Changes to Testing Data and its Effect on Robustness of Classification Models

*Presented By: Himanshu Badera*

*Engineering Data Science Department*

*University of Houston*

*Houston, TX*

[*hbadera@uh.edu*](mailto:hbadera@uh.edu)

***Abstract* —**

**This study is to explore the effects of three data augmentation techniques on classification of chest x-rays. We used a data classification method to analyze structured or unstructured data and organize it into categories based on file types and contents. In the context of computer vision, data augmentation is changing the images from their original state to augmented state. Data augmentation can be used to change the accuracies of a data classification model used in machine learning. This research is to study the effects of three types of data augmentation on classification models used on datasets comprised of four types of chest x-rays: Normal, Covid-19, Pneumonia, and Tuberculosis. The three data augmentation methods used were changing the tint, applying a gaussian blur, and applying a lens flare. Our research used both a DenseNet framework and a Convolutional Neural Network to obtain results. This study found that the three data augmentation techniques used did not increase the accuracy of the classification compared to the accuracy when the models were trained on data that had not undergone data augmentation.**

***Keywords — Deep Neural Network, DenseNet, CNN, Covid-19, CXR, Gaussian Blur, Tinted Images, Lens Flare***

**I PROBLEM STATEMENT**

Changes to Testing Data and its Effect on Robustness of Classification Models

**II INTRODUCTION**

Covid-19 is a respiratory illness discovered in 2019 that is caused by the SARS-CoV-2 virus [1]. Tuberculosis is a disease caused by Mycobacterium tuberculosis. Tuberculosis tends to affect the lungs of the person infected [2]. Pneumonia is an infection of the lungs [3]. Tuberculosis, pneumonia, and Covid-19 can be seen on chest x-rays. The early availability of public datasets of CXR images from COVID-19 subjects fueled almost entirely research on assisted COVID-19 diagnosis. Using these data can also cover the limitations of other tools. The use of X-ray images has the advantage of being available in most hospitals and laboratories, as well as being easy. In the absence of common symptoms like fever, X-ray images of the chest can detect the disease well. Given that analyzing chest X-ray images is one method of diagnosing COVID-19, the use of computer vision and Deep Learning can aid in the diagnosis of this disease. Since the disease spread, many researchers have used machine learning, Computer Vision and Deep Learning methods with positive results. Because of the sensitivity of the Covid-19 diagnosis, one of the major challenges we face in our research is determining the accuracy of the diagnosis. Even though CT scan has been shown in several studies to be useful for detecting COVID-19 and pneumonia, CT scans are not suitable for COVID-19 screening due to the high cost and radiation exposure. Chest X-ray imaging (CXR), on the other hand, is inexpensive and widely used for screening. The number of publicly available CXR images of COVID-19 and pneumonia was limited, so we developed a CNN model that could be accurate and robust even with a small amount of CNN training data. The proposed method included transfer learning, which uses CNN models that have been pre-trained on a large dataset to improve accuracy and robustness. On CXR images, the model was tested to see if it distinguishes COVID-19, pneumonia, tuberculosis, and normal x-rays. In this paper, we develop and test a model that predicts the classification of COVID-19, pneumonia, normal and tuberculosis based on CXRs, which will be used as an aid in patient care management. In addition, we propose a method for training the network when the dataset is unbalanced. The goal of this study is to see how three different methods of data augmentation affect classification models on datasets with four different types of chest x-rays: Normal, Covid-19, Pneumonia, and Tuberculosis. Combining two robust deep convolutional neural networks and optimizing the training parameters accomplishes this. Due to the scarcity of open-source data, we are concentrating on improving detection efficiency.

**III LITERATURE REVIEW**

The proposed multi-network architecture's training and validation makes use of multiple datasets, which are described further below.

The overall chest X-ray dataset was obtained from [4]'s Kaggle. This dataset includes Chest X-ray images of patients infected with COVID-19, Pneumonia, Tuberculosis and ones who are not infected (Normal). We looked at the x-ray images in this dataset, and there were 7097 images available with COVID-19, Pneumonia, tuberculosis, and Normal.

The data augmentation techniques used are adding a tint to the chest x-ray, adding a gaussian blur, and adding a lens flare. No image had multiple data augmentation techniques used on it.

The datasets are named after what types of data augmentations were used on the training images. The Gaussian Blur was an effect applied in the photo editing software. The size of the effect for both X and Y was 1.5 as are the GIMP presets. The Tint effect was applied in GIMP by creating a new layer as an overlay. Six colors were applied to images: e08be1, 5d621e, 3d7e74, bba83b, c3494d, and cb6207. Any image that was tinted only had one color as a tint. The Lens Flare data augmentation technique was applied through GIMP at 0.5 for both the X and Y positions. The Lens Flare effect also had a radius of 3. The datasets created from these were then added to each other to create data sets such as the “Lens Flare and Gaussian Blur” data set. The sets’ test sets were mildly modified to keep an 80% and 20% mix with the train and test sets.

The use of Gaussian blur helps in reduction of possible distinct features such as letters or labels on images. Tinted filters help in removing any biased color patterns in the images and lens flare helps in removal of possible bias around the sternum.

**IV EXPERIMENTAL SETUP AND COLLECTION OF DATA**

Initially, I had a total of 7097 images of Chest X-Rays of Covid-19, Pneumonia, Tuberculosis and Normal healthy. I used various software to trim, crop, and blur, etc. the images as shown in Table 1. This is the dataset used to check the accuracies in this Project. I divided the images into multiple sub-categories and conducted the experiment on them. The dataset is shown in the table below.

|  |  |
| --- | --- |
| **Set** | **Images** |
| *Gaussian Blur* | 117 |
| *Tinted Images* | *173* |
| *Lens Flare* | *100* |
| *Lens Flare and Gaussian Blur* | *213* |
| *Lens Flare and Tinted Images* | *272* |
| *Tinted Images and Gaussian Blur* | *287* |
| *All Changed Images* | *389* |
| *Original Dataset* | *7097* |

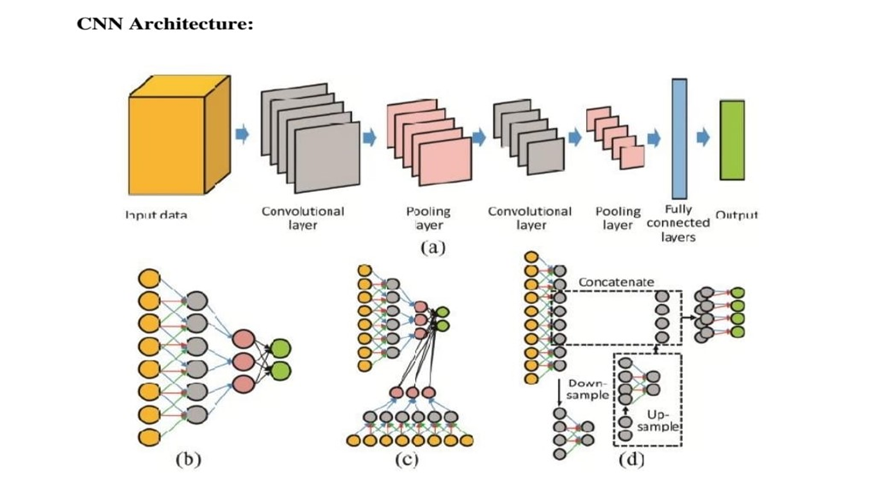
**Table 1 – Dataset**

**V PROPOSED METHOD**

In this Project, I used the DenseNet model and CNN model from Keras library and imported the original dataset [5].

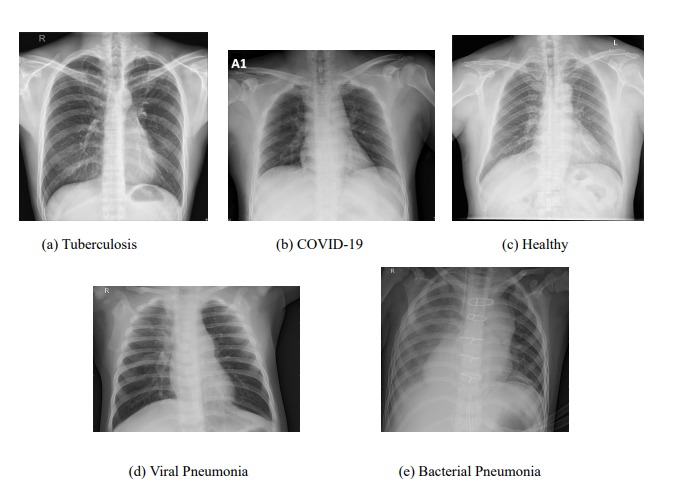
The way CNN model behaves is that it imitates the human brain. It recognizes patterns, image classification, and natural language processing which can be used for image processing. It contains Convolutional layers, max pooling layers and nonlinear activation layers. The main layer for CNN is the convolutional layer which performs an operation called convolution which generates a feature map. The ReLU function is also used with the convolutional layer which increases the nonlinearity of the input x-ray image.

The architecture used for the Convolutional Neural Network consists of 6 layers consisting of alternating sequences of convolution (CONV) and pooling (POOL) layers. The first 2D convolution layer comprises a filter of size 7x7 and stride of 2 and the second layer comprises filter of size 5x5 and stride of 2. Varying number of channels are used – 128, 64 respectively. 4×4 max pooling layers with stride of 2 are used. In all but the last fully connected layer, a RELU activation function is used. A SoftMax activation function is used for the final layer.



**Figure 1**

The DenseNet architecture proposed by Huang [6] a is computationally efficient and demonstrates representation effectively. The main feature of this architecture is that the feature maps in the current layer are concatenated with all the preceding layers as shown in figure 2. Due to that the number of channels is reduced and accommodated in the convolutional layers and the trainable parameters are diminished which results in efficient computation. Also, the concatenation of the feature maps from the preceding layers demonstrates representation effectively.

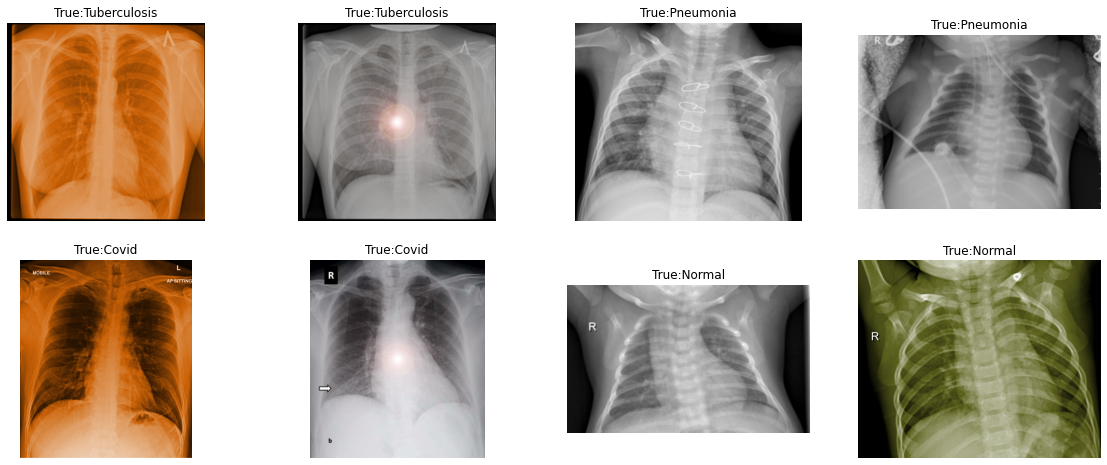


**Figure 2 – Original X-Rays**

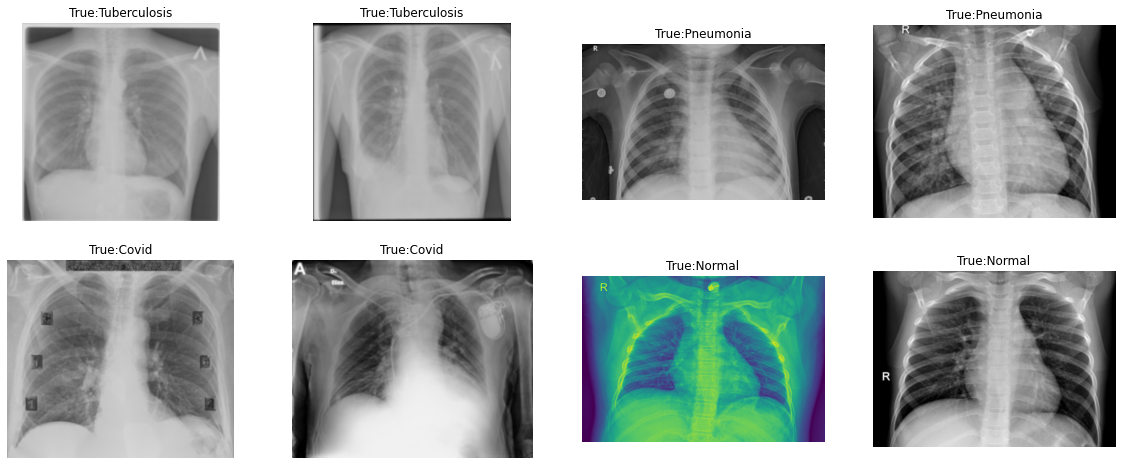
**VI** **DATA PREPROCESSING**

All X-ray scans were down sampled and downsized to 128x128x3 pixel dimensions to improve model generalization and processing speed. To be compatible with the most prevalent transfer learning models, the images were transformed to RGB scale and their labels were one-hot encoded.

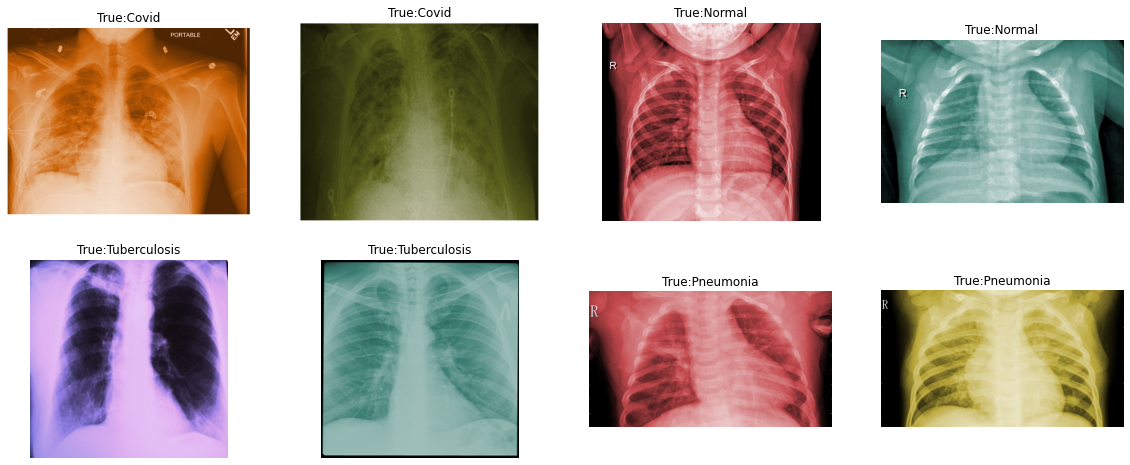
The data was then divided into three sets: training, validation, and test. Across all split datasets, the same illness distribution was maintained. There were 7152 photos in the training set, 1574 in the validation set, and 776 in the test set.



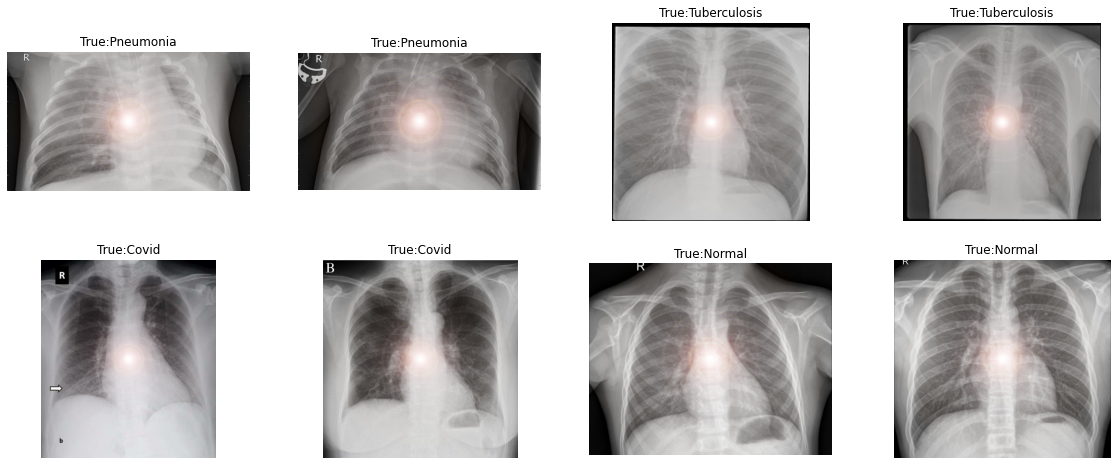
**Figure 3 – Images after preprocessing (All changed)**



**Figure 4 – Gaussian Blur**

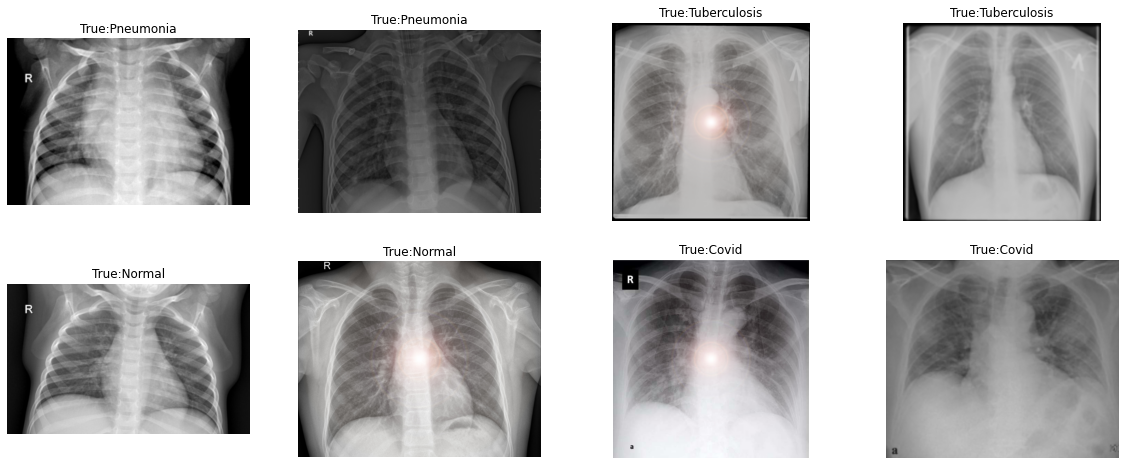


**Figure 5 – Tinted Blur**

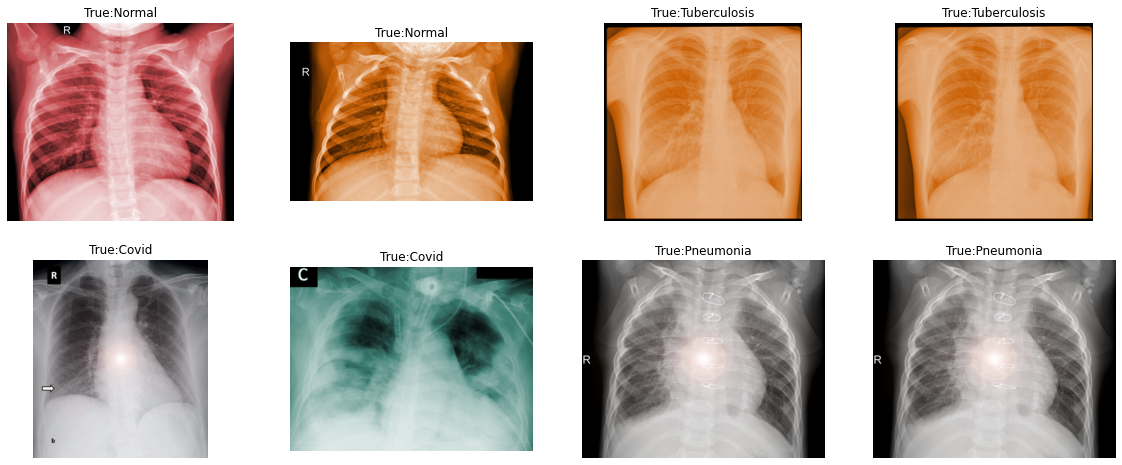


**Figure 6 – Lensflare Blur**

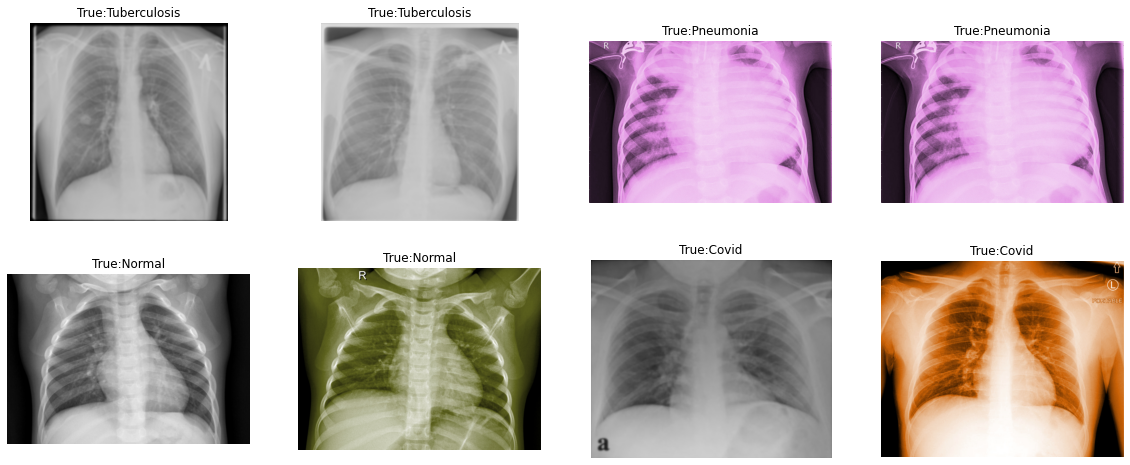
***After that, I mixed two blurred images and made a single one***



**Figure 7 – Lensflare and Gaussian Blur**



**Figure 8 – Lensflare and Tinted Blurred**



**Figure 9 – Tinted and Gaussian Blur**

***The below table shows different blurred images of Covid-19, Tuberculosis, Pneumonia, and Tuberculosis.***

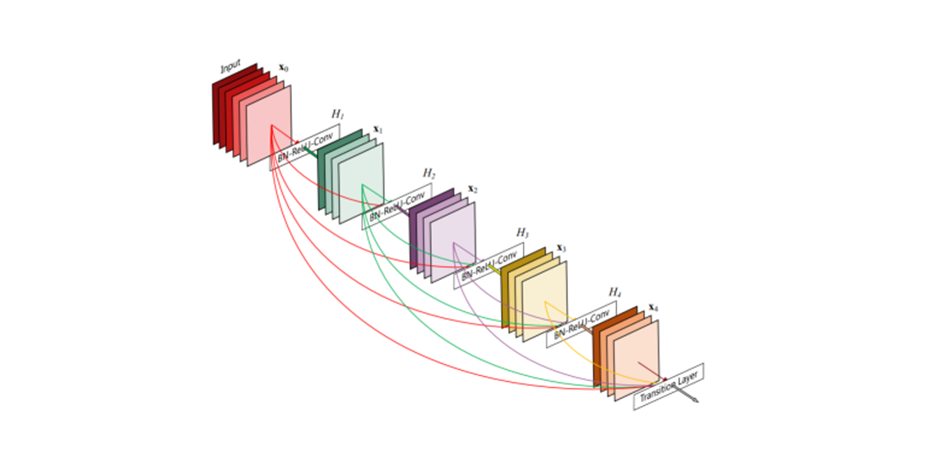
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Covid-19** | **Normal** | **Pneumonia** | **Tuberculosis** |
| *All Test* | *20* | *22* | *60* | *16* |
| *All Train* | *80* | *90* | *77* | *64* |
| *Gaussian Test* | *6* | *6* | *6* | *6* |
| *Gaussian Train* | *24* | *24* | *23* | *22* |
| *Lensflare Test* | *5* | *5* | *5* | *5* |
| *Lensflare Train* | *19* | *21* | *20* | *20* |
| *Tinted Test* | *9* | *11* | *9* | *6* |
| *Tinted Train* | *37* | *45* | *34* | *22* |
| *Lensflare + Gaussian Test* | *10* | *10* | *10* | *10* |
| *Lensflare + Gaussian Train* | *43* | *45* | *43* | *42* |
| *Lensflare + Tinted Test* | *14* | *16* | *14* | *10* |
| *Lensflare + Tinted Train* | *56* | *66* | *54* | *42* |
| *Tinted + Gaussian Test* | *15* | *17* | *14* | *10* |
| *Tinted + Gaussian Train* | *61* | *69* | *57* | *44* |

**Table 2**

**VII INNOVATION / THINGS WHICH I DID ON MY OWN (NEW)**

The presented model's convolutional layer was DenseNet-121, which was trained using the ImageNet dataset. The following algorithm and architecture were used for training:

1. DenseNet-121 is used to apply convolutions to training images, keeping the model weights the same and not changing them.
2. Use global average pooling in two dimensions
3. Make the data flat.
4. The ReLU activation function feeds data into a 32-node dense layer that is fully linked.
5. Final disease categorization, feed data into a fully connected dense layer with 5 nodes, using the softmax activation function.



**Figure 10**

The default Adam optimizer and categorical cross-entropy loss function were used due to the model's multiclass nature. Training was carried out over 20 epochs with a variable batch size.

In the epoch chosen for the final model, the training accuracy steadily increased to 88.71 percent. The accuracy of the validation was 82.97 percent, and the accuracy of the test set was 83.4 percent.

There have been a lot of experiment done in the past year but no one has ever tried to twist the data and then check the accuracies. I did this and got success in it.

I achieved our overall model accuracy for classification of 83.4%, which we realize could be improved and worked upon, however the classification accuracy for covid-19 chest x-rays was excellent with a precision of 0.95.

The model's inability to distinguish between bacterial and viral pneumonia accounts for the majority of the drop in accuracy. While this reduces the model's capacity to be used as a general-purpose diagnostic tool across all three lung classes, it has no effect on its core goal of detecting COVID-19 infections.

**VIII CODES**

**import** numpy **as** np *# linear algebra*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

**import** random

**import** os

**import** cv2

**from** IPython.display **import** Image

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

*#from keras.utils import plot\_model*

**from** sklearn.metrics **import** classification\_report

**from** collections **import** Counter

**import** tensorflow **as** tf

**from** keras.utils.vis\_utils **import** plot\_model

**import** keras

**from** tensorflow.keras.preprocessing.image **import** ImageDataGenerator

**from** keras.callbacks **import** ReduceLROnPlateau

**from** tensorflow.keras.applications.inception\_v3 **import** InceptionV3

**from** tensorflow.keras.applications.resnet50 **import** ResNet50

**from** tensorflow.keras.applications.vgg16 **import** VGG16

**from** keras **import** Model, layers

**from** keras.models **import** Sequential

**from** tensorflow.keras.optimizers **import** Adam, SGD

*#from keras.optimizers import Adam, SGD*

**from** keras.layers **import** GlobalMaxPooling2D, GlobalAveragePooling2D, Dropout, Dense, Input, Conv2D, MaxPooling2D, Flatten,MaxPooling3D

**from** keras **import** regularizers

**from** keras.layers **import** Activation,Dense

**import** pandas **as** pd

*#pull from datasets*

*# gauss*

*#seg\_train\_folders = '/content/gdrive/MyDrive/datasets/gaussian blur/Train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/gdrive/MyDrive/datasets/gaussian blur/Test/'*

*#seg\_pred\_folders = '/content/gdrive/MyDrive/val/'*

*#pull from datasets*

*# tint*

*#seg\_train\_folders = '/content/gdrive/MyDrive/datasets/tinted images/Train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/gdrive/MyDrive/datasets/tinted images/Test/'*

*#seg\_pred\_folders = '/content/gdrive/MyDrive/val/'*

*#pull from datasets*

*# lens*

*#seg\_train\_folders = '/content/gdrive/MyDrive/datasets/lens flare/Train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/gdrive/MyDrive/datasets/lens flare/Test/'*

*#seg\_pred\_folders = '/content/gdrive/MyDrive/val/'*

*#pull from datasets*

*# lensandgauss*

*#seg\_train\_folders = '/content/gdrive/MyDrive/datasets/lensandgauss/Train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/gdrive/MyDrive/datasets/lensandgauss/Test/'*

*#seg\_pred\_folders = '/content/gdrive/MyDrive/val/'*

*#pull from datasets*

*# lensandtint*

*#seg\_train\_folders = '/content/gdrive/MyDrive/datasets/lensandtint/Train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/gdrive/MyDrive/datasets/lensandtint/Test/'*

*#seg\_pred\_folders = '/content/gdrive/MyDrive/val/'*

*#pull from datasets*

*# tintandgauss*

*#seg\_train\_folders = '/content/gdrive/MyDrive/datasets/tintandgauss/Train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/gdrive/MyDrive/datasets/tintandgauss/Test/'*

*#seg\_pred\_folders = '/content/gdrive/MyDrive/val/'*

*#pull from datasets*

*#all*

*#seg\_train\_folders = '/content/gdrive/MyDrive/datasets/All/Train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/gdrive/MyDrive/datasets/All/Test/'*

*#seg\_pred\_folders = '/content/gdrive/MyDrive/val/'*

*#root\_path = '/content/gdrive/MyDrive/Colab Notebooks/skin\_images/'*

*#train\_pred\_test\_folders = os.listdir(root\_path)*

*#seg\_train\_folders = '/content/train/' #one more seg\_train folder within*

*#seg\_test\_folders = '/content/test/'*

*#seg\_pred\_folders = '/content/testval/'*

quantity\_tr **=** {}

quantity\_te **=** {}

**for** folder **in** os**.**listdir(seg\_train\_folders):

quantity\_tr[folder] **=** len(os**.**listdir(seg\_train\_folders**+**folder))

**for** folder **in** os**.**listdir(seg\_test\_folders):

quantity\_te[folder] **=** len(os**.**listdir(seg\_test\_folders**+**folder))

quantity\_train **=** pd**.**DataFrame(list(quantity\_tr**.**items()), index**=**range(0,len(quantity\_tr)), columns**=**['class','count'])

quantity\_test **=** pd**.**DataFrame(list(quantity\_te**.**items()), index**=**range(0,len(quantity\_te)), columns**=**['class','count'])

figure, ax **=** plt**.**subplots(1,2,figsize**=**(20,5))

sns**.**barplot(x**=**'class',y**=**'count',data**=**quantity\_train,ax**=**ax[0])

sns**.**barplot(x**=**'class',y**=**'count',data**=**quantity\_test,ax**=**ax[1])

print("Number of images in the train set : ", sum(quantity\_tr**.**values()))

print("Number of images in the test set ; ",sum(quantity\_te**.**values()))

plt**.**show()

**def** save\_history(history, model\_name):

*#convert the history.history dict to a pandas DataFrame:*

hist\_df **=** pd**.**DataFrame(history**.**history)

*# save to json:*

hist\_json\_file **=** model\_name**+**'\_history.json'

**with** open(hist\_json\_file, mode**=**'w') **as** f:

hist\_df**.**to\_json(f)

*# or save to csv:*

hist\_csv\_file **=** model\_name**+**'\_history.csv'

**with** open(hist\_csv\_file, mode**=**'w') **as** f:

hist\_df**.**to\_csv(f)

**def** plot\_accuracy\_from\_history(history, isinception**=False**):

color **=** sns**.**color\_palette()

**if**(isinception **==** **False**):

acc **=** history**.**history['acc']

val\_acc **=** history**.**history['val\_acc']

**else**:

acc **=** history**.**history['accuracy']

val\_acc **=** history**.**history['val\_accuracy']

epochs **=** range(len(acc))

sns**.**lineplot(epochs, acc, label**=**'Training Accuracy')

sns**.**lineplot(epochs, val\_acc,label**=**'Validation Accuracy')

plt**.**title('Training and Validation Accuracy')

plt**.**legend()

plt**.**figure()

plt**.**show()

**def** plot\_loss\_from\_history(history):

color **=** sns**.**color\_palette()

loss **=** history**.**history['loss']

val\_loss **=** history**.**history['val\_loss']

epochs **=** range(len(loss))

sns**.**lineplot(epochs, loss,label**=**'Training Loss')

sns**.**lineplot(epochs, val\_loss, label**=**'Validation Loss')

plt**.**title('Training and Validation Loss')

plt**.**legend()

plt**.**figure()

plt**.**show()

**def** do\_history\_stuff(history, history\_file\_name, isinception**=False**):

save\_history(history, history\_file\_name)

plot\_accuracy\_from\_history(history, isinception)

plot\_loss\_from\_history(history)

train\_datagen **=** ImageDataGenerator( rescale **=** 1.0**/**255.,shear\_range**=**0.2,zoom\_range**=**0.2)

*# we are rescaling by 1.0/255 to normalize the rgb values if they are in range 0-255 the values are too high for good model performance.*

train\_generator **=** train\_datagen**.**flow\_from\_directory(seg\_train\_folders,

batch\_size**=**32,

shuffle**=True**,

class\_mode**=**'categorical',

target\_size**=**(150, 150))

validation\_datagen **=** ImageDataGenerator(rescale **=** 1.0**/**255.) *#we are only normalising to make the prediction, the other parameters were used for agumentation and train weights*

validation\_generator **=** validation\_datagen**.**flow\_from\_directory(seg\_test\_folders, shuffle**=True**, batch\_size**=**1, class\_mode**=**'categorical', target\_size**=**(150, 150))

inv\_map\_classes **=** {v: k **for** k, v **in** validation\_generator**.**class\_indices**.**items()}

print(validation\_generator**.**class\_indices)

print(inv\_map\_classes)

**def** show\_few\_images(number\_of\_examples**=**2, predict\_using\_model**=None**):

figure1, ax1 **=** plt**.**subplots(number\_of\_examples,len(os**.**listdir(seg\_train\_folders)), figsize**=**(20,4**\***number\_of\_examples))

ax1 **=** ax1**.**reshape(**-**1)

axoff\_fun **=** np**.**vectorize(**lambda** ax:ax**.**axis('off'))

axoff\_fun(ax1)

axs **=** 0

**for** i, folder **in** enumerate(os**.**listdir(seg\_train\_folders)):

image\_ids **=** os**.**listdir(os**.**path**.**join(seg\_train\_folders,folder))

**for** j **in** [random**.**randrange(0, len(image\_ids)) **for** i **in** range(0,number\_of\_examples)]:

display **=** plt**.**imread(os**.**path**.**join(seg\_train\_folders,folder,image\_ids[j]))

plt**.**axis('off')

ax1[axs]**.**imshow(display)

title **=** 'True:'**+**folder

**if**(predict\_using\_model):

predicted\_classname **=** inv\_map\_classes[np**.**argmax(inception\_best\_model**.**predict(np**.**array([display])))]

title **=** title**+**'\nPredict :'**+**predicted\_classname

ax1[axs]**.**set\_title(title)

axs**=**axs**+**1

show\_few\_images(2)

**from** keras **import** applications

tf**.**keras**.**backend**.**clear\_session()

*# epoch config*

benchmark\_epoch **=** 20

vgg\_epoch **=** 20

resnet\_epoch **=** 20

inception\_epoch **=** 20

img\_width, img\_height **=** 150, 150

nb\_train\_samples **=** 6000

nb\_validation\_samples **=** 770

epochs **=** 20

batch\_size **=** 32

n\_classes **=** 4

**from** keras.applications **import** densenet

**def** build\_model():

base\_model **=** densenet**.**DenseNet121(input\_shape**=**(img\_width, img\_height, 3),

weights**=**"imagenet",

include\_top**=False**,

pooling**=**'avg')

**for** layer **in** base\_model**.**layers:

layer**.**trainable **=** **True**

x **=** base\_model**.**output

x **=** Dense(1000, kernel\_regularizer**=**regularizers**.**l1\_l2(0.01), activity\_regularizer**=**regularizers**.**l2(0.01))(x)

x **=** Activation('relu')(x)

x **=** Dense(500, kernel\_regularizer**=**regularizers**.**l1\_l2(0.01), activity\_regularizer**=**regularizers**.**l2(0.01))(x)

x **=** Activation('relu')(x)

predictions **=** Dense(n\_classes, activation**=**'softmax')(x)

model **=** Model(inputs**=**base\_model**.**input, outputs**=**predictions)

**return** model

densenet\_model **=** build\_model()

densenet\_model**.**compile(loss**=**'categorical\_crossentropy', optimizer**=**'adam', metrics**=**['acc'])

*# from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint, Callback*

*# early\_stop = EarlyStopping(monitor='val\_loss', patience=8, verbose=1, min\_delta=1e-4)*

*# reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.1, patience=4, verbose=1, min\_delta=1e-4)*

*# callbacks\_list = [early\_stop, reduce\_lr]*

filepath **=** "densenet\_-model-{epoch:02d}-{val\_acc:.2f}.hdf5"

reduce\_lr **=** ReduceLROnPlateau(monitor**=**'val\_loss', factor**=**0.05, patience**=**20, min\_lr**=**0.000002)

checkpoint **=** tf**.**keras**.**callbacks**.**ModelCheckpoint(filepath, monitor**=**'val\_acc', verbose**=**1, save\_best\_only**=True**, mode**=**'max')

early\_stopping **=** tf**.**keras**.**callbacks**.**EarlyStopping(monitor**=**'loss', patience**=**20)

history **=** densenet\_model**.**fit(train\_generator,epochs**=**benchmark\_epoch, verbose**=**1, validation\_data **=** validation\_generator,callbacks**=**[reduce\_lr,early\_stopping,checkpoint])

densenet\_model**.**save(filepath)

do\_history\_stuff(history, 'densenet\_model')

*#CNN architecture*

*#random architecture*

benchmark\_model **=** Sequential()

*# Input here is 4D array (batchsize, height, width, channels) - we have already created the train\_generator with batch size 32*

*# 32 Images of size each 150x150 with 3 color channels will be input into this layer*

benchmark\_model**.**add(Conv2D(128, kernel\_size**=**7, activation**=**'relu', input\_shape**=**(150,150,3)))

benchmark\_model**.**add(MaxPooling2D(pool\_size**=**(4,4), strides**=**(2,2)))

benchmark\_model**.**add(Conv2D(64, kernel\_size**=**5, activation**=**'relu'))

benchmark\_model**.**add(MaxPooling2D(pool\_size**=**(4,4), strides**=**(2,2)))

benchmark\_model**.**add(Flatten())

benchmark\_model**.**add(Dense(128,activation**=**'relu'))

benchmark\_model**.**add(Dense(4,activation**=**'softmax'))

benchmark\_model**.**compile(optimizer**=**'adam', loss**=**'categorical\_crossentropy', metrics**=**['acc'])

benchmark\_model**.**summary()

filepath **=** "bench\_mark\_-model-{epoch:02d}-{val\_acc:.2f}.hdf5"

reduce\_lr **=** ReduceLROnPlateau(monitor**=**'val\_loss', factor**=**0.05, patience**=**20, min\_lr**=**0.000002)

checkpoint **=** tf**.**keras**.**callbacks**.**ModelCheckpoint(filepath, monitor**=**'val\_acc', verbose**=**1, save\_best\_only**=True**, mode**=**'max')

early\_stopping **=** tf**.**keras**.**callbacks**.**EarlyStopping(monitor**=**'loss', patience**=**20)

history **=** benchmark\_model**.**fit(train\_generator,epochs**=**benchmark\_epoch, verbose**=**1, validation\_data **=** validation\_generator,callbacks**=**[reduce\_lr,early\_stopping,checkpoint])

benchmark\_model**.**save(filepath)

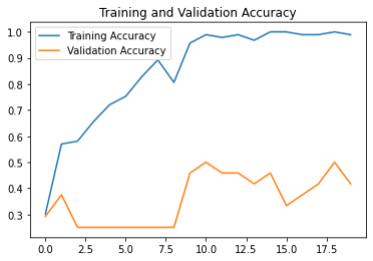
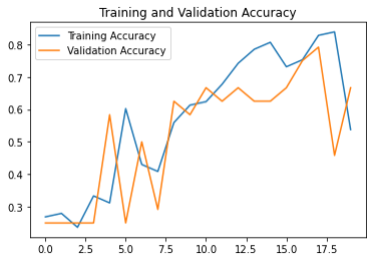
do\_history\_stuff(history, 'benchmark\_model')

**IX OBSERVATION / GRAPHS**

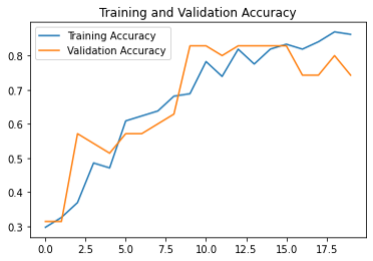
This Graphical representation of accuracies of DenseNet and CNN of different subcategories of blurred images are shown below. The first image of each category on the left side is of DenseNet and the second image of each category on the right is of CNN:

**DenseNet CNN**

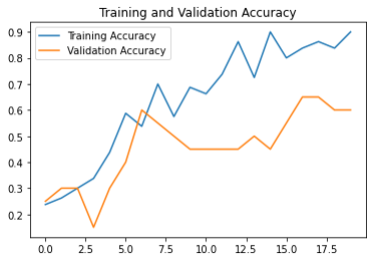
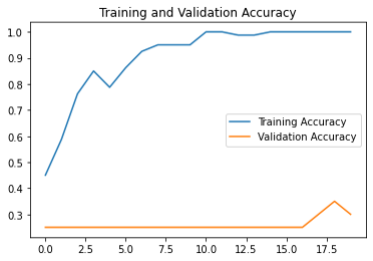
**Gaussian**

** **

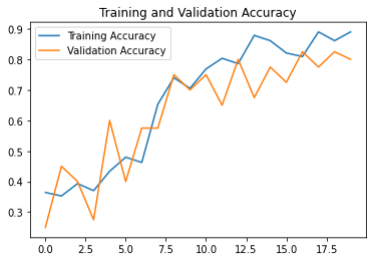
**Tinted**



**Lens Flare**



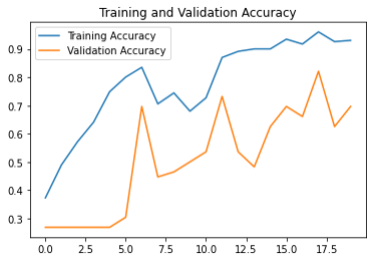
**Lens Flare and Gaussian Blur:**



**Lens Flare and Tinted Images:**



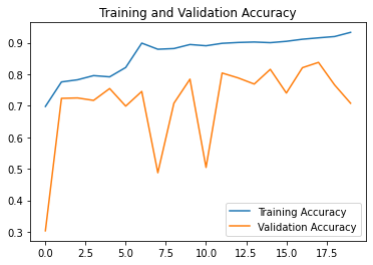
**Tinted and Gaussian Blur**



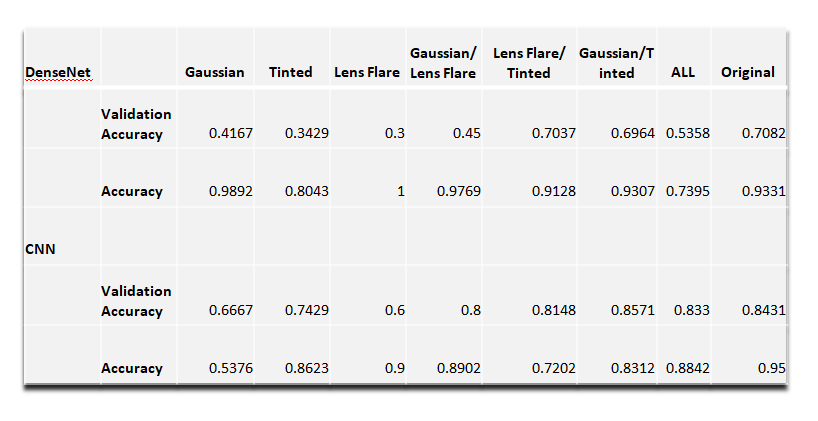
**All of the changed datasets (Gaussian Blur, Tinted images, and Lens Flare):**



**Original Dataset**



**X RESULT**



**Table 2**

**XI CONCLUSION**

In this paper, we used the DenseNet-121 model to create a convolutional neural network that can differentiate cases of COVID-19, pneumonia and tuberculosis from chest X-rays. It is more resilient and dependable than most other published models since it uses a larger dataset and more control classes. It can tell the difference between COVID-19 and healthy X-rays, viral pneumonia, and tuberculosis X-rays.

This research offered an alternative COVID-19 screening method based on identifying specific features in chest X-Ray pictures. If successful, it will provide medical practitioners with assistive intelligence to help them deal with the severity of the pandemic. The preliminary findings are promising, with the potential for replication on larger and more diversified data sets.

**XII REFERENCES**

1. World Health Organization, “Coronavirus Disease (COVID-19),” *World Health Organization*, 2021.

2. Centers for Disease Control and Prevention, “Basic TB facts,” *Centers for Disease Control and Prevention*, Mar. 20, 2016. https://www.cdc.gov/tb/topic/basics/default.htm.

3. Kaggle Chest X-Ray Pneumonia, Covid-19, Tuberculosis

4. Huang G., Liu Z., Van Der Maaten L. & Weinberger K. Densely connected convolutional networks. Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition. pp. 4700-4708 (2017)

5. Cohen, Joseph Paul, Lan Dao, Karsten Roth, Paul Morrison, Yoshua Bengio, Almas F. Abbasi, Beiyi Shen et al. "Predicting covid-19 pneumonia severity on chest x-ray with deep learning." *Cureus* 12, no. 7 (2020).

6. A. Signoroni *et al.*, “BS-Net: Learning COVID-19 pneumonia severity on a large chest X-ray dataset,” *Medical Image Analysis*, vol. 71, p. 102046, Jul. 2021, doi: 10.1016/j.media.2021.102046.

7. M. Rahimzadeh and A. Attar, “A modified deep convolutional neural network for detecting COVID-19 and pneumonia from chest X-ray images based on the concatenation of Xception and ResNet50V2,” *Informatics in Medicine Unlocked*, vol. 19, p. 100360, 2020, doi: 10.1016/j.imu.2020.100360.

8. M. Nishio, S. Noguchi, H. Matsuo, and T. Murakami, “Automatic classification between COVID-19 pneumonia, non-COVID-19 pneumonia, and the healthy on chest X-ray image: combination of data augmentation methods,” *Scientific Reports*, vol. 10, no. 1, Oct. 2020, doi: 10.1038/s41598-020-74539-2.

9. J. P. Cohen *et al.*, “Predicting COVID-19 Pneumonia Severity on Chest X-ray With Deep Learning,” *Cureus*, Jul. 2020, doi: 10.7759/cureus.9448.

10. T. Ozturk, M. Talo, E. A. Yildirim, U. B. Baloglu, O. Yildirim, and U. Rajendra Acharya, “Automated detection of COVID-19 cases using deep neural networks with X-ray images,” *Computers in Biology and Medicine*, Apr. 2020, doi: 10.1016/j.compbiomed.2020.103792.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |