Article

Using Neural Networks and Tensorflow to detect the presence of Pneumonia in a Patient

Shawn Gonsalves [1], Dr. Xin Ye [2]

Department of Computer Science, California State University, San Marcos, California, CA 92096, USA; gonsalveshawn@gmail.com; **(760) 884-1154**
Department of Computer Science, California State University, San Marcos, California, CA 92096, USA; xye@csusm.edu; **(760) 750-8241**

Abstract: This study proposes the design, development and deployment of a Convolutional Neural Network model that detects the presence of Pneumonia in a patient based on his/ her X-Ray images. The model is trained from scratch using Python and different libraries associated with Python, so it separates itself from the already pre-trained models. A chest x-ray is the best test for diagnosing pneumonia. The diagnosis of Pneumonia disease is a human-dependent study. [1] An x-ray exam will allow your doctor to see your lungs, heart, and blood vessels to help determine if the person suffers from pneumonia. When interpreting the x-ray, the radiologist will look for white spots in the lungs (called infiltrates) that identify an infection. Having a model that detects this presence eliminates the two types of human error, which are Mistakes and Violations. Once the model is trained with enough training images of both Normal and Pneumonia X-rays, then the prediction for on a new X-Ray can be made within a matter of seconds. The trained Convolutional neural network model is put into production by deploying the JSON model on a web server. The deployment process is done using Tensorflow-JS, where the model is first fed into the browser, and then a frontal X-ray image is uploaded for inference. The model predicts if it belongs to the Normal or the Pneumonia positive class.

Keywords: X-ray, Overfitting, Keras, ANN's, CNN, model, tensor, Flatten MaxPooling, Dropout, Radiologist, Transfer Learning, Relu.

---

1. Introduction

Pneumonia is a disease that infects the lungs and is caused by bacteria, viruses, or fungi. [2] The early diagnosis of this disease is the key to prevent the spread of pneumonia throughout the body.

We need a system that detects the presence of this disease at a rate that is as fast as the Radiologists, if not more and something which is easily available to the end-user. [3] In the proposed technique. We implement the concept of Artificial neural networks to develop a Machine Learning model into production which is a computational model that is inspired by the way biological neural networks in the human brain process information. ANN's have proven to be the breakthrough in the ML industry by portraying their prowess in the fields of Speech Recognition, Computer Vision as well as text processing.

The input for the model is images of frontal-view of chest X-rays, and it outputs either of the two values, i.e., If the image represents Normal Chest X-Ray or Pneumonia positive X-ray image. The deployment of the model on the browser eases the work of the end-user as he/she just has to upload the chest X-ray image on the browser and the browser will predict the class of the image(Normal/Pneumonia).

Background

Pneumonia can lead to the death of the infected person, due to the impotence of exchanging gas in the lungs. This disease mainly affects infants and young children and adults over the age of 65. Nearly 4 million people die every year, and as low as 420 million people get infected by this disease. The diagnosis of this disease is very crucial, especially detecting it early. The best method in diagnosing the disease is using X-Ray images of the patients. This diagnosis of Chest X-ray images of the affected patients is done by expert radiologists. Radiology is the study of dealing with X-rays and other high-energy radiation, especially the use of such radiation for the diagnosis and treatment of disease. [1] The examination of X-Ray images helps in determining if you have any complications related to pneumonia, such as abscesses or pleural effusions (fluid surrounding the lungs). Thus the Radiologists have to diagnose each X-ray image closely, which takes quite some time.

The advancements in Computer Vision along with Deep Learning, has been constructed and perfected, primarily over one particular algorithm, Convolutional Neural Networks. [3] We use Convolutional Neural Networks (ConvNets) to build our model from scratch. A model built from scratch helps fine tune the hyperparameters as per the requirements. Also, training the model from scratch is more efficient in terms of size and training time as there may be less number of layers as compared to any pretrained model trained for more than one purpose. For Instance, the VGG16 Architecture is trained for 1000 different categories. Most of the models built from scratch have a relatively lower accuracy and much higher loss when tested on a dataset that the model has not seen before. Also, there are not many popular models that have been deployed on the web server having a relatively high prediction accuracy that does this detection on a Chest X-Ray images.

Related Work

Recently, there has been a huge demand for detecting various kinds of medical problems with the help for Artificial Intelligence (AI). Common medical related problems like Breast cancer detection, Brain Tumor detection, Computer Aided Design (CAD) has been primarily used for Chest radiographs to detect the presence of pneumonia because it helps in early diagnosis of the disease [13]. However, the accuracy was significantly lower than the models developed using Deep neural networks and Convolutional neural networks to detect this disease. There have been previous studies to detect the presence of Pneumonia at an

early stage but [16] is one of the first ones to present automations of different parameters on x-ray images, which can help diagnose the disease at very early stage.

The examination of Chest X-Rays to detect any disease was carried out in [14] which presents an efficient image categorization and retrieval system. The screening of Chest radiographs to check the presence of Tuberculosis was carried out in [15]. A Deep-learning based method using Dense-Net architecture was proposed in [17] which uses Supervised Learning technique. Another similar experiment uses Long Short-Term Memory (LSTM) architectures [18] which focuses on 14 interdependent finding of the disease. This implies that the model does the classification between 14 different classes of images which yielded a comparatively lower accuracy of 71.3%. The analysis of Chest X-ray images for different body part segmentation was performed in [19]. A proposed research [20] based on the classification of pediatric pneumonia images using a residual structure with dilated convolution. This research mitigates the problem of low image resolution, partial occlusion or overlap and improves the performance of the model by avoiding negative impact of the structured noise on the performance of the model.

2. Materials and Methods

2.1. Dataset

The dataset was imported and downloaded from the Kaggle website. [4] The Chest Xray dataset is organized into 3 separate directories: the train directory, test directory, Validation directory and contains sub-directories for each image category (Pneumonia/Normal). So, there are a total of 5,863 frontal Chest X-Ray images and 2 categories (Pneumonia/Normal) as positive and negative Pneumonia images respectively.

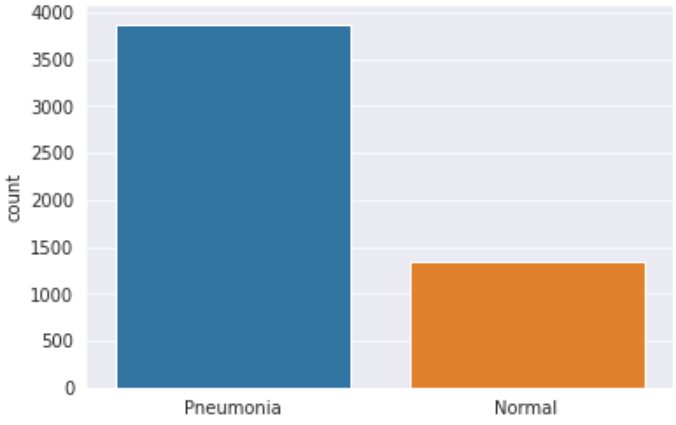


Fig.1

Fig.1 represents the graph of the number of Pneumonia and Normal images.

For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing some unreadable scans and all low-quality images. Two expert physicians then graded the diagnoses for the images before being cleared for training the AI system. The evaluation dataset was then checked by a third expert to account for any grading errors.

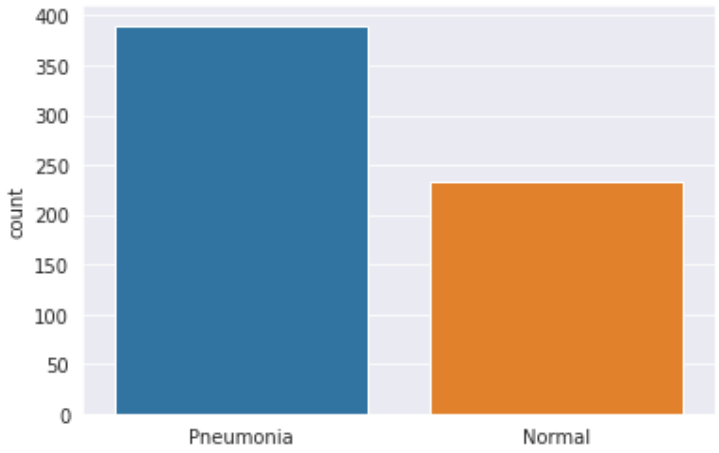


Fig.2

The above Graphical representation shows the number of Test Normal and Pneumonia images.
We see that the data here is imbalanced, so we want to augment the training images by applying some transformations to increase the number of examples.

2.2 Data Preprocessing and Augmentation

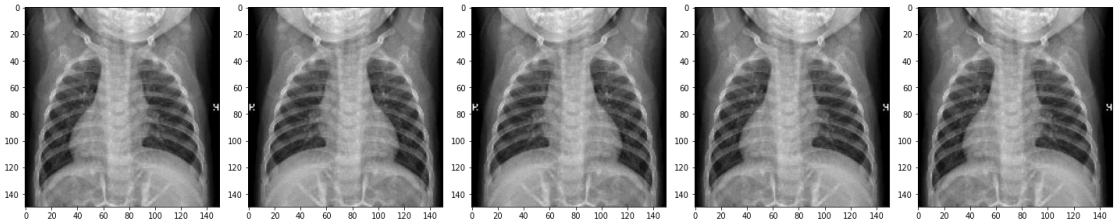


Fig.3

The training images of Normal Chest X-Ray were much smaller than its Pneumonia counterpart. Thus, the model could overfit because it may not generalize the image of Normal X-ray of training directory. So, we need to augment the size of the training directory artificially. Thus, to Augment the number of training Normal X-ray images, pre-processing was done on those images. Fig.3 represents the original version of the image, as found in the dataset. Whenever we want to train our model using CNN to recognize images or objects, in general, we want it to detect that image regardless of its size or position in the image.

The value of each pixel in the image data is an integer in the range [0,255]. For the model to work correctly, these values need to be normalized to the range [0,1]

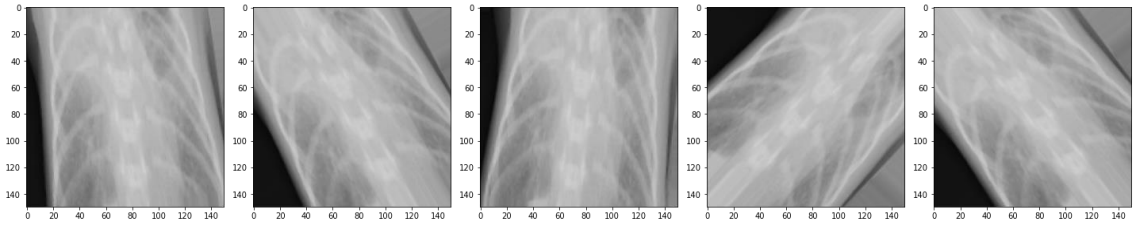


Fig.4

Thus, we start with rescaling the input in the [0, 255] range to be in the [0, 1] range. Then we flip random train images horizontally. Then we rotate a few random images by specifying the angle of rotation and increase the zoom range of random images. Fig 4 represents the augmented image after rotation by 45 degrees. Fig 5 represents the augmented image after applying the zoom factor.

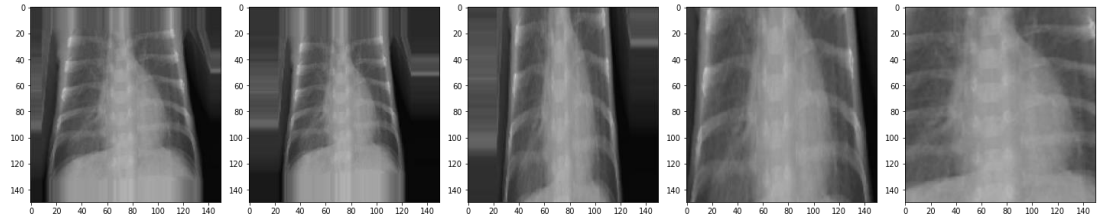


Fig.5

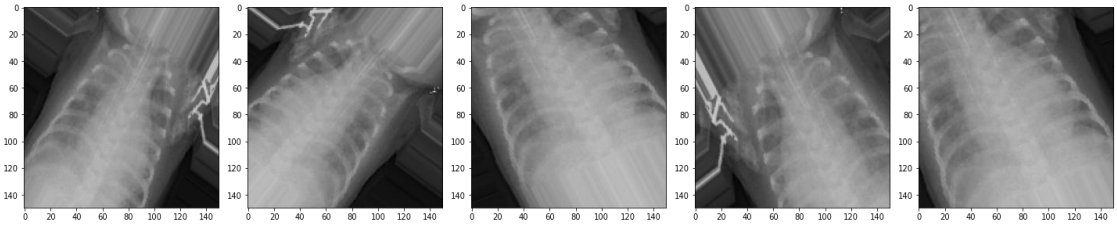


Fig.6

After applying these transformations our image can look something like in the above figure. By applying just a couple of these transformations to our training data, we can easily augment the number of training examples and create a very robust model as shown in Fig.6. Some popular augmentations people use are grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations, and much more. We kept the test and validation images as it is because we want to perform inference, after our model has done its training.

2.3. *Architecture*

The model is built from scratch by specifying the model as a Sequential model which sequences each layer of the model. **Sequential** is the easiest way to build a **model** in Keras. Each layer has weights that correspond to the layer that follows it. We use the combinations of Convolutional Neural networks (ConvNets) and Max-Pooling layers to build our model. [5] ConvNets are considered as an important tool by most machine learning practitioners and are proven very effective in areas such as image recognition and classification. Given an input image, a Convolutional Layer applies the learned filters an generates a feature map which is then used to summarize the filter presence in the input. Maxpooling Layer is followed the Convolutional Layer which down-samples the feature map and creates a new set of the same number of pooled feature maps. The model is built using a Maxpooling layer which highlights the most present feature from a window stride. [6] Most models in Deep Learning are re-built using pre-trained models by experts and reloaded for classification, using a technique called Transfer Learning. In transfer learning, developers cannot remove the network layers to find optimal AI models with confidence. If the developer removes the first layers, then it will affect the dense layers as the number of trainable parameters will change. The parameters and the hyper-parameters can be tuned according to the performance of the model.

The optimization algorithm used is Adam. Adam optimizer is considered among the best optimizers where the dataset is large with noisy/or sparse gradients. Adam optimizer involves to gradually improvise the Learning rate unlike the Stochastic Gradient Descent which has a constant Learning rate throughout the process.

2.4. *Model*

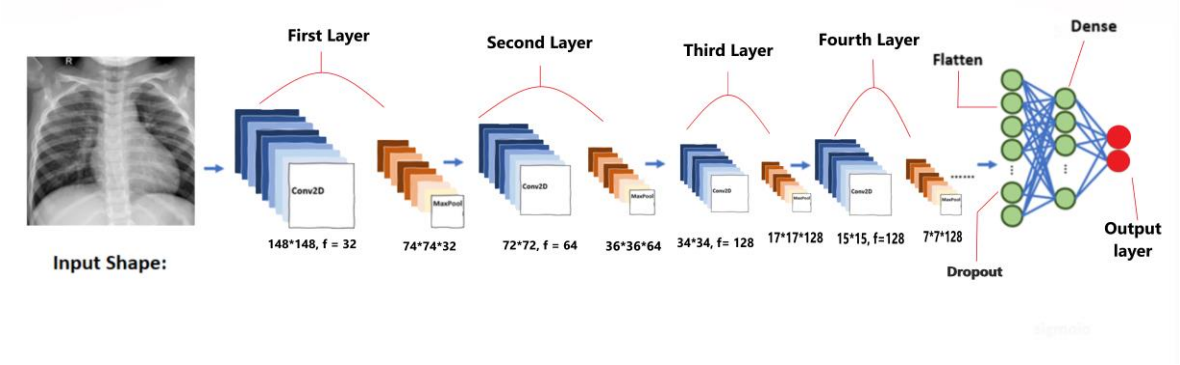


Fig 6.

In the creation of our model, we are using Convolutional Neural Networks. [7] The model can only process images with one input size. So, the model is first converted to that specific height and width. The first layer is a Conv2D filter (3,3) being applied to the input image and creating 32 output (convoluted) images. So, we are going from a single input image to 32 convoluted output images after applying this layer to our model. After that, the 32 outputs are reduced in size using a MaxPooling2D (2,2) with a stride of 2. Stride is the number of pixels shifts over the input matrix. When the stride value is 1 then we move the filters to 1 pixel at a time. When the stride is 2 then we move the filters to 2 pixels at a time and so on.

The next three combinations of Conv2D and max-pooling layers have units 64 and two 128's, respectively. So, we create 64 convoluted outputs in the second round of the Conv and max-pooling layer and 128 convoluted output images in the remaining two rounds.

Sometimes, one part of the layer ends up with larger weights, and the other with very small weights. This situation results in parts having larger weights playing a major role in the training process and ignoring the parts with smaller weights and they don't get trained very much. Thus, after the model is trained for a few epochs it slowly starts to memorize the trained data which in turn causes overfitting.[8] Overfitting results in the model, not being able to generalize to the data in the validation set.[9] One way to prevent the model from Overfitting is by applying a Dropout Layer to the model. The dropout layer shuts off random neurons in the layers. Turning off random neurons in the layers forces the other neurons to pick up the slack and take a more active part in the training. Some neurons may get turned off many times, while some neurons won't turn off. As this is done over and over it makes the neural network more resistant because it cannot rely on all the neurons being there to solve the problem.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026
=================================================================
Total params: 3,453,634
Trainable params: 3,453,634
Non-trainable params: 0
```

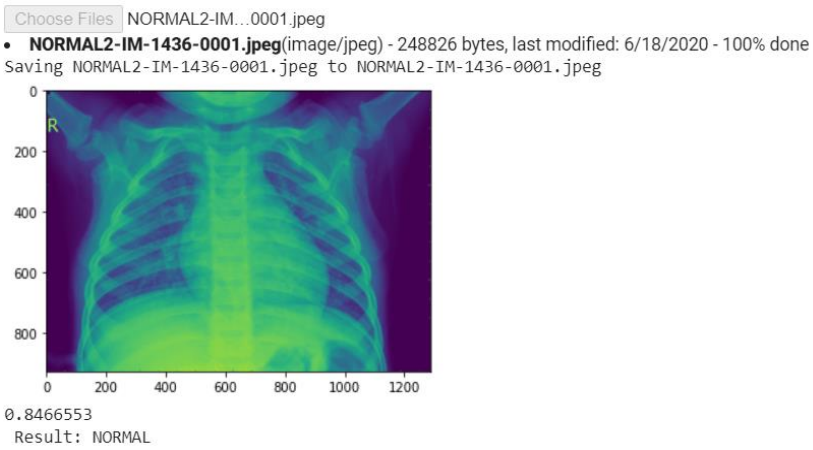
The classifier also requires individual features (vectors) to perform computations. Therefore, the output of the feature extractor (CNN part) is converted into a 1D feature vector for the classifiers. This process is known as **flattening** where the output of the convolution operation is flattened to generate one lengthy array(1-dimensional) for the dense layer to utilize in its final classification process.

The final layers of the model are the Dense layer with 512 neurons with RELU as an activation function. The rectified linear activation function is a linear function that will output the input directly if it is positive, otherwise, it will output zero.

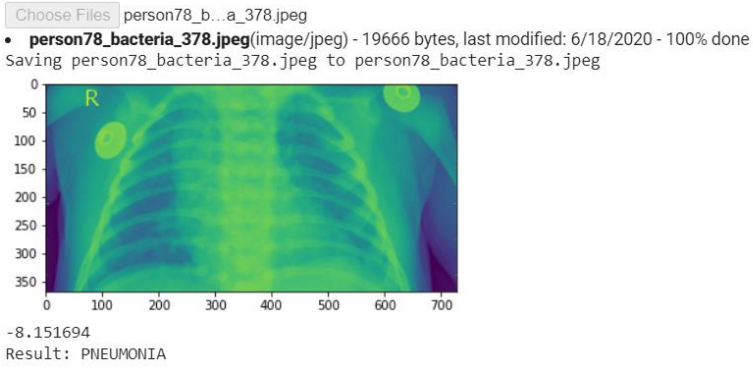
[10] The trained model is ready for the inference process. The model can be saved in the path specified using the **model.save()** method. The entire model can be saved in two different file formats (SavedModel and HDF5) and be used without access to the original Python code.

3. Results

The inference process in Image Classification begins by using the files.upload() method to input the image. This image gets converted to a NumPy array and its shape can be expanded by using the np.expand_dims. The processed image is then rescaled and fed into the model.predict() method for inference.

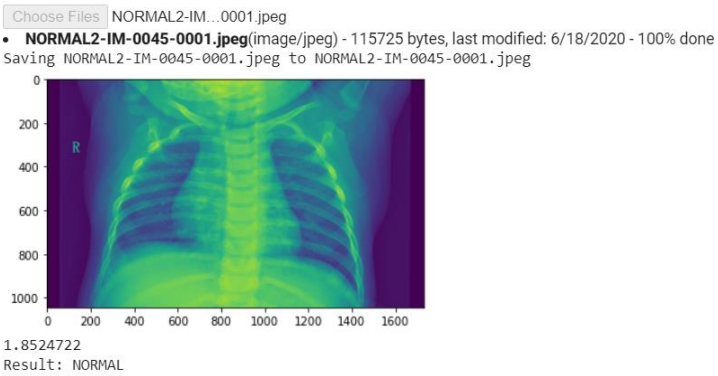


The above image is from the Val directory and Normal sub-directory and the model correctly predicts the class of the image (Normal).



The above image is from the test directory and PNEUMONIA sub directory and the model correctly predicts the class of the image (PNEUMONIA).

The above image is from the Val directory, and PNEUMONIA sub-directory, and the model correctly predicts the class of the image (PNEUMONIA).



The above image is from the test directory and Normal sub-directory and the model correctly predicts the class of the image (Normal).

This research will go a long way in improving the health of patients in low-income and under-developed environments. This study was, however, limited by the depth of data.

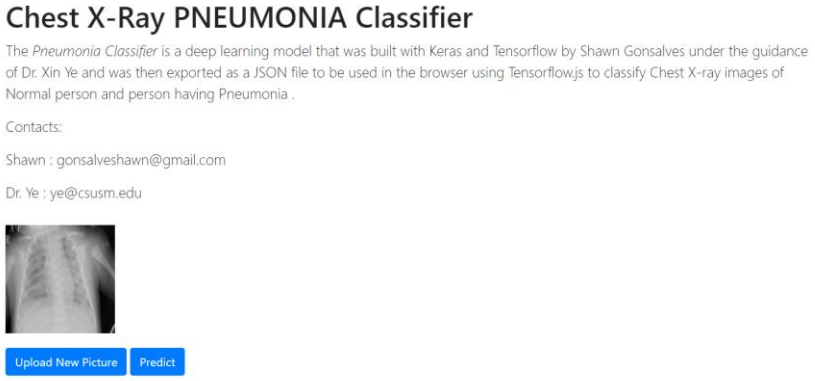
```
total_train  = len(os.listdir(      train_normal_dir ) ) + len(os.listdir(      train_pneumonia_dir ) )
print('Total training images are:', total_train)
total_test  = len(os.listdir(      test_normal_dir ) ) + len(os.listdir(      test_pneumonia_dir ) )
print('Total test images are:', total_test)
```

```
Total training images are: 5216
Total test images are: 624
```

The dataset was limited to 5216 training images and 624 for the testing. However, Obtaining a comparatively larger dataset of patients and non-patients from different parts of the world, the model can be improvised significantly.

4. Deployment

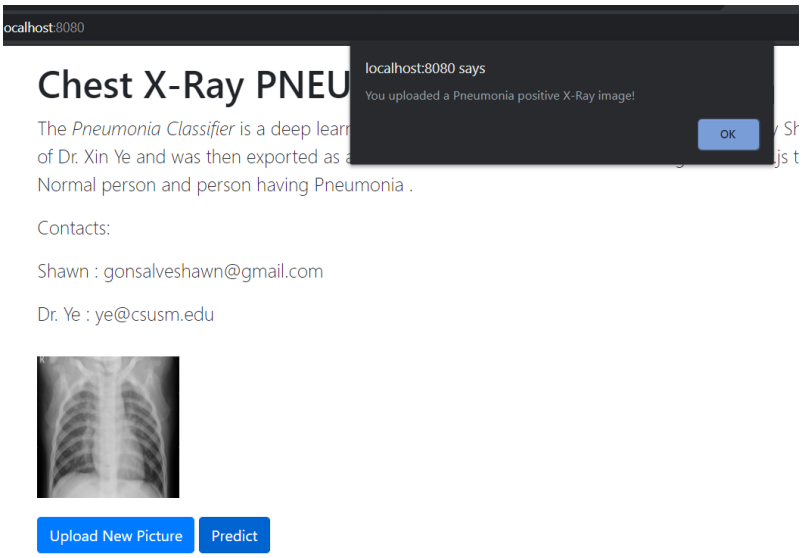
The trained model is then converted to a .JSON file using the TensorflowJS-converter. [11] This converter function saves the .JSON model along with some sharded weights associated with the model.json file in a binary format, all of which are saved in the path specified. TensorFlow.js provides model loading APIs that can be used to fetch these model assets and run inference in the browser.

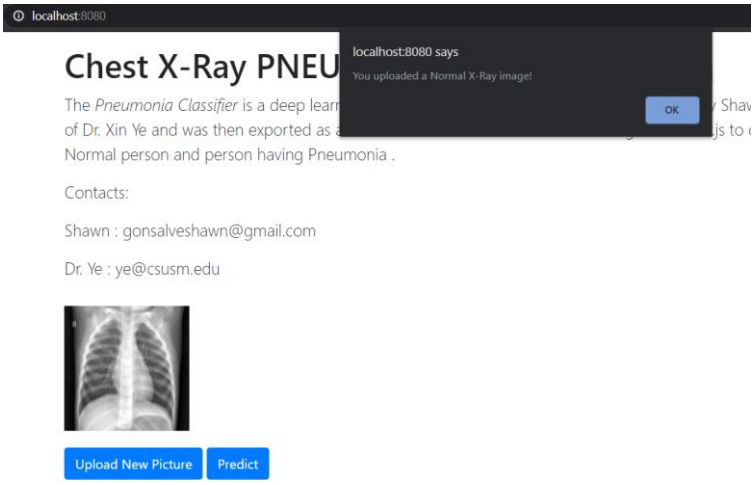


To load the JSON file we use "tf.loadLayersModel()" with the path for the specified model. The model is hosted on a web server along with sharded weights, and the path where it is hosted is provided as a path link in the javascript file.

After successfully loading the model in the browser, next up is the inference part. For the prediction, we need to load the image into the browser. This is done by using the getElementById() method.

Next up is creating a tf.Tensor from the input image that has a shape and dataType. This is performed using tf.browser.fromPixels() method provided by Tensorflow.js. The next process is resizing images to size using bilinear interpolation. After resizing and normalizing the image tensor, the image is fed into the model.predict() method as a 4D tensor by specifying its shape and image size. This resulting array of prediction is of 2 classes and the higher values specifying if the input image belong to the Normal class or a Pneumonia class.





Once a new picture has been uploaded, we use the Predict button to do the prediction of the image. To compare the prediction, we check if the predicted value is positive or negative. According to the labeling, the prediction array will consist of [Normal, Pneumonia] if the value is higher for the prediction[0], it is a Normal image else it is a Pneumonia positive image. As shown in the above picture, it sends out an alert to the tab in the browser where the inference takes place displaying where the input image provided is a case of Normal image or a Pneumonia positive image.

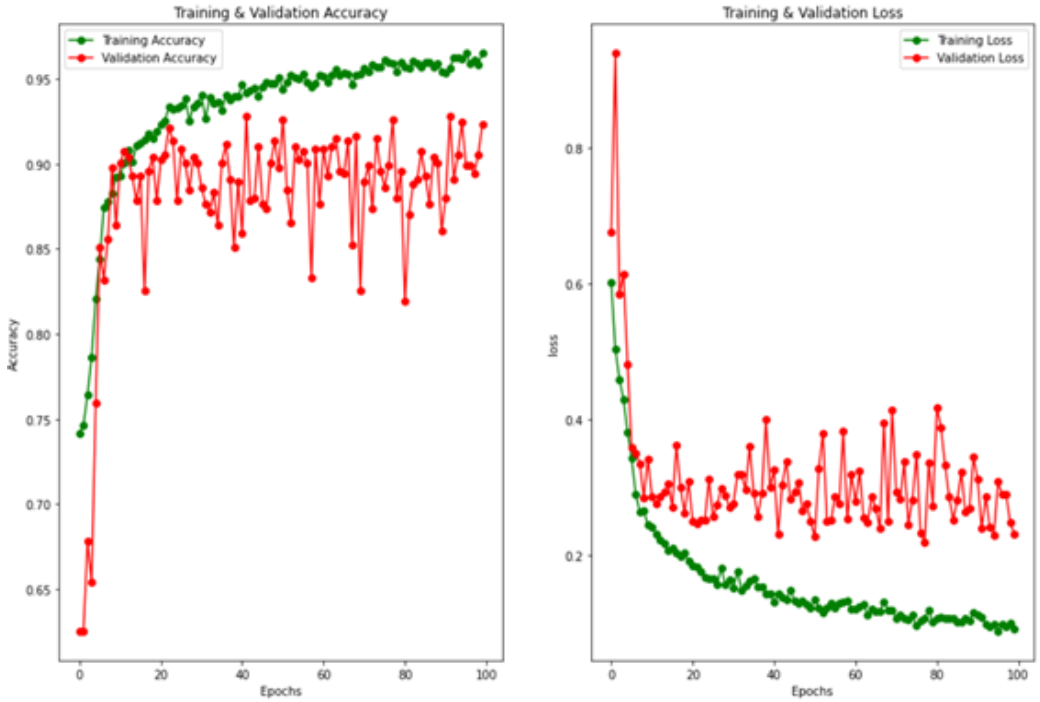
5. Conclusions.

The trained model reported impressive results when it was trained for 60 epochs. The resulting training accuracy was 0.9484, and training loss was 0.1260. [12]Accuracy is the number of correctly

predicted data points out of all the data points. More formally, it is defined as the summation of number of true positives (TP) and true negatives(TN) divided by the summation of number of true positives(TP), true negatives(TN), false positives(FP), and false negatives(FN).

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}$$

While designing and configuring the architecture of the Machine Learning model, it is required to select a loss function to train our neural network model to the optimal. For the loss function the metric used is SparseCategoricalCrossentropy, as this Research is a type of a classification problem where the output is interpreted as probabilities of membership between the classes**.** Loss is a value that portrays how poor the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is higher. The goal of training a model is to find a set of weights and biases such that the loss attains its minimum value, on average, across all examples.



The validation loss and validation accuracy were also impressively high due to the addition of a Dropout layer on the Normalized data which turned off random neurons in the network and the Data preprocessing which helped the model to not memorize the training data and ultimately not over-fit. This resulted in the model achieving the Validation accuracy as high as 91.35% and validation loss as 0.24 as shown in the figure below.

```
[ ] result  = model.evaluate_generator(test_data_gen, steps=len(test_data_gen), verbose=1)
    print('Loss:', result[0])
    print('Accuracy:', result[1])

    7/7 [==============================] - 4s 590ms/step - loss: 0.2461 - accuracy: 0.9135
    Loss: 0.24607081711292267
    Accuracy: 0.9134615659713745
```

This Research demonstrates how to build a model that could predict the Pneumonia X-ray images as positive or negative. Our Research depicts on the process to design a model that is built from scratch by specifying each layer in the network and does not use any pre-trained model, so we need not apply a transfer learning approach. Then the trained model is evaluated using separate unseen data to avoid biased predictions.

The future work for this research will be to detect the presence of COVID-19 using frontal X-Ray Images of patients. Distinguishing frontal X-ray images that contain Coronavirus and pneumonia has been a big issue in recent times, and our next approach will tackle this problem.

References

1. The Radiology information for Patients; Radiological Society of North America (RSNA); "Pneumonia - RadiologyInfo.org." 23 Jan. 2019, https://www.radiologyinfo.org/en/pdf/pneumonia.pdf.
2. "Pneumonia - Diagnosis and treatment - Mayo Clinic." https://www.mayoclinic.org/diseases-conditions/pneumonia/diagnosis-treatment/drc-20354210.
3. "ArIES, IIT Roorkee – Medium." https://medium.com/@ariesiitr/an-artificial-neural-network-ann-is-a-computational-model-that-is-inspired-by-the-way-biological-c17b07166d4c.
4. "Chest X-Ray Images (Pneumonia) | Kaggle." https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
5. .A. W. Harley, "An Interactive Node-Link Visualization of Convolutional Neural Networks," in ISVC, pages 867-877, 2015
6. J. Browlee, "A Gentle Introduction to Transfer Learning for Deep Learning." https://machinelearningmastery.com/transfer-learning-for-deep-learning/.
7. "Chest X-rays Pneumonia Detection using Convolutional ...." https://towardsdatascience.com/chest-x-rays-pneumonia-detection-using-convolutional-neural-network-63d6ec2d1dee.
8. Model Overfitting; "Overfitting - Wikipedia." https://en.wikipedia.org/wiki/Overfitting.
9. J. Browlee, "A Gentle Introduction to Dropout for Regularizing Deep Neural Networks," p. 3, 3 December 2018. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.
10. "Using the SavedModel format | TensorFlow Core." 16 Jul. 2020, https://www.tensorflow.org/guide/saved_model.
11. "Model conversion | TensorFlow.js." 08 Apr. 2020, https://www.tensorflow.org/js/guide/conversion.
12. "Accuracy (error rate) Definition | DeepAI." https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate.
13. ."Computer-aided diagnosis in chest radiography for ...." 01 Aug. 2008, https://www.sciencedirect.com/science/article/pii/S1386505607001803.
14. Avni, U., Greenspan, H., Konen, E., Sharon, M., & Goldberger, J. (2010). X-ray categorization and retrieval on the organ and pathology level, using patch-based visual words. IEEE Transactions on Medical Imaging, 30(3), 733-746.
15. Jaeger, S., Karargyris, A., Candemir, S., Folio, L., Siegelman, J., Callaghan, F., ... & Thoma, G. (2013). Automatic tuberculosis screening using chest radiographs. IEEE transactions on medical imaging, 33(2), 233-245.
16. Saul CJ, Urey DY, Taktakoglu CD. Early Diagnosis of Pneumonia with Deep Learning. arXiv preprint arXiv:1904.00937. 2019 Apr 1.
17. Antin B, Kravitz J, Martayan E. Detecting Pneumonia in Chest X-Rays with Supervised Learning.
18. Yao, L., Poblenz, E., Dagunts, D., Covington, B., Bernard, D., & Lyman, K. (2017). Learning to diagnose from scratch by exploiting dependencies among labels. arXiv preprint arXiv:1710.10501.
19. Boussaid, H., & Kokkinos, I. (2014). Fast and exact: ADMM-based discriminative shape segmentation with loopy part models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4058-4065).
20. Gaobo Liang and Lixin Zheng. 2020. A transfer learning method with deep residual network for pediatric pneumonia diagnosis. Comput. Methods Prog. Biomed. 187, C (Apr 2020). DOI:https://doi.org/10.1016/j.cmpb.2019.06.023