A Customized CNN based Approach for Pneumonia Detection in Chest X-ray Images

Patel Pranav Girishbhai

11701217 [RKE063B003]

p.patelpranav245@gmail.com

+91 9428504781

Abstract

Pneumonia causes the death of around 700,000 children every year and affects 7% of the global population. Chest X-rays are primarily used for the diagnosis of this disease. However, even for a trained radiologist, it is a challenging task to examine chest X-rays. In this work, an efficient model for the detection of pneumonia trained on digital chest X-ray images is proposed, which could aid the radiologists in their decision making process. A novel approach based on a weighted classifier is introduced, which combines the weighted predictions from the state-of-the-art deep learning models such as Conv2d, MaxPooling2D, Flatten, Dense, and AlexNet in an optimal way. This approach is a supervised learning approach in which the network predicts the result based on the quality of the dataset used. Transfer learning is used to fine-tune the deep learning models to obtain higher training and validation accuracy. After training a model the model accuracy, I get is around 90% with 96% recall which is quite good considering the size of the data that is used.

Keywords: deep learning, transfer learning, sequential model, image processing

1. Introduction

Effectively classifying medical images play an essential role in aiding clinical care and treatment. For example, Analysis Z-ray is the best approach to diagnose pneumonia which causes about 50,000 people to die per year in the US, but classifying pneumonia for chest X-ray needs professional radiologist which is a rare and expensive resource for some regions.

The use of traditional machine learning methods, such as SVM in medical image classification, began long ago. However, these methods have the following disadvantages: the performance is far from the practical standard, and the developing of them is quite slow in recent years. Also, the feature extracting and selection are time-consuming and vary according to different objects. The deep neural networks (DNN), especially the convolutional neural networks (CNNs), are widely used in changing image classification tasks and have achieved significant performance since 2012. Some research on medical image classification by CNN has achieved performances rivalling human experts. Technologies used for this project are keras, scilkit-learn , numpy, pandas, matplotlib, opencv, and keras layers like Conv2d, Dense, Flatten , optimizer like Adam, and tensorflow. Keras is an open source library that provides a python interface for artificial neural network. Keras acts as an interface for the Tensorflow library. Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Moreover Numpy forms the foundation of

the Machine Learning stack. TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical imager analysis, natural language processing, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Pretrained Model are also used for model building. What is pre-trained model. A pre-trained model is a model that has been previously trained on a dataset and contains the weights and biases that represent the features of whichever dataset it was trained on. Learned features are often transferable to different data. For example, a model trained on a large dataset of bird images will contain learned features like edges or horizontal lines that you would be transferable your dataset.

2. Methods

2.1 Outline of Methodology

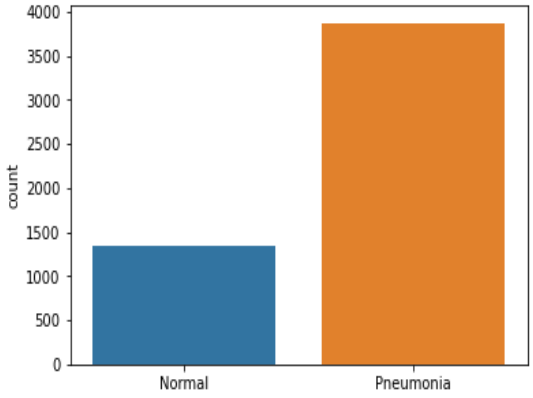
2.1.1 Dataset Used

Data used for the project is been used from the Kaggle. Data has three directory in it i.e. test, train, val (validate). Total images in it are 5216, 624, and 16 respectively. In that data is divided into two class namely NORMAL, PNUMONIA.

Link of dataset used: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

2.1.2 Pre-processing

In pre-processing after importing the dataset I declare the variable with three different names i.e. train_folder, test_folder, val_folder where I store the all images in it of respective folder. After that I plot the count plot of Normal and Pneumnia.



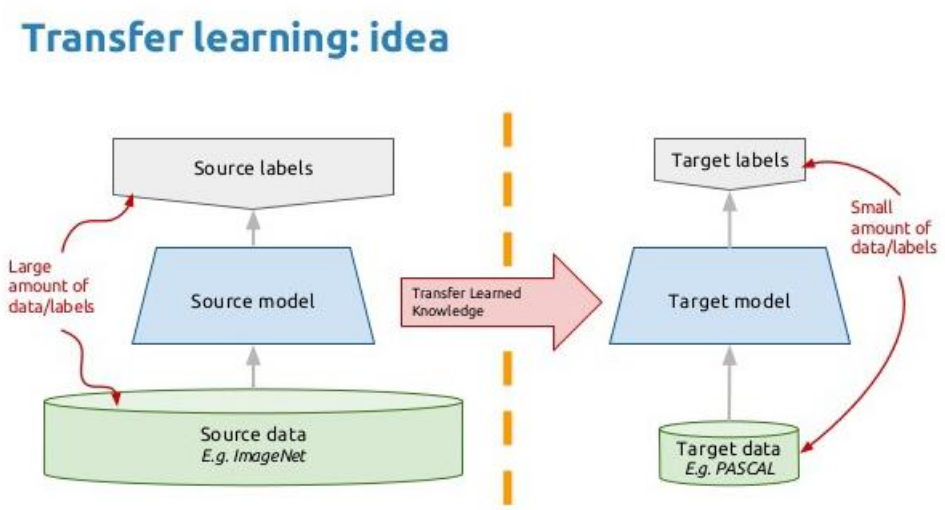
2.1.3 Model Building

The methods used for the model are

2.1.3.1 CNN (Convultion Neural Network).

CNN stands for Convolutional Neural Network which is a specialized neural network for processing data that has an input shape like a 2D matrix like images. CNN's are typically used for image detection and classification.

2.1.3.2 Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.



2.1.3.3 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU's.

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

2.1.4 Functions and Methodology Used

2.1.4.1 Conv2d

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

2.1.4.2 Activation

An activation function is a function that is added into an artificial neural network in order to help the **network learn complex patterns in the data**. When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding **what is to be fired to the next neuron**. That is exactly what an activation function does in an ANN as well. **It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell**.

2.1.4.3 MaxPooling

Pooling is a feature commonly imbibed into Convolutional Neural Network (CNN) architectures. The main idea behind a pooling layer is to "accumulate" features from maps generated by convolving a filter over an image. Formally, its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network. The most common form of pooling is max pooling.

Max pooling is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation. The other forms of pooling are average, general.

2.1.4.4 Dropout

Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

As a neural network learns, neuron weights settle into their context within the network. Weights of neurons are tuned for specific features providing some specialization. Neighboring neurons become to rely on this specialization, which if taken too far can result in a fragile model too specialized to the training data. This reliant on context for a neuron during training is referred to complex co-adaptations.

You can imagine that if neurons are randomly dropped out of the network during training, that other neurons will have to step in and handle the representation required to make predictions for the missing neurons. This is believed to result in multiple independent internal representations being learned by the network.

The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.

2.1.4.5 Flatten

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a ***fully*-connected** layer. In other words, we put all the pixel data in one line and make connections with the final layer.

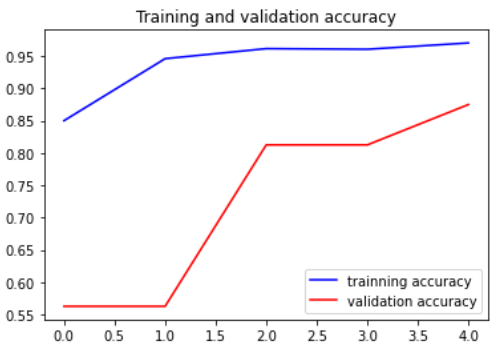
2.1.4.6 Dense

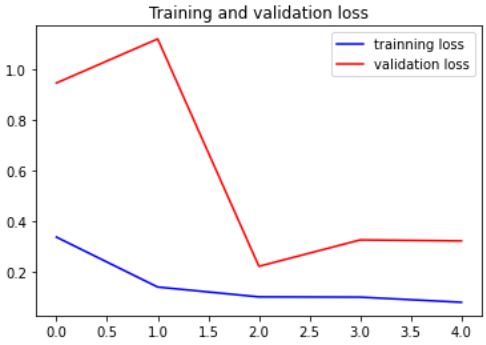
Dense layer is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.

3. Result and Discussion

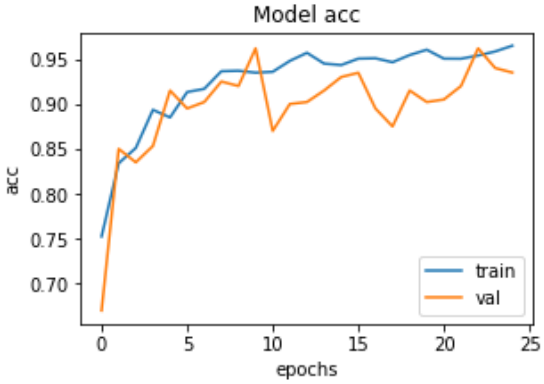
The primary goal of my CNN and pre-trained based model is to correctly diagnose pneumonia among normal chest x-ray images. For this I prepare model using above technologies and for this I use the Kaggle inbuilt server ,CPU and GPU for the processing and building a model.
First CNN model has given the accuracy of 81.25% at loss of 65.8 % .
VGG16 model has given the accuracy of 93.5 % at a loss of 20%.

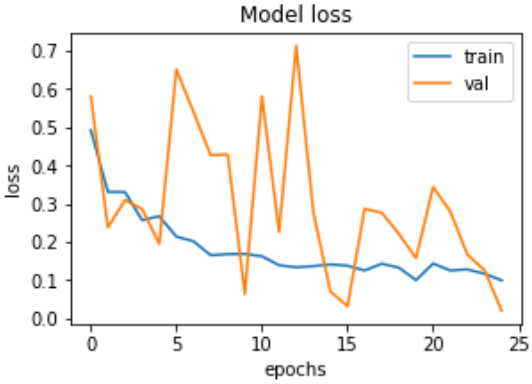
Result of CNN model:





Result of VGG-16 model:





4. Conclusion

In this article, my goal is to implement a deep learning-based approach to classify pneumonia from chest X-ray images using transfer learning , CNN and some pre-trained model. In this project, I tried the transfer learning approach and used the pretrained architectures, AlexNet, CGG-16 trained on the ImageNet dataset, to extract features able to build model with accuracy around 85% which I think is quite as the dataset used for the model building is quite large.

5. Refrences

https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/discussion
https://neurohive.io/en/popular-networks/vgg16/

https://keras.io/api/layers/
https://towardsdatascience.com/deep-learning-for-detecting-pneumonia-from-x-ray-images-fc9a3d9fdba8
https://towardsdatascience.com/chest-x-rays-pneumonia-detection-using-convolutional-neural-network-63d6ec2d1dee

GITHUB LINK:

https://github.com/pranav245/-Pneumonia-Detection-in-Chest-X-ray-Images