Detecting pediatric pneumonia from X-rays using CNN

By Sameeha Ramadhan

THE OUTLINE

01 **INTRODUCTION**
The Project's Goal

02 **WHAT IS PNEUMONIA?**
A Brief Description

03 **THE DATA**
Obtaining & Preprocessing

04 **THE MODEL**
CNN & Evaluation

05 **CONCLUSION**
Summary & Recommendations

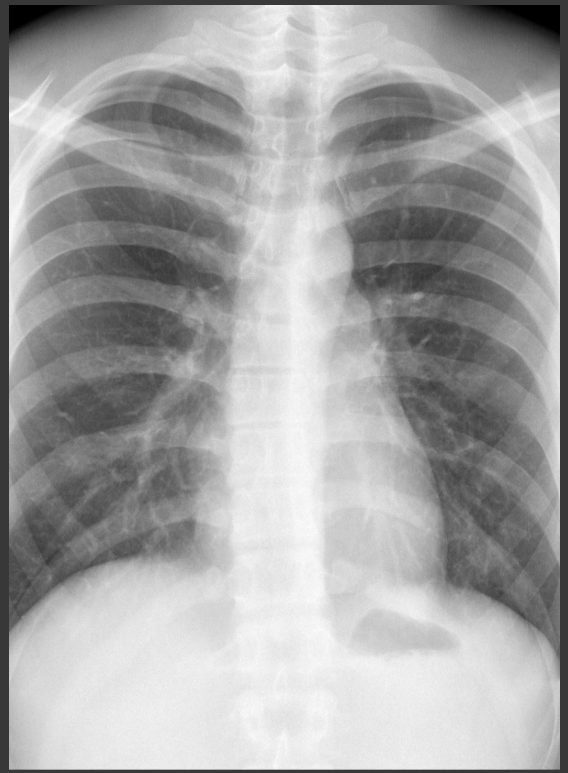
01

Introduction

The Project's Goal

INTRODUCTION

The goal of this project is to build an algorithm that can read X-Ray images and determine if pneumonia is present or not. I've chosen to use Convolutional Neural Network, or CNN.



02

WHAT IS PNEUMONIA?

PNEUMONIA

WHAT IT IS

Pneumonia is an infection that affects one or both lungs.

WHAT IT DOES

It causes the air sacs, or alveoli, of the lungs to fill up with fluid or pus.

SYMPTOMS

Range from mild to serious and may include a cough with or without mucus fever, chills, and trouble breathing

CAUSES

Bacteria, viruses, or fungi may cause pneumonia.

03

The Data

GATHERING AND PREPROCESSING

SOURCE

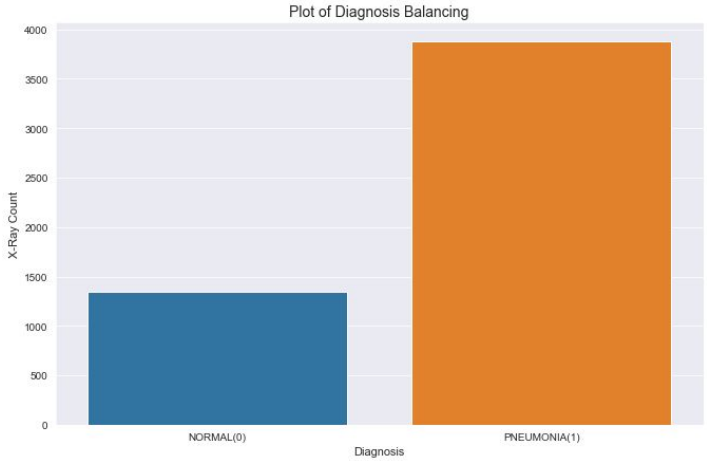
kaggle.com

CONTAINS

5756 X-Ray images

STRUCTURE

3 subsets: ‘train’, ‘test’, and ‘val’



UNBALANCED

The distribution wasn’t balanced but left untouched

256 COLORS

All images were rescaled to this size

0.3 ZOOM RANGE

All augmented images were zoomed and vertically flipped.

After viewing the subsets, ‘val’ was dropped for its extremely small size (only 0.3% of the data).

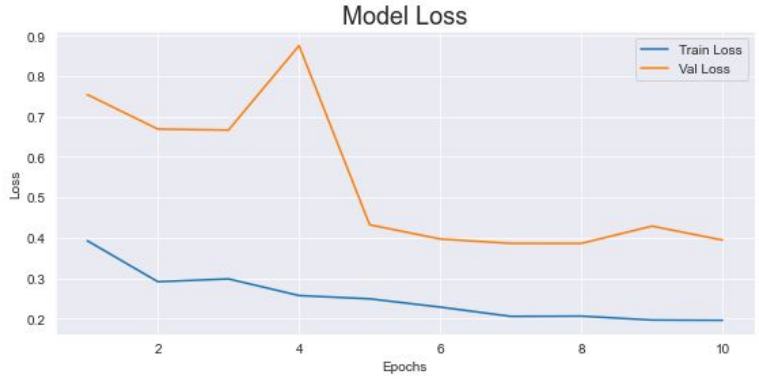
04

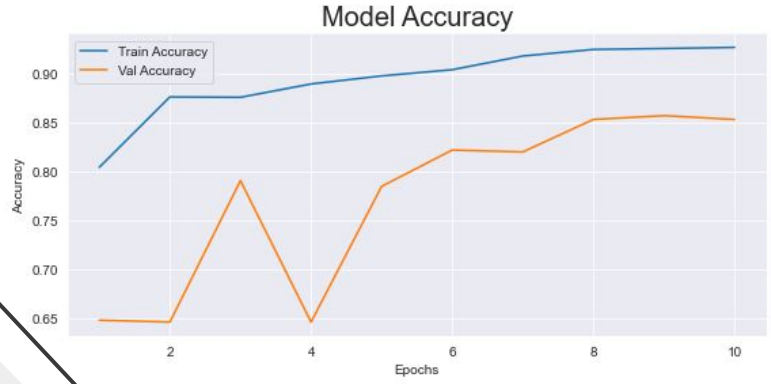
The Model

Convolutional Neural Network (CNN)

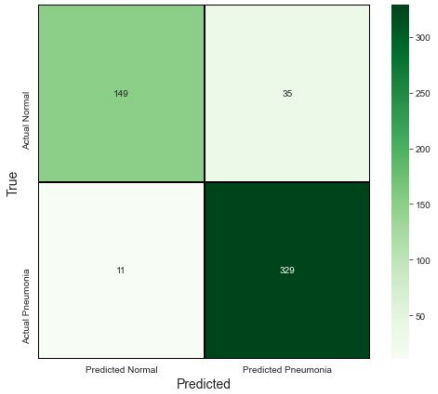
- 8 different models were tested
- The final model has 5 convolutional blocks
- Each was tested with 10 epochs and if accuracy was more than 70%, epochs increased up to 50

```
inputs = Input(shape=(img_dims, img_dims, 3))

#1st Convolution
x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(inputs)
x = Conv2D(filters=16, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = MaxPool2D(pool_size=(2, 2))(x)

#2nd Convolution
x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=32, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)

#3rd Convolution
x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)

#4th Convolution
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=128, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)
x = Dropout(rate=0.2)(x)

#5th Convolution
x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = SeparableConv2D(filters=256, kernel_size=(3, 3), activation='relu', padding='same')(x)
x = BatchNormalization()(x)
x = MaxPool2D(pool_size=(2, 2))(x)
x = Dropout(rate=0.2)(x)

# Fully Connected Layer
x = Flatten()(x)
x = Dense(units=512, activation='relu')(x)
x = Dropout(rate=0.7)(x)
x = Dense(units=128, activation='relu')(x)
x = Dropout(rate=0.5)(x)
x = Dense(units=64, activation='relu')(x)
x = Dropout(rate=0.3)(x)

# Output layer
output = Dense(units=1, activation='sigmoid')(x) #<--Converting a real value into a interpretable probability

# Creating model and compiling
model = Model(inputs=inputs, outputs=output)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Callbacks
checkpoint = ModelCheckpoint(filepath='best_weights.hdf5', save_best_only=True, save_weights_only=True)
#hdf5 (Hierarchical Data Format) is to store the weights of the best performing epoch
lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2, verbose=2, mode='max') #<--Reduce learning rate
#when the metric has stopped improving.
early_stop = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=1, mode='min')
```





MODEL RESULTS



For unbalanced data, Recall is the better accuracy metric.
Our model will only misdiagnose 11 out of 340 patients.

```
TEST METRICS ----------------------
Accuracy: 91.22137404580153%
Precision: 90.38461538461539%
Recall: 96.76470588235294%
F1-score: 93.4659090909091

TRAIN METRIC ----------------------
Train Accuracy: 95.13
```

*It's clear from the train accuracy that the model is overfitted

05 Conclusion

This project has shown how to classify positive and negative pneumonia diagnosis' from a set of X-Ray images displayed deep learning being used in real world problems.

Recommendations

- Source more data with more images to allow for a more balanced distribution.
- Attempt to run the models with a greater number of epochs (such as 100 or more) if necessary to determine if there is convergence. I've unsuccessfully attempted this.
- Fine tune and test other parameters to reduce overfitting.
- With this project as a base, our work can be built upon to detect more complex X-Rays, such as cancers, broken bones and more.

THANKS

shramadhan@gmail.com
GitHub: @samtuleen
linkedin.com/in/sameeha-ramadhan-3a1bba140