Classifying Chest X-ray Images with Deep Learning By Using Sagemaker

Jingde Guo

March 20, 2020

General Background

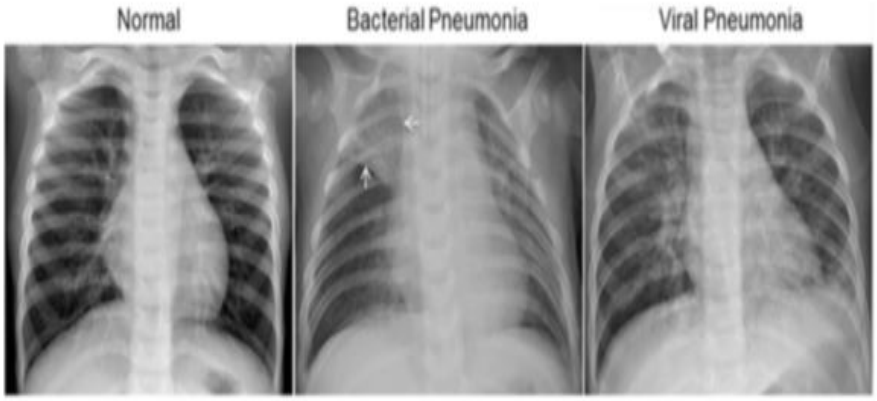
Deep learning is one of the family members of the machine learning based on artificial neural networks. With the Convolutional Neural Network(CNN), images can be classified well.

In medical image science, deep learning can be used to classify the X-ray images. In this project Chest X-rays are used to classify whether Pneumonia exists. With the model that can classify whether the patient has Pneumonia, doctors can reduce time to read X-ray images. The model will give the possibility of the existence of Pneumonia, which can be used as a auxiliary diagnosis.

The reason I choose this topic is due to the outbreak of the coronavirus, such classification model can be used to speed up the diagnostic process. Furthermore, it is also a good challenge for me to put everything I learned in practice.

Datasets and inputs

The dataset is retrieved from Kaggle:
https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.



The sample of the images as above. The size of images varies around 300kb.

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (0/1). 0 represents Normal and 1 represents Pneumonia. There are 5,902 X-Ray images (JPEG). Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou. All chest X-ray imaging was performed as part of patients' routine clinical care. For the analysis of chest x-ray images, all chest radio graphs were initially screened for quality control by removing all low quality or unreadable scans. The diagnoses for the images were then graded by two expert physicians before being cleared for training the AI system. In order to account for any grading errors, the evaluation set was also checked by a third expert(Mooney, 2018).

Problem statement

The problem is to create a model that can classify the chest x-ray image in terms of whether Pneumonia exists. A deep learning method will be used, and Amazon SageMaker will be utilized. A high level method for SageMaker training will be deployed in building the model.

Benchmark model

The benchmark model can be found in Kaggle. The best model in Kaggle Kernels right now has recall ratio 0.98 and precision ratio 0.79. The model he used is based on partial transfer learning and rest of the model will be trained from scratch. My goal is to get the similar result based on the same test dataset.

The link is: https://www.kaggle.com/aakashnain/beating-everything-with-depthwise-convolutionModel

Evaluation metrics

Accuracy ratio can be used as an evaluation matrix, since the higher accuracy ratio the better the model can classify the images. Despite the existence of the imbalance in the dataset, accuracy ratio will not be affected due to it accounts for both false negative and false positive.
The formulas can be written as:

$$\text{Accuracy } = \frac{tp+tn}{tp+tn+fp+fn}$$

where tp denotes true positive, fp denotes false positive, tn denotes true negative, and fn denotes false negative.

Analyze the Problem

The dataset consists three files namely train, valid, and test. The number of images in each file are 5232, 46, and 624 respectively. First we should label all file names with 0 or 1 indicating weather the person has Pneumonia. The labeled dataframe is shown as following:

Training images:

	Pic	pneumonia
0	chest_xray/train/1/person1079_bacteria_3019.jpeg	1.0
1	chest_xray/train/1/person586_bacteria_2420.jpeg	1.0
2	chest_xray/train/0/NORMAL2-IM-1008-0001.jpeg	0.0
3	chest_xray/train/1/person1604_virus_2782.jpeg	1.0
4	chest_xray/train/1/person28_bacteria_143.jpeg	1.0

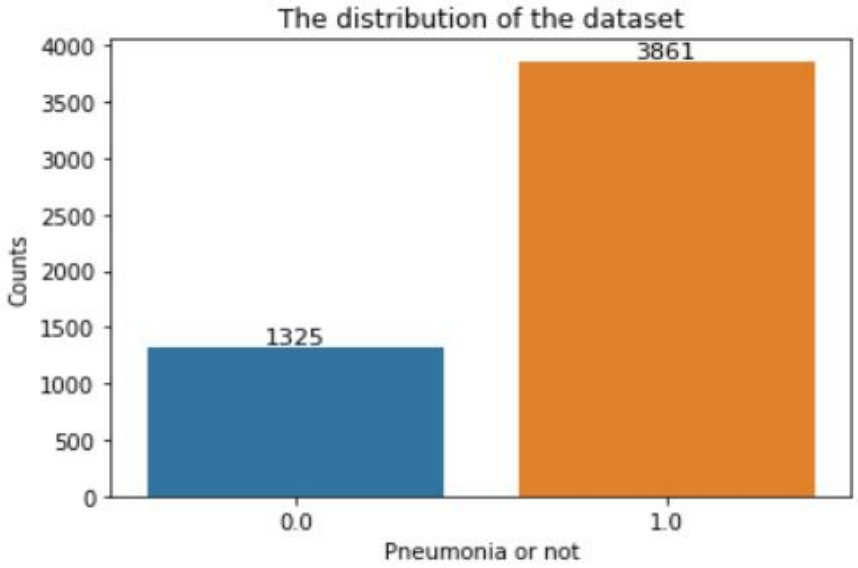
Validation images:

	Pic	pneumonia
0	chest_xray/valid/0/IM-0129-0001.jpeg	0.0
1	chest_xray/valid/0/IM-0156-0001.jpeg	0.0
2	chest_xray/valid/0/IM-0127-0001.jpeg	0.0
3	chest_xray/valid/1/person124_virus_238.jpeg	1.0
4	chest_xray/valid/1/person122_virus_229.jpeg	1.0

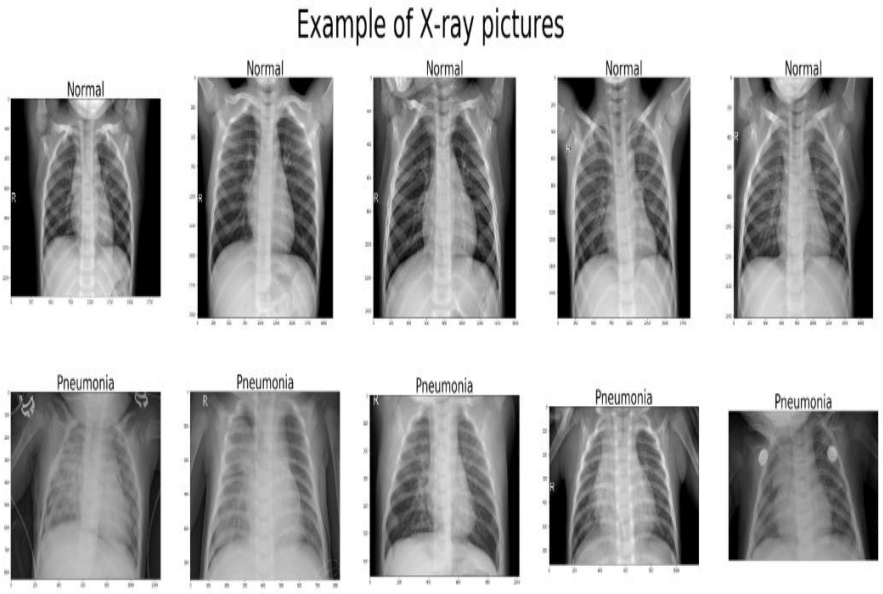
Test images:

	Pic	pneumonia
0	chest_xray/test/1/person103_bacteria_489.jpeg	1.0
1	chest_xray/test/0/NORMAL2-IM-0030-0001.jpeg	0.0
2	chest_xray/test/1/person83_bacteria_412.jpeg	1.0
3	chest_xray/test/1/person57_virus_113.jpeg	1.0
4	chest_xray/test/0/IM-0017-0001.jpeg	0.0

The distribution of the training dataset is:



Furthermore, the example images of the dataset with labels are shown as following:



Since this is a binary classification problem, the built-in image classification algorithm can be used. The Amazon SageMaker image classification algorithm is

a supervised learning algorithm. It uses a convolutional neural network (ResNet) that can be trained from scratch or trained using transfer learning when a large number of training images are not available.

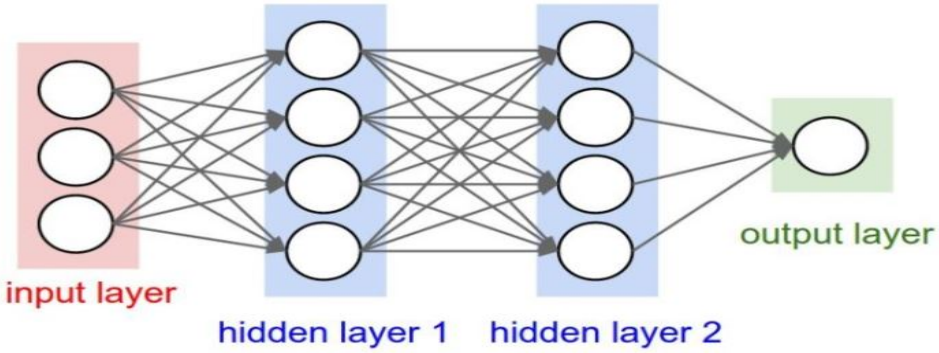
The convolutional neural network works as the following picture. The computer deal with the picture different from what we see.

What We See

What Computers See

When a computer sees an image (input an image), it looks at a large number of pixel values. Depending on the resolution and size of the picture, it will see an array of 32 x 32 x 3 (3 refers to the RGB values). To make this clear, suppose we have a 480 x 480 color image in JPG format, and its corresponding array has 480 x 480 x 3 elements. Each of these numbers has a value from 0 to 255, which describes the gray scale of the pixel corresponding to that point. When we humans classify images, these numbers are useless, but they are the only input available to a computer. The idea is that when you provide this array to a computer, it will output numbers describing the probability that the image belongs to a particular category (for example: 80% are cats, 15% are dogs, and 5% are birds).

The convolutional neural network deal with these pictures as following:



Input layer: Many neurons receive a large amount of non-linear input information. The input message is called the input vector.

Output layer: The information is transmitted, analyzed, and weighed in the neuron link to form the output result. The output message is called the output vector.

Hidden layer: is a layer composed of many neurons and links between the input layer and the output layer. If there are multiple hidden layers, it means multiple activation functions.

Commonly used non-linear activation functions are sigmoid, tanh , relu, etc. The first two sigmoid / tanh are more common in the fully connected layer, and the latter relu is common in the convolution layer.

The function expression of sigmoid is as follows:

$$g(z) = \frac{1}{1+e^{-z}}$$

Central to the convolutional neural network is the convolutional layer that gives the network its name. This layer performs an operation called a "convolution". Due to their structural characteristics of local area connection, weight sharing, and down-sampling, convolutional neural networks excel in image processing. The particularity of convolutional neural networks compared to other neural networks is mainly in the aspects of weight sharing and local connection. Weight

sharing makes the network structure of convolutional neural networks more similar to biological neural networks. Local connections are not like traditional neural networks, where each neuron in layer n-1 is connected to all neurons in layer n, but between neurons in layer n-1 and some neurons in layer n connection. The effect of these two features is to reduce the complexity of the network model and the number of weights.

Convolutional neural networks have long been one of the core algorithms in the field of image recognition, and have stable performance when the learning data is sufficient. For general large-scale image classification problems, convolutional neural networks can be used to construct hierarchical classifiers. They can also be used in fine-grained recognition to extract discriminative features of images for other purposes.

Anyway, with the convolutional neural network, the image classification problem can be tackled.

Implementation

First we should prepossess the data in order to fit the requirement of the algorithm. The idea is we first set up the role and the session of Sagemaker and then define the bucket name that we will use later. Then we create four folders namely 'train', 'train_lst', 'validation', and 'validation_lst'. Then we use shutil package to copy required images to the target files, and write our annotations as lst file and store it in '_lst' files.

Secondly we can upload our processed data to the S3. After the upload we can define the parameters of the model.

```
prefix = 'output'
output_path = 's3://{}/{}'.format(bucket, prefix)
model = sagemaker.estimator.Estimator(
    container,
    role=role,
    train_instance_count=1,
    train_instance_type='ml.p2.xlarge',
    train_max_run=36000,
    input_mode='File',
    output_path=output_path,
    sagemaker_session=sess
)
```

```
# Hyperparameters
model.set_hyperparameters(
    num_layers=18,
    use_pretrained_model=1,
    image_shape='3,224,224',
    num_classes=2,
    mini_batch_size=32,
    resize=224,
    epochs=10,
    learning_rate=0.001,
    num_training_samples=5186,
    augmentation_type='crop_color_transform'
)
```

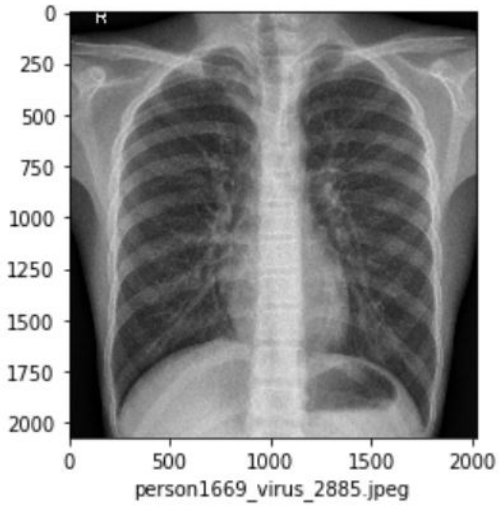
In our case, we have 5186 training samples and 2 output classes. The learning rate is set to be 0.001. The original images have different sizes, therefore, they are resized to 224 such that the input image dimension is '$3\times224\times224$'. The number of layers is set to be 18. Since we use Sagemaker built-in algorithm, we do not need to normalize it by ourselves, the algorithm will do it during the training process .

Then we can train our model with cloud computing. After the training job finished we can deploy our model as an endpoint on Sagemaker. The detail can be found in the Jupyter Notebook.

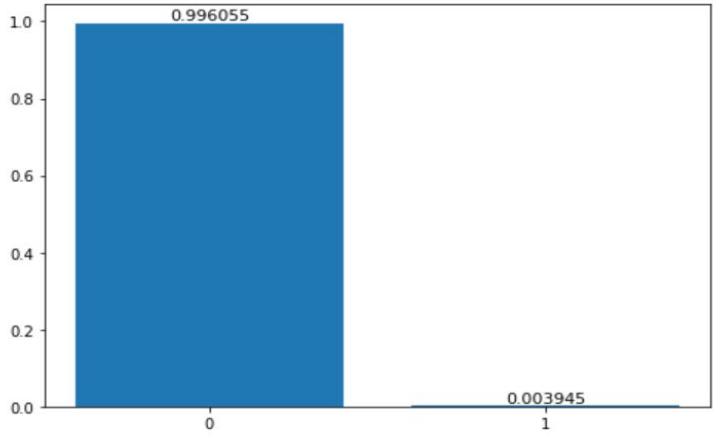
Results

The result of the training accuracy ratio is 98.61% and the validation accuracy ratio is one. The result is remarkable, but we should first try to predict some examples to see how the model works.

First we choose a picture of a normal one from test set.

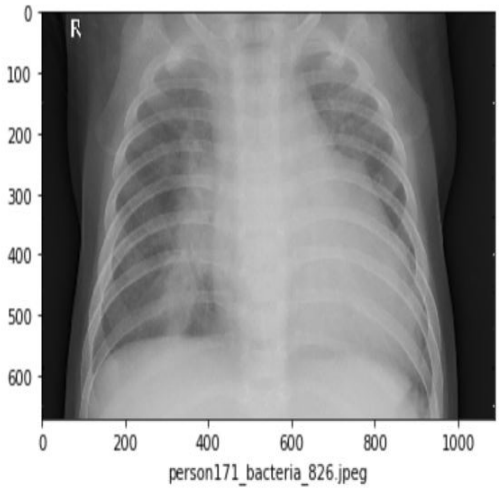


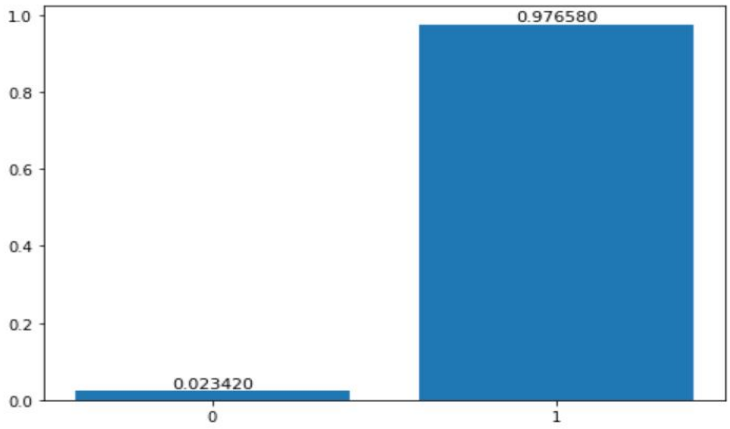
This is a normal lung, the model predicts the probability of each class is:



Since 0 indicates the normal lung, the model performs well.

Then we choose an image from a pneumonia case.





We can see the model still correctly classify the image.

Then we should use evaluation metrics to evaluate the model. The formulas can be written as:

$$\text{Precision} = \frac{tp}{tp+fp}$$

$$\text{Recall} = \frac{tp}{tp+fn}$$

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn}$$

	predict 0	predict 1
true 0	216.0	17.0
true 1	64.0	325.0

```
Accuracy:   0.870
Recall:     0.835
Precision:  0.950
```

In this task, recall ratio should be as higher as possible. Based on our model predictions, the recall ratio is nearly 83.5%, which is pretty good. However, if the model will be used in reality the recall ratio should be improved such that number of the miss diagnosis will decrease. Because miss diagnosis will do harm to the patients but false positive can be double checked by the doctor. Next we should improve the recall ratio based on the cutoff values, if the model predicts the probability of being 1 is greater than 0.1, the image will be classified as pneumonia.

When we set the cutoff value equals to 0.1, the model yields better results.

	predict 0	predict 1
true 0	182	51
true 1	12	377

```
Accuracy:   0.899
Recall:     0.969
Precision:  0.881
```

After we set the lower cutoff value, the model has higher accuracy ratio. The recall ratio improves a lot to nearly 97%. However we only lose a little bit precision ratio. The usual cutoff value is 0.5 but in this diagnostic problem we should choose relatively lower cutoff value in order to decrease the number of false negatives.

Conclusions

In this project, an image classification model is built to classify the chest x-ray images. At the end, we got remarkable results, with recall ratio 96.9%. The benchmark model which was performed on Kaggle has recall ratio 98% but sacrifice the precision ratio too much, which has only 79%. The overall accuracy is 82.7% but mine has nearly 90%.

On the other hand, the dataset might be too small with only nearly 6000 images. A dataset with more images is appreciated to elaborate the model. However, the methods and the thoughts of dealing with this problem is useful for further researches.

Reference

Mooney, P. (2018, March 24). Chest X-Ray Images (Pneumonia). Retrieved from https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Docs.aws.amazon.com. 2020. [online] Available at: <https://docs.aws.amazon.com/sagemaker/latest/dg/I
HowItWorks.html> [Accessed 19 March 2020].