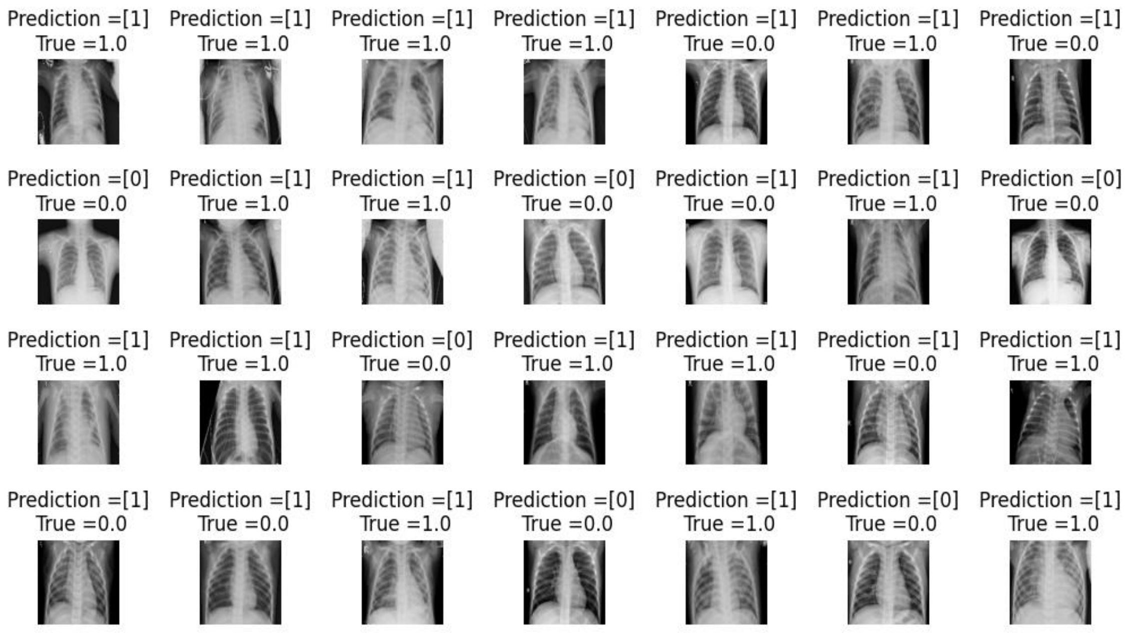
Prediction =[1] True =1.0
Prediction =[1] True =1.0
Prediction =[1] True =1.0
Prediction =[1] True =1.0
Prediction =[1] True =0.0
Prediction =[1] True =1.0
Prediction =[1] True =0.0
Prediction =[0] True =0.0
Prediction =[1] True =1.0
Prediction =[1] True =1.0
Prediction =[0] True =0.0
Prediction =[1] True =0.0
Prediction =[1] True =1.0
Prediction =[0] True =0.0
Prediction =[1] True =1.0
Prediction =[1] True =1.0
Prediction =[0] True =0.0
Prediction =[1] True =1.0
Prediction =[1] True =1.0
Prediction =[1] True =0.0
Prediction =[1] True =1.0
Prediction =[1] True =0.0
Prediction =[1] True =0.0
Prediction =[1] True =1.0
Prediction =[0] True =0.0
Prediction =[1] True =1.0
Prediction =[0] True =0.0
Prediction =[1] True =1.0

Figure 7: Predictions for test image

Output 29: Output of X-ray Image Classification

```
1/1 [==============================] - 0s 131ms/step
The X-ray image is classified as: [[0.9999861]]
```

Figure 18

4. Conclusion

The design and implementation of a CNN system that has the functionality of detecting and interpreting X-ray scan of pneumonia disease has been achieved. The implementation is carried out using a convolutional neural-network's application due to the its essentiality and technological impact in proffering solutions in medical diagnostics. The CNN system was designed using an algorithm of convolutional neural network and TensorFlow framework. The findings from the study established that the implemented system based on convolutional neural network algorithm is 76% accurate.

References

[1] J. Mäkelä, "Bioelectric measurements: magnetoencephalography," in *Comprehensive biomedical physics*: Elsevier BV, 2014, pp. 47-72.

[2] L. Hardesty, "MIT News Office. Explained: Neural networks," ed: MIT News. April, 2017.

[3] S. R. Khaze, M. Masdari, and S. Hojjatkhah, "Application of artificial neural networks in estimating participation in elections," *arXiv preprint arXiv:1309.2183*, 2013.

[4] M. V. Valueva, N. N. Nagornov, P. A. Lyakhov, G. V. Valuev, and N. I. Chervyakov, "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation," *Mathematics and Computers in Simulation,* vol. 177, pp. 232-243, 2020/11/01/ 2020, doi: https://doi.org/10.1016/j.matcom.2020.04.031.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[6] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," presented at the Proceedings of the 25th international conference on Machine learning, Helsinki, Finland, 2008. [Online]. Available: https://doi.org/10.1145/1390156.1390177.

[7] O. Avilov, S. Rimbert, A. Popov, and L. Bougrain, "Deep Learning Techniques to Improve Intraoperative Awareness Detection from Electroencephalographic Signals," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 20-24 July 2020 2020, pp. 142-145, doi: 10.1109/EMBC44109.2020.9176228.

[8] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, 24-27 July 2017 2017, vol. 01, pp. 7-12, doi: 10.1109/CBI.2017.23.

[9] H. L. Chang, C. C. Wu, S. P. Lee, Y. K. Chen, W. Su, and S. L. Su, "A predictive model for progression of CKD," (in eng), *Medicine (Baltimore),* vol. 98, no. 26, p. e16186, Jun 2019, doi: 10.1097/md.0000000000016186.

[10] K. Fokianos, I. Sarrou, and I. Pashalidis, "Increased radiation exposure by granite used as natural tiling rock in Cypriot houses," *Radiation Measurements,* doi: DOI: 10.1016/j.radmeas.2007.02.001 vol. 42, no. 3, pp. 446-448, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6TVS-4N3GFS4-3/2/cdd1de80b3944ca79c1733a1f1734bed.