Predicting Pneumonia from X-ray Images

Raúl Castellanos
raulcastellanos@uchicago.edu

Claire Boyd
ckboyd@uchicago.edu

Benjamín Leiva
bleiva@uchicago.edu

Jack Gibson
jpgibson@uchicago.edu

Abstract

Pneumonia identification can be facilitated through the application of Machine Learning models. We worked with a Kagle dataset of 5,856 X-ray images comprising both normal and pneumonia cases. To ensure balanced representation, the dataset was rebalanced with an 80-10-10 split for training, testing, and validation respectively. The evaluation of the models' performance relies on two commonly used performance metrics: recall and accuracy. Motivated by the need for more reliable diagnostic criteria, this paper reviews various computer vision methods and identifies the Convolutional Neural Network (CNN) and Logistic Regression approaches as the most promising techniques for this task. After implementing these two models (CNN and Logistic Regression), conducting hyperparameter tuning, and testing various data transformations, the results of eight models show that logistic regression is most effective at predicting pneumonia with a recall of 99.4%. Additional research can explore if this may be due to sample size, number of epochs, or the success of certain image augmentations.

Introduction

Pneumonia, a detectable disease, poses a significant health concern. Pneumonia has an incidence rate of 24.8 cases per 10,000 adults in the United States, which increases with age. Chest X-rays have been widely used for pneumonia detection, however the interpretation of these images by doctors can be subjective and prone to variability. In recent years, the application of machine learning models has emerged as a promising approach to more objectively diagnose pneumonia.

Our project aims to provide a comprehensive analysis of pneumonia detection, focusing on testing different computer vision approaches to accurately predict pneumonia in chest X-ray images of children. The preliminary sections of this paper outline the motivation behind using machine learning models for this task, present various methods employed in previous studies, and emphasize the superiority of the Convolutional Neural Network (CNN) approach for pneumonia detection. Additionally, we discuss commonly used machine learning metrics within the medical field.

The subsequent sections of this paper walk through the approach our team took to build an effective model to classify pneumonia, organized as follows: The Data section provides insights into the original dataset, along with our rationale for rebalancing it to create a new dataset with an 80% training, 10% testing, and 10% validation split. In the Model section, we introduce the two implemented models, CNN and Logistic Regression and discuss the hyperparameter tuning process. Furthermore, we discuss the four transformations applied to the images, which include a base transformation, contrast enhancement, color augmentation, and a combined approach to augmenting the input images. Finally, we conclude by presenting accuracy, recall, and runtime measures for each method, reporting that logistic regression was most effective at predicting pneumonia with a recall rate of 99.4%. Code to replicate our evaluation process with our models can be found here.

Motivation

What is pneumonia and how can it be detected using X-rays?

According to the National, Heart, Lung and Blood Institute pneumonia is "an infection that affects one or both lungs. It causes the air sacs, or alveoli, of the lungs to fill up with fluid or pus. Bacteria, viruses, or fungi may cause pneumonia."[1] According to Regunath and Oba the incidence of this disease in the United States is 24.8 cases per 10,000 adults with higher rates as age increases.[2] Pneumonia is the eighth leading cause of death and first among infectious causes of death.

To diagnose this disease doctors "review your medical history, perform a physical exam, and order diagnostic tests such as a chest X-ray".[3] According to Kundu et al (2021), chest X-ray imaging is the most frequently used method for diagnosing pneumonia.[4]

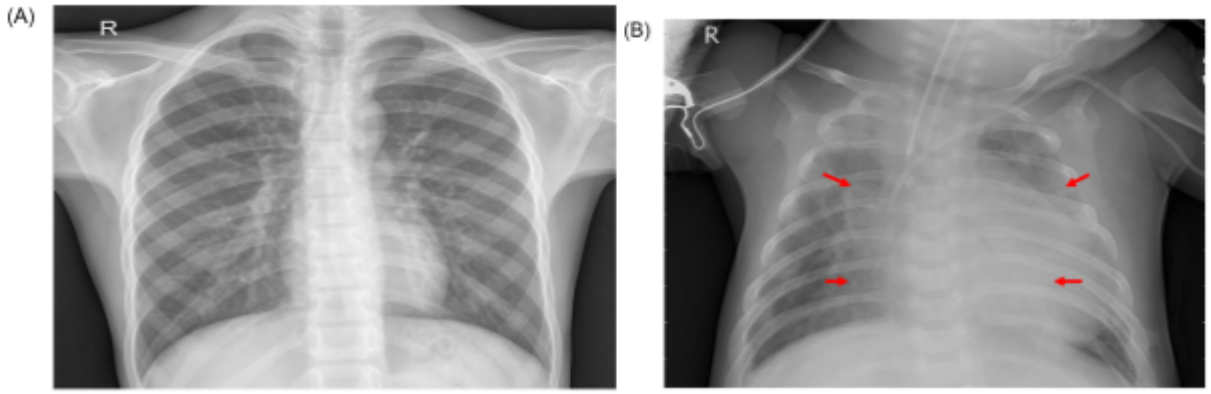


Figure 1: Examples of two X-ray plates that display (a) a healthy lung and (b) a pneumonic lung. Example taken from Kundu, et al.

Figure 1 shows an example of two X-rays: one with and one without pneumonia. Kundu et al (2021) explains how to read an X-ray image: "The white spots in the pneumonic X-ray (indicated in the right image with red arrows), called infiltrates, distinguish a pneumonic from a healthy condition."[5] These white spots are not present in a normal patient (left image).

However, chest X-ray examinations for pneumonia detection are prone to subjective variability.[6] For example, Neuman et al (2012) claims that there is "poor overall agreement by emergency medicine, infectious diseases, and pulmonary medicine physicians, and even radiologists, in their interpretation of chest radiographs for the diagnosis of pneumonia."[7] Due to the severity of pneumonia, failure to detect and treat positive cases could lead to additional health concerns that may be life threatening.

[1] National, Heart, Lung and Blood Institute
[2] Hariharan Regunath, Yuji Oba.
[3] National, Heart, Lung and Blood Institute
[4] Kundu, et al
[5] Kundu, et al
[6] Kundu, et al
[7] Neuman

Why do we need machine learning in this context?

The examination of chest X-rays is a challenging task and is prone to subjective variability[8]. Thus, an automated system for the detection of pneumonia could help add consistency to doctors' diagnoses of the infection. Amer Kareem explains that "technological advancements in artificial intelligence (AI) have proven to be helpful in the diagnosis of disease. For instance, techniques like CNN are utilized for classifying Chest X-rays in order to determine whether pneumonia is present. Some of the exciting research has been done in areas like abnormal-patterns detection."[9] In sum, predicting pneumonia is a good task for machine learning because it is a binary classification task (e.g. pneumonia or normal) and there are ample labeled datasets available to adequately train models.

Machine learning models for computer vision

Detecting medical images is a complex task that requires an effective computer vision approach. Below we explore two different types of machine learning models that are relevant for the task of detecting pneumonia in chest x-ray images.

Logistic regression

Logistic regression is a statistical model used to predict the probability of a binary outcome based on one or more independent variables. It is a type of regression analysis commonly used for classification problems, where the goal is to assign observations to one of two classes (in this case: pneumonia or normal). In terms of computer vision, logistic regression is often used as a "baseline" model that researchers use to compare the results of more complex models. This is the case because logistic regression has a simpler interpretation than other computer vision machine learning models and it can be more computationally efficient to use for larger datasets. For example, Nazish et al (2021) use logistic regression as a baseline model to compare against a support vector machine model in order to classify COVID X-ray images.[10] Similarly, Stokes et al (2021) use logistic regression as a baseline to diagnose pneumonia and bronchitis cases compared to more complex models.[11]

Convolutional neural networks (CNN)

Neural networks, also known as artificial neural networks (ANNs), are computational models inspired by the structure and functioning of biological neural networks. They consist of interconnected nodes called artificial neurons or units, organized in layers, and they are designed to simulate the information processing capabilities of the brain. They are especially helpful in computer vision learning tasks because they help break down complex images into further distilled versions of the image, learning and classifying along the way until making a final prediction.

Researcher Amer Kareem explains, in a recent study, that chest diseases like breast cancer, tuberculosis, and pneumonia can be effectively detected and diagnosed using artificial neural networks (ANN).[12] Preprocessing techniques, such as histogram equalization and image filtering, are

[8] Kundu
[9] Kareem
[10] Nazish
[11] Stokes
[12] Kareem

employed to enhance the imaging process by reducing noise and improving image clarity. A study using ANN to detect pneumonia achieved an accuracy of 92% using a feed-forward neural network with a dataset of 80 patients.[13] However, changes in the position and size of chest X-rays significantly affected the accuracy. While pattern recognition techniques are effective for medical image detection, the proposed method has limitations in detecting size and structural changes. Therefore, a neural network model capable of addressing these limitations should be developed.

The neural network type most relevant to this task is the convolutional neural network (CNN). CNNs perform well by identifying local patterns in images through multiple convolutional layers, which apply filters across the image to detect features at different scales. CNN works well in an X-ray context because CNNs use pooling layers to reduce the spatial dimensions while preserving the learned features, making them robust to small translations or shifts in the image. Figure 2 shows how a CNN is trained by transforming the input (in our case X-rays), into various parameters used to define the individual output. The idea of using the CNN approach is to limit the network distinctions between the predicted and actual outcomes.

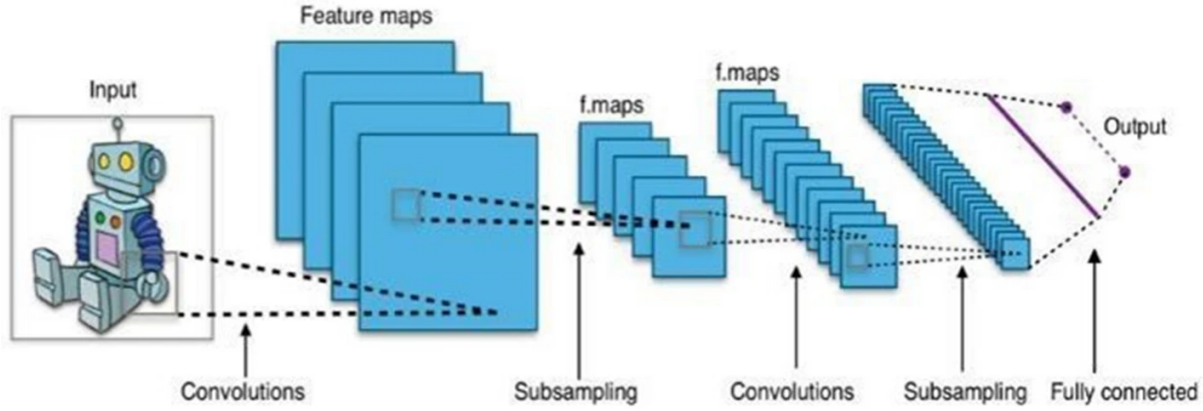


Figure 2:CNN architecture. Example taken from Kareem, et al.

Performance metrics in the medical context

Hicks explains that no single metric captures all the desirable properties of a model, which is why several metrics are typically reported to summarize a model's performance.[14] Depending on the classification task and the input data, the right performance metrics must be carefully and intentionally selected to ensure the model choice is optimizing for the right considerations. In the following section, we are going to discuss how accuracy and recall are appropriate metrics to use in the medical context.

Accuracy

Accuracy is the ratio between the correctly classified samples and the total number of samples in the evaluation dataset. Hicks explains that "this metric is among the most commonly used in applications

[13] Kareem
[14] Hicks

of machine learning in medicine, but is also known for being misleading in the case of different class proportions since simply assigning all samples to the prevalent class is an easy way of achieving high accuracy." Accuracy is calculated as follows:

$$Accuracy = \frac{correctly\ classified\ samples}{all\ samples} = \frac{TP + TN}{TP + FP + TN + FN} = \frac{True\ (Normal) + True(Pneumonia)}{Total}$$

Though it can be an important metric to use as a baseline and is easy to interpret, accuracy can give misleading results especially in the case that the input dataset is imbalanced. Optimizing for accuracy with an imbalanced dataset (unequal amounts of positive/negative cases in the labels) might skew the final model towards the majority label.

Recall

Hicks explains that recall is a metric that "denotes the rate of positive samples correctly classified, and is calculated as the ratio between correctly classified positive samples and all samples assigned to the positive class."[15] Recall can be measured as follows:

$$Recall = \frac{number\ of\ true\ positive\ samples}{number\ of\ samples\ classified\ positive} = \frac{TP}{TP + FN} = \frac{True\ (Normal)}{True(Normal) + False\ (Pneumonia)}$$

The author explains that this metric is one of the most important for medical studies, since it is desired to miss as few positive instances as possible, which translates to a high recall. In our calculation of this metric, we defined positive as being labeled 'normal' and negative as 'pneumonia.' With the vast majority of images in our dataset labeled as 'pneumonia,' the model was initially more likely to predict an image of 'pneumonia.' To assess the model's ability to correctly identify features of the non-majority label, our measure of recall calculates the total number of 'normal' cases it correctly identified out of the total number of images it classified as 'normal.'

Data

For this project, we used a dataset from Kaggle which contains 5,856 X-rays, manually labeled as either 'normal' or 'pneumonia.'[16] Each input image had three relevant characteristics:

- *Channel*: number of combinations of primary colors that determines color of image
- *Width*: horizontal number of pixels
- *Height*: vertical number of pixels

The combination of height and width determines the *size* of the image (i.e. the number of pixels it has). Pixels are the smallest addressable elements of an image, and are the units that store the information we need to predict the presence of pneumonia. Larger image sizes create more data for the model to learn on.

[15] Hicks
[16] Kaggle

Descriptive statistics

To better understand the data we are dealing with, we start by exploring descriptive statistics that are relevant for our purpose of predicting pneumonia. First, we take a look at the characteristics of our images:

	Mean	SD	Min	25%	50%	75%	Max	N
Channel	1	0	1	1	1	1	3	5,856
Height	971	383	127	688	888	1187	2713	5,856
Width	1328	364	384	1056	1281	1560	2916	5,856
Size	1,418,909	970,443	48,768	727,888	1,136,928	1,852,074	7,532,028	5,856

Table 1: Descriptive statistics of X-rays characteristics

As we can see, almost all images have only one channel, something that we expected due to the black-and-white nature of X-rays. In terms of size, the median image has dimensions 888x1281, which makes our X-rays more wide than tall. This makes sense given that these are taken to analyze the chest area and not the whole torso of the patient. If we plot the distribution of both width and height, we have a clearer visual of size composition:

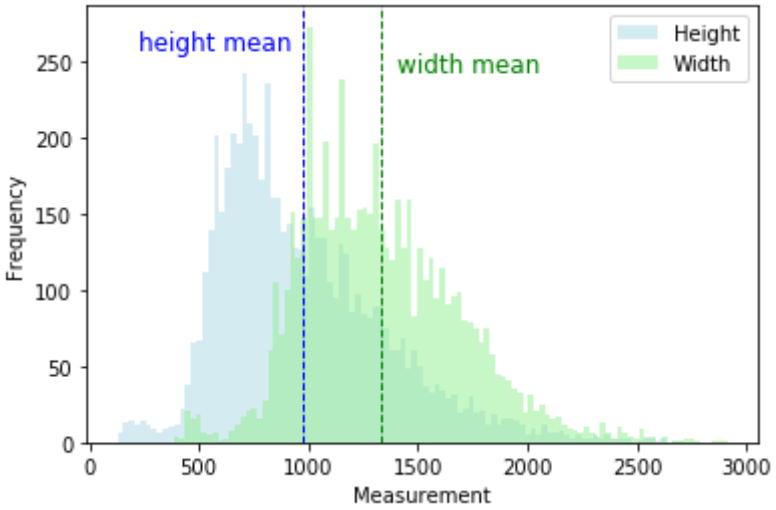


Figure 3: Distribution of X-ray's height and width

Additionally, the mean X-ray has around 1,400,000 pixels, providing ample potential features for the model to evaluate..

Then, we focus on the distribution of the labeling of our data. This is an important step because having a skewed distribution of labels will affect our ability to correctly predict a label given the uneven availability of X-rays for each diagnostic.

	NORMAL	PNEUMONIA
Count	1,583	4,273
Percentage	27%	73%

Table 2: Label distribution on full dataset

Looking at the table above, we observe that ‘normal’ X-rays represent 27% of our sample, while ‘pneumonia’ images take 73% of the dataset. This confirms the skewness of our labels. Now we need to check if this ratio propagates through our three subsets of data (training, validation, and testing). If that’s the case, then rebalancing our data splitting might improve the performance of our model.

Rebalancing

The 80-10-10 split is a common split used for machine learning tasks. According to Pragati Baheti: “The main idea of splitting the dataset into a validation set is to prevent our model from overfitting i.e the model becomes really good at classifying the samples in the training set but cannot generalize and make accurate classifications on the data it has not seen before.”[17]

The original dataset from Kaggle is organized into three folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). However, the order of this dataset does not follow the 80-10-10 split division. 89% of the original images are in the train dataset. Within this split, 25.70% of the images are ‘normal’ and the rest are labeled as ‘pneumonia.’ 10.65% of the images of the dataset are in the test folder and within this folder 37.5% of the images are labeled ‘normal.’ Finally, only 16 images are in our validation folder, this is only less than 1% of the total images in our dataset. A visual representation can be seen in the left part of Figure 4.

We decided to rebalance our data. Instead of treating all images in a single folder, we adopted a stratified approach to split the data. First, we divided the data into two groups: normal images (identified by starting with "IM-0" or "NORM") and pneumonia images (beginning with "pers"). Next, the images were randomly assigned, with 80% allocated for training, 10% for testing, and 10% for validation, within each group. A visual representation of this procedure can be observed on the right side of Figure 4.

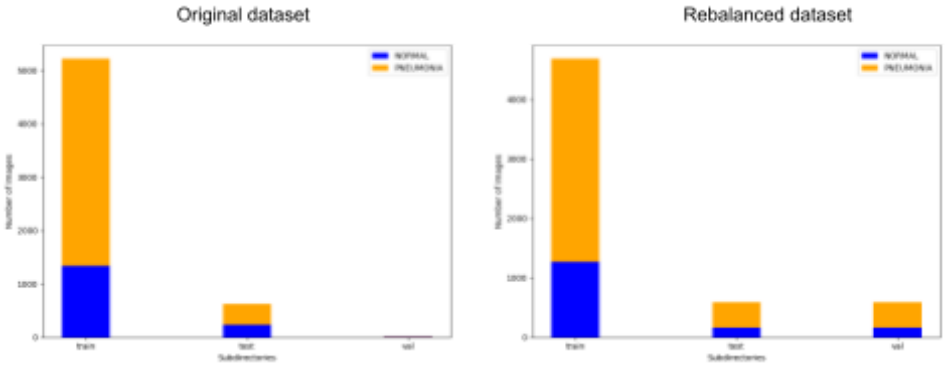


Figure 4: Original distribution of images and distribution after rebalance

[17] Pragati Baheti

Models

1. Convolution Neural Network (CNN)

As demonstrated by previous literature, convolution neural networks often serve as the gold-standard for computer vision techniques in the context of medical image diagnostics. In implementing our own version of a CNN optimized to perform on chest X-rays, we considered two primary attributes: computational expense and model performance. Drawing inspiration from past research, we considered a handful of network architectures including: GoogLeNet, AlexLeNet, ResNet, and LeNet-5. These models appeared frequently throughout similar literature examining the effectiveness of CNN in diagnosing diseases such as pneumonia and COVID-19 from chest X-rays.[18] A distinguishing feature among these models is the number of layers. For instance, a simpler model such as LeNet consists of 2 convolutional, pooling-layers with a flattened convolutional layer and 2 fully-connected layers, while more complex models such as ResNet can contain up to 75 hidden layers.[19] Although prior studies "indicate that deeper networks are more likely than shallower networks to learn relevant features"[20] when dealing with medical imaging, these added layers come with a costly expense in computation, leading us to choose a simpler network, LeNet-5, for this paper. Additionally there exists a wide body of literature supporting the use of the LeNet model on medical images and its success in computer vision tasks more broadly[21]

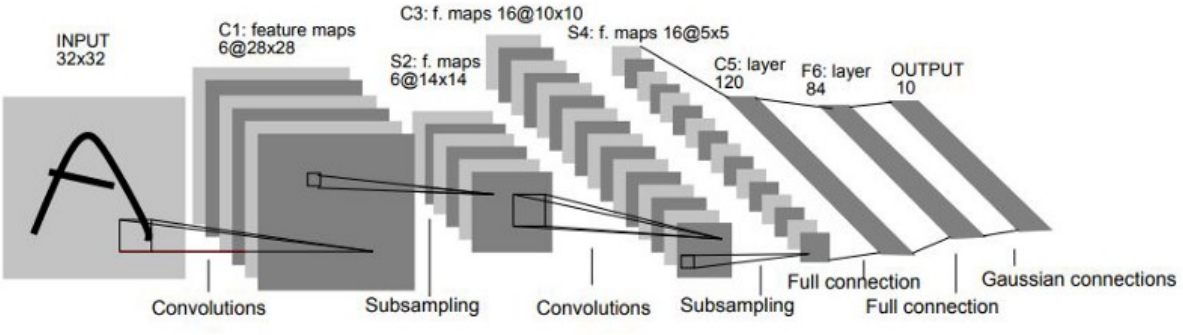


Figure 5

The structure of LeNet-5 is depicted visually in Figure 5.[22] Note that in our model, the input is a 256x256 image of a chest X-ray, however the rest of the model follows the structure outlined in Figure 5. The first convolution layer features 6 5x5 layers, which reduces the original image and adds layers to produce an output of 252x252x6. Following this a pooling layer is applied with a 2x2 filter that decreases our features by half, generating an output of 126x126x6. A second convolution layer with a 5x5 kernel is then applied with 16 filters, resulting in a 122x122x16 output. Max pooling with a 2x2 kernel is reapplied, creating a final output of 61x61x16 which results in 59,536 total features. After the convolution layers, the filters are flattened so that the image can linearly transform from 59,396 features to 120. Following an activation function, the image again undergoes linear transformation from 120 inputs to 84 outputs. The last fully connected layer transforms the 84 feature

[18] Kumar et al.
[19] Militante et al.
[20] Kumar et al
[21] Lecun
[22] A more detailed breakdown of the network's architecture can be found in Table 1 of the Appendix.

vector into 2 features, to allow the model to make binary classification of pneumonia or normal (0 or 1).

$$f'(x) = \begin{cases} 0 \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}$$

Figure 6

The activation functions between the convolution and pooling layers, and within the fully connected layers are ReLU (shown in Figure 6). Given the size of our images and the number of features in the model, we believed ReLU would better optimize the performance of the model due to its computational inexpensiveness. First unlike, sigmoid or other activation functions, ReLU does not require the use of an exponential calculation or a summation improving its computational efficiency. Furthermore, ReLU leads to increased sparasity within the layers of the network, as negative values are set to zero, increasing the network's ability to quickly learn relevant features within the images. We found the use of ReLU as the final activation function to also be appropriate, given that the model is tasked with binary classification. The use of sigmoid was considered in order to map inputs to a probability associated with the classification label. However, testing of the network featuring sigmoid, instead of ReLU, provided no noticeable increase in model performance[23] and led to a slight increase in computation time due the function's increased complexity.

2. Logistic Regression

We also chose to build a logistic regression model to serve as an outside (or "baseline") option for our neural network. This will help us put our main model's performance in context, specially in terms of its accuracy and recall scores, as well as its runtime.

This classifier works as follows. First, for every, X-ray, it takes its *j* feature's values (**x**) and applies a linear transformation to them using predetermined weights (β), while also adding a specific bias (β_0):

$$z = \beta_0 + \sum_j \beta_j x_j \quad (1)$$

Then, we take this result as an input for computing the image's probabilities of being a normal or pneumonia case, using a sigmoid (activation) function:

$$P(Y = 0|\, x) \;=\; 1 \,/\, \left(1 + e^{-z}\right) \quad ; \quad P(Y = 1|\, x) = \pi_i = \; e^z \,/\, (1 + e^z) \quad (2)$$

Once we have the probabilities of each label for every X-ray, we estimate a cross-entropy loss function with said probabilities as its inputs and, if desired, a regularization term (λ) which penalizes model complexity and overfitting. This whole step is commonly referred as 'forward propagation':

$$L = -\sum_i log P\left(y^{(i)}|x^{(i)}, \beta\right) + \frac{1}{2}\lambda \sum_j \beta_j^2 \quad (3)$$

[23] Comparing the results of the model trained on non-augmented images, the model predicted the test data with 90% accuracy when using ReLu, and 89% using Sigmoid. Additionally, the use of Sigmoid added around 2.5 minutes of run-time, which increased as the model was trained on the transformed image datasets.

Our main goal is to minimize this function, since that means we'll get the best predictions for our whole dataset. To do so, we apply a stochastic gradient descent approach to find the β's to minimize *L*. Then, for a certain step size or learning rate (η) chosen beforehand, we incur 'back propagation' by updating the values of β by estimating the following equation:

$$\beta_j^{l+1} = \beta_j^l - \eta\left(\partial L/\partial \beta_j\right) \tag{4}$$

where,

$$\partial L/\partial \beta_j = \sum_i \left[-\left(y_i - \pi_i\right)x_j\right] + \lambda\beta_j \tag{5}$$

is the partial derivative of our loss function *L* with respect to the weight of feature *j*. Finally, with our optimal β's at hand, for every X-ray we estimate *z* and use a softmax function to predict its label. Graphically, this whole process can be represented as:

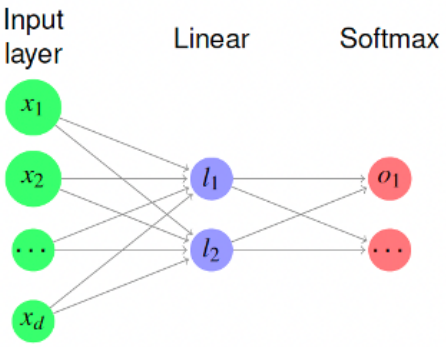


Figure x: Logistic Regression process

3. Hyperparameters

Hyperparameters were tuned using the convolution neural network, but were used across both the CNN and the logistic regression models to ensure consistency in our tests. With computational efficiency in mind, we chose to train our models in a 15 epoch period. We found this to be enough epochs to allow the model to converge across both the training and validation data. Given more computational power and time, running the model for more epochs (e.g. 30 to 50 epochs) would further optimize its performance.. However, the 15 epoch period provided enough training to provide consistent loss, accuracy, and recall metrics, which allowed us to meaningfully compare results between varying approaches to transforming our input images. The loss function used in both models was cross-entropy, which given the binary classification task at hand, efficiently computes loss when an image is predicted as 'pneumonia' or 'normal.'

With a relatively short training period, selecting the appropriate learning rate was critical to ensuring the model could converge within 15 epochs. We tested values ranging from 0.01 to 0.00001 to identify the optimal rate. Higher values (e.g. 0.01) created inconsistent adjustments in the weights and biases so that model could not converge to a stable point. Initially we landed on a rate of 0.00001, the results of which are featured in the left-hand column of Figure 1 in the Appendix. After further testing, this

rate proved unsuitable when we trained the model on the transformed images. The increased complexity caused by augmenting the images meant that the model could no longer effectively optimize its weights with this size learning rate in 15 epochs. To maintain the runtime of our model while adjusting for this, we evaluated under a slightly larger rate of 0.0001 which led to better convergence when the model was trained with transformed images. The results of the model trained with this rate using non-transformed and color-augmented images are depicted in the middle and right-hand columns of Figure 1 of the Appendix.

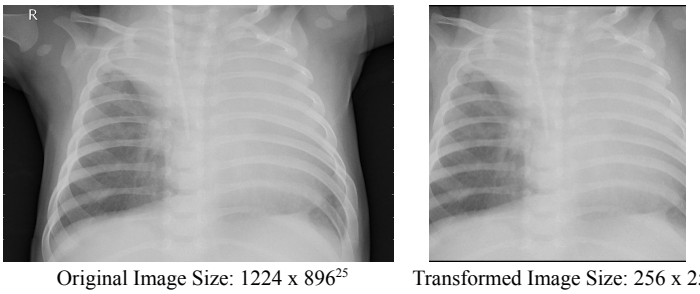
4. Image Transformations

A primary goal of this paper was to evaluate the effect of different image transformations on model performance. Stemming from the machine learning cliche 'garbage in, garbage out,' we believed that improving the quality of our inputs would boost performance while keeping computational costs low, as opposed to seeking increased performance through added hidden layers. To test this hypothesis we created a set of three transformations, plus a base case to serve as a control. These transformations are explained in brief in the following section.

Transformation 0: Base

Our initial transformation was designed to serve as a control case to evaluate the performance of more-highly transformed images against. It features no augmentation to the image's color, orientation, or contrast. Two modifications were made to the original images. First, all images were center cropped by a dimension of 712x712, which was done in an attempt to focus the model on the relevant parts of the image. Figure 7 shows that the cropped image no longer includes the exposed X-ray background featured in the original images and is centered more squarely on the chest. Second, all images were resized to 256x256, which was done to improve the runtime of our model. Initial testing[24] found that a 256x256 was the smallest image size that did not significantly hurt the performance of the model. Additional transformations to color and contrast were layered on top of this base transformation so that all images seen by the model were cropped and resized to the aforementioned specifications.

Figure 7: Base Image Transformation



Original Image Size: 1224 x 896[25] Transformed Image Size: 256 x 256

[24] Image sizes of 512x512, 124x124, and 64x64 were evaluated as well. Sizes below the 256 x 256 mark led to significantly worse performance in the 15 epoch training period. Sizing down from 512 x 512 to 256 x 256 displayed little impact on test performance while decreasing the run-time.
[25] Depicted image sizes are not to scale.

Transformation 1: Contrast Enhancement

The first transformation we evaluated increased the contrast of our original image. This was done to mimic common computer vision edge enhancements, such as Sobel edge detection and Canny edge detection[26] that have been suggested to improve a CNN's ability to classify images. In our attempts to implement similar algorithms, however, we were met with suboptimal results due to the poor image quality brought on by one, the grayscale aspect of the image which makes edges difficult to detect and two, the decreased pixelation caused by resizing the image. Instead, we increased the contrast of the X-ray to minimic edge detection. While this approach does not, in reality, detect the edges of the image's features, increasing the contrast does sharpen and clarify the edges between the brighter and darker areas of the image allowing the CNN to better recognize and separate between the relevant (brighter) aspects of the image and the background of the X-ray. An example of the output of this transformation is displayed in Figure 8.[27] Additionally, these transformations were implemented at random on a 0-10 scale, meaning that certain images received no increases to contrast, while other images were randomly assigned varying degrees of contrast enhancement. This was done to increase the variety of inputs the CNN witnesses in an attempt to speed the learning process by exposing it to a wider set of image features.

Transformation 2: Color Augmentation

Our second transformation leveraged morphological operations to augment the color of the x-rays. With all our images loaded as grayscale images, the color augmentation can be viewed as an alternative form of contrast enhancement in the fact that its aim is to sharpen the brighter (and more relevant) areas of the image, where pneumonia is likely to appear while lessening the darker aspects of the image. The key distinction in this approach compared to the contrast transformation is that the morphological transformation allows us to enhance the contrast of only the brighter areas of the image while subtracting darker elements from the image[28], unlike in the previous transformation which increases the contrast for the image as a whole. Additionally, the color transformations were applied consistently to all images unlike the contrast adjustment which were randomly varied.

Figure 8: Top-Hat and Black-Hat Transformation

$$A_{top} = A - (A \circ B) \quad (1)$$

$$A_{bot} = (A \bullet B) - A \quad (2)$$

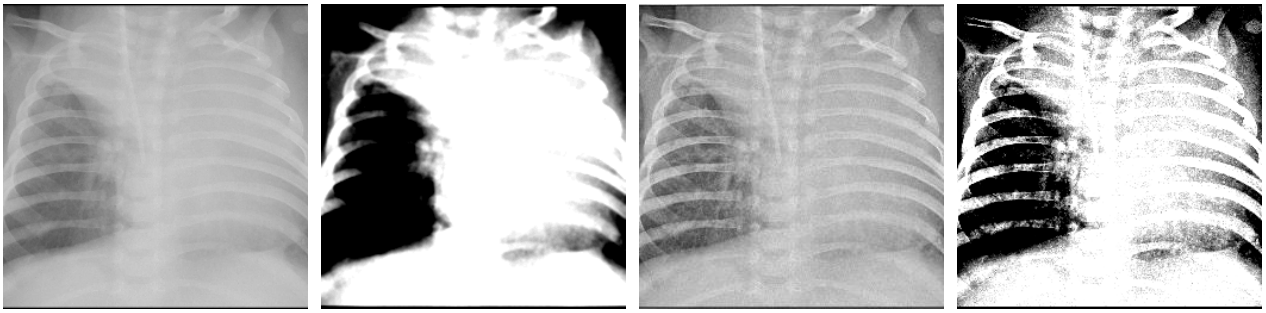
$$A_{enhance} = A + A_{top} - A_{bot} \quad (3)$$

The transformation process for this augmentation is described in equations in Figure 8, in which A is the input image and B is the structuring element, which our implementation is a 15x15 kernel. The structuring element is applied to the input image to create the top-hat layer and black-hat layer as described in the first and second equations. Then the top-hat layer is added to the input image to enhance the bright aspects of images while the black-hat layer is subtracted from the original image to

[26] Vairalkar and Nimbhorka, Othman et al.
[27] Code to replicate the example image outputs can be found here.
[28] Kushol

darken the dark regions of the image. An example of the resulting output image can be viewed in Figure 9.

Figure 9: All Transformation



Base | Base + Contrast | Base + Color | All

Transformation 3: Color and Contrast Augmentation

Our final transformation combined the two previous approaches into one joint transformation. The images undergo the morphological transformation described in the previous section, and then undergo a randomly applied contrast enhancement. The goal of this transformation is to test whether increased variation of the input images increases the model's performance on top of techniques such as contrast and color enhancement aimed at better focusing the CNN on the relevant areas of the image. Figure 9 depicts an example output of a chest X-ray having undergone this transformation.

Results and Conclusion

As discussed in the above sections, both the CNN and logistic regression models were trained in a 15 epoch period, with a learning rate of 0.0001, and then tested using the set-aside test data. This process[29] was repeated four times for each class of model to evaluate all the image transformation techniques. Accuracy and recall metrics calculated based on the test data are displayed in Figure 10.

Figure 10: Test Performance Metrics

	Method	**Recall (%)**	**Accuracy (%)**	**Run time**
LeNet-5 (CNN)	Base	73.4	90.9	~30 min
	Base + Color	88.6	92.3	~45 min
	Base + Contrast	67.1	89.4	~45 min
	All	70.3	88.7	~77 min
Logistic Regression	Base	99.4	80.9	~10 min
	Base + Color	98.5	82.5	~15 min
	Base + Contrast	91.6	87.6	~18 min
	All	96.3	86.7	~25 min

[29] Code to replicate our evaluation process with our models can be found here.

From the literature, we expected our convolutional neural network model to outperform logistic regression and originally only included logistic regression in our analysis plan as a baseline. However, using recall as our core metric of interest, the test performance across all eight models of interest show that the logistic regression outperforms the LeNet-5 model in every image augmentation method by 10-20%.

Reflecting on this performance, there are two core reasons why we think logistic regression may have performed better in this context. First, the transformation methods we used on the images before feeding them into the model could have obscured the relationships between pixels more than desired making them less effective in the CNN model. Second, the logistic regression might have performed better given the small sample size of our dataset compared to other studies well documented in the literature. Additionally, the quicker training period due to the low number of epochs (15) may have further improved its performance.

Furthermore, the results across both models show the trade off between additional image augmentation and training speed. The last column in Table 10 indicates the runtime needed to train the model and obtain results for recall and accuracy. Although there are many factors that may affect runtime (e.g. available space on local disc, working on a server, etc.), it is important to notice the trend in increased run time when the model includes additional image transformations. In selecting a model for use, researchers must often make a tradeoff between computational efficiency and accuracy, which is demonstrated by the results from the eight models above.

Lastly, in selecting the “best” model, it is important to weigh multiple accuracy metrics. Although we prioritize optimizing for recall, there are models in the above table that perform more consistently across both metrics. In a medical context, it may make sense to choose a model that has high recall without overly jeopardizing baseline accuracy. In this instance, that might mean the logistic regression trained on images with all augmentations might be the best model across the board, as it collectively maximizes both metrics.

In conclusion, the results of our eight models show that logistic regression is highly effective at predicting pneumonia with a recall of 99.4%. Further research can explore if this may be due to sample size, number of epochs, or the success of certain image augmentations.

References

Baheti, P. "Train, Validation, and Test Set Split: Why, How, and When?" Retrieved from https://www.v7labs.com/blog/train-validation-test-set#:~:text=The%20main%20idea%20of%20splitting,it%20has%20not%20seen%20before.

Hicks, S.A., Strümke, I., Thambawita, V., Hammou, M., Riegler, M.A., Halvorsen, P., Parasa, S. "On evaluation metrics for medical applications of artificial intelligence." Sci Rep, vol. 12, no. 1, Apr. 2022.

Kareem, A., Liu, H., & Sant, P. "Review on Pneumonia Image Detection: A Machine Learning Approach." Hum-Cent Intell Syst, vol. 2, pp. 31-43, 2022.

Kumar, A., Kim, J., Lyndon, D., Fulham, M., and Feng, D. "An Ensemble of Fine-Tuned Convolutional Neural Networks for Medical Image Classification." IEEE Journal of Biomedical and Health Informatics, vol. 21, no. 1, pp. 31-40, Jan. 2017.

Kundu, R., Das, R., Geem, Z.W., Han, G.T., Sarkar, R. "Pneumonia detection in chest X-ray images using an ensemble of deep learning models." PLOS ONE, vol. 16, no. 9, 2021.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

Lorraine, K.J.Silva, Teja, K., Devi, G., Harika, K. (2014). "Comparative Analysis of Various Edge Detection Techniques and Cancer Cell Detection using Sobel Algorithm."

National Heart, Lung, and Blood Institute. "Pneumonia." Retrieved from https://www.nhlbi.nih.gov/health/pneumonia#:~:text=Pneumonia%20is%20an%20infection%20that,or%20fungi%20may%20cause%20pneumonia.

Nazish, Ullah, S. I., Salam, A., Ullah, W., & Imad, M. (2021). COVID-19 Lung Image Classification Based on Logistic Regression and Support Vector Machine. In A. M. Musleh Al-Sartawi, A. Razzaque, & M. M. Kamal (Eds.), Artificial Intelligence Systems and the Internet of Things in the Digital Era (Vol. 239). Lecture Notes in Networks and Systems.

Neuman, M.I., Lee, E.Y., Bixby, S., Diperna, S., Hellinger, J., Markowitz, R., Servaes, S., Monuteaux, M.C., and Shah, S.S. "Variability in the interpretation of chest radiographs for the diagnosis of pneumonia in children." J. Hosp. Med., vol. 7, pp. 294-298, 2012.

Regunath, H., Oba, Y. "Community-Acquired Pneumonia." In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2023. Updated 2022 Nov 15.

Kaggle Dataset. "Chest X-ray Images (Pneumonia)." Retrieved from https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia.

Kushol, R., Nishat, R.M., Rahman, A.B.M.A., Salekin, M.M. "Contrast Enhancement of Medical X-ray Image Using Morphological Operators with Optimal Structuring Element." arXiv:1905.08545v1 [cs]. May 2019.

Stokes, K., Castaldo, R., Franzese, M., Salvatore, M., Fico, G., Gurbeta Pokvic, L., Badnjevic, A., Pecchia, L. "A machine learning model for supporting symptom-based referral and diagnosis of bronchitis and pneumonia in limited resource settings." Biocybernetics and Biomedical Engineering, vol. 41, no. 4, pp. 1288-1302, 2021.

Appendix

Table 1: Neural Network Architecture

Step	**Purpose**	**Output Dimensions**
Input	Feed images into the model	[1, 256, 256]
Convolutional layers		
Conv2d()	Splits data into 6 channels, using a 5x5 kernel. Because we use a step size of 1 and 0 padding, we only decrease the dimensions of height/width by 4 while increasing the number of channels to 6. Though this step originally increases the magnitude of our data (by increasing channels to 6), it is an important step to capture the trends across pixels with all available pixels	[6, 252, 252]
BatchNorm2d()	Normalizes the data by subtracting the mean and dividing by the standard deviation in each batch of the input data. This layer only changes the values within the layer, without altering the dimensions of the layer	[6, 252, 252]
ReLU()	ReLU (Rectified Linear Unit) works as an activation function to replace all negative values in a neural network layer with 0s (e.g. relu(x) = max(0, x). Similar to normalization, it only changes the values of the layer without altering its shape	[6, 252, 252]
MaxPool2d()	Takes the max of a kernel of data in order to simplify the layers dimensionally. Common parameters for this operation are setting both kernel and stride to 2 in order to simplify both the height and width of the input layer by 2, effectively minimizing the dimensions of the object by a factor of 2 (¼ of the input layer)	[6, 126, 126]
Conv2d()	*see above*	[16, 122, 122]
BatchNorm2d()	*see above*	[16, 122, 122]
ReLU()	*see above*	[16, 122, 122]
MaxPool2d()	*see above*	[16, 61, 61]
Fully connected layers		
Flatten()	This is a module that flattens the output of the convolutional layers to a one-dimensional tensor. It's used after the second set of convolutional and pooling layers to convert the output from a 16x61x61 tensor to a 1D tensor of length 16x61x61=59,536	[64, 59536]

Linear()	Applies linear transformation to map 59,536 features into 120	[64[30], 120]
ReLU()	*see above*	[64, 120]
Linear()	*see above* This layer further reduces the dimensionality of the feature representation	[64, 84]
ReLU()	*see above*	[64, 84]
Output layer		
Linear()	*see above* This module creates the output layer with 84 input features and 2 output features. The two output features correspond to the two classes in the binary classification problem: normal and pneumonia	[64, 2]

[30] Represents a batch size of 64 which remains constant across all the fully connected and output layers.

Figure 1: Training and Validation Performance Metrics