Convolutional Neural Network for Lung Cancer Detection

Amada Echeverria, Mulin Tang, Muye Zhao
Doctor Truong Huy Nguyen
CISC 6930 L02 Data Mining (Fall 2017)
December 2017

1. Abstract

The goal of this application-based project is to improve detection of true positives with respect to cancerous lesions in the lungs so that the right patients can more promptly receive crucial treatment and have more time with their doctors. We aim to accomplish this by creating algorithms that classify lesions in the lungs as cancerous or not cancerous. We use CNN (Convolutional Neural Network), an image classification algorithm used for analyzing visual imagery, along with a data set of over one thousand high-resolution lung scans provided by the National Cancer Institute, which is also used as a part of the data of the Data Science Bowl 2017. We use log loss and accuracy as our evaluations. Due to the small data size and sample bias, our model has a good performance on log loss but not on accuracy. To improve accuracy, we can feed our model new data or consider other machine learning methods which may be more suitable for 3D image data.

2. Introduction

We chose this problem because one of the team members' grandmother suffered from lung cancer. A doctor in Mexico had previously misdiagnosed a lesion as non-cancerous, and, five years later, realized the diagnosis had been a false-negative. The grandmother was then told she actually did have lung cancer, and it had been developing for five years, a time during which it could possibly have been cured! We found that 20% of lung cancer deaths could be reduced with early detection, and, thus, we were motivated to make a difference.[1] Having a personal experience as a catalyst and motivational factor, our team set out to contribute to ending this disease.

Lung cancer is rampant and treatment is expensive, a problem which attracted us because we believe it is not one to accept, but one that can be changed. Our guiding belief was that high quality data, machine learning, and good intentions are more powerful than disease! Detecting lung cancer in its early stages is one of the most effective ways to increase the chances of recovery of those who develop it, and, as such, this data mining problem should be given a lot of importance. The goal of the project is to create algorithms—with high accuracy—that classify lesions in the lungs as cancerous or not. Creating better lung cancer detection algorithms is extremely important to improving and lengthening patients' lives. Additionally, from a more economic, as opposed to humanist, perspective, the cost of health care is very high, so being able to detect sickness in advance, and prevent or improve it, can help reduce costs, an added benefit.

We decided to use high-quality scans of lungs, provided by the Data Science Bowl on Kaggle to train our Convolutional Neural Network (CNN), minimize the rate of false positives (which is too high now), help doctors spend more time with the afflicted, and facilitate people getting healthcare earlier as opposed to later. It was important for us to ask good questions, detect patterns intelligently, and work on a project of social, economic, and humanitarian significance. Additionally, it is also significant as a data mining task.

[1] Aberle DR, Adams AM, Berg CD, et al. "Reduced lung-cancer mortality with low-dose computed tomographic screening." N Engl J Med. 2011;365:395-409.

3. Dataset

Our data comes from National Cancer Institute, which is also used for Data Science Bowl 2017 Stage 1.

3.1 Data Description

In this dataset, we are given over one thousand low-dose CT images from high-risk patients in Digital Imaging and Communications in Medicine (DICOM) format. Each image contains a series with multiple axial slices of the chest cavity. Each image has a variable number of 2D slices, which can vary based on the machine taking the scan and on the patient. The DICOM files have a header that contains the necessary information about the patient id, as well as scan parameters such as the slice thickness.

Each patient id has an associated directory of DICOM files. The patient id is found in the DICOM header and is identical to the patient name. The exact number of images will differ from case to case, varying according to the number of slices. Images were compressed as .7z files due to the large size of the dataset.

Below is the list of data files we have used in this project:

stage1.7z	contains all images for the first stage of the competition, including both the training and test set. This is file is also hosted on BitTorrent.
stage1_labels.csv	contains the cancer ground truth for the stage 1 training set images
sample_images.7z	a smaller subset set of the full dataset, provided for people who wish to preview the images before downloading the large file.
data_password.txt	contains the decryption key for the image files

3.2 Data File Structure

Before we tailored into our actual stage 1 dataset, we first took a closer look at the sample_images, which is a smaller set of data provided for people who wish to preview the attributes of the image and get a sense of how to manipulate the actual data. The sample_images.7z file is made up of several subdirectories, each linking with a single patient ID and containing about 100-300 DICOM files, except there is one patient that has over 400 DICOM files.

3.3 Data Cleansing

We then looked through the actual stage1 data with the same structure found in the sample data file. The stage 1 dataset is almost 225GB. We had to use an external hard drive to store the stage1 dataset and run preprocessing code on it. After a closer look, we found that the stage1 file has 1595 subdirectories, meaning 1595 patients. Even though we have scan data for each of these 1595 patients, we don't have the 'cancer or not ' info for every single one of these people. 'Cancer or not' info is indicated in the stage1_labels.csv file. For convenience, we call these patients with missing "cancer or not" information unlabeled patients. Since we cannot use unlabeled patients to feed into our CNN, we have to filter them out.

3.4 Validation Test Set Preparation

After filtering unlabeled patients, based on stage1_labels.csv, we have only 1397 valid patients left and determined that we could only take these 1397 patients into consideration. Among these 1397 patients, we used the hold-out method to feed 1297 training data into the CNN and use 100 data as our validation dataset.

4. Experiments

4.1 Resize Scan Image Data

In our experiment process, we first examined the image data of each patient, each scan image being the same size, 512*512, which is quite large to use for CNN. We choose to resize them into 150*150 by utilizing a Python package, CV2.

4.2 Normalize Scan Image Data

Since our lung scan data is in 3D fashion, after resizing 2D scan images, we decided to take a look at the last dimension where 2D scan images stack up in a certain order. We then found that the number of scan images for each patient was different but always over 100, meaning we needed to normalize the number of scan images for each patient before we proceeded further. Assuming the 3D lung scan was an ordered list of 2D scan images each with different length for each patient, for normalization (inspired by Sentdex on Kaggle and Ned Batchelder on Stackoverflow) we choose to reshaped the ordered list with different lengths into fixed length (we picked 20 here) list of list of scan images—fixed length list of chunk of scans in other words.

4.3 Convolutional Neural Network with 3D data

Traditionally, when a doctor looks through scan images and tries to find a suspicious malignant knot, he or she uses OsiriX and looks at each slice of the 3D scan image in a sliding window fashion. Similarly, we want to feed the 3D data into our CNN and look for suspicious patterns in every location in a 3D space. Therefore, besides finding the pattern of possible malignant knot, we also need to keep track of the location that the sliding window has visited, which can be done by proper design of the CNN algorithm. CNN is popularly used with image data nowadays. The basic structure is Convolution -> Pooling -> Fully Connected Layer -> Output.

The Convolution part takes in the original data, scans through every possible place in the 3D scan space and creates feature maps from it. In the Pooling part, it selects a region and then takes the maximum value in that region and sets it as the new value for the entire region. This is also called the process of downsampling. Each convolution and pooling step is a hidden layer. It was up to us to choose how many hidden layers we wanted and we choose 1 for the project. The last step before the output is a fully connected layer, which utilizes a typical neural network where all nodes are fully connected. Sigmoid function is usually the default activation function in this process. Based on our research, it is just this one layer that needs to be fully connected and it is not necessary for the convolution layer.

4.4 Convolutional Neural Network with Tensorflow

In this project, we created the CNN by using Tensorflow. Tensorflow was developed by the Google Brain team for internal Google use and has became an open source software library for deep learning in 2015. The benefit for us is that Tensorflow allows us to perform specific machine learning operations with large efficiency.

5. Results

The main evaluation method we choose to use is log loss function which is also the criteria used by Kaggle Data Science Bowl.

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right]$$

where

- n is the number of patients in the test set
- $\hat{y}_i$ is the predicted probability of the image belonging to a patient with cancer
- y_i is 1 if the diagnosis is cancer, 0 otherwise
- $\log()$ is the natural (base e) logarithm

Note: the actual predicted probabilities are replaced with max(min(p, $1\text{-}10^{-15}$), 10^{-15}). A smaller log loss is better.

We use accuracy as our other criteria for evaluation.

Accuracy = (number of correct labeled data / number of all data) ×100%

Epoch	**Loss**	**Accuracy**
1	207,534,499,032.0	0.60000002
2	18,922,230,385.25	0.60000002
3	7,375,737,343.5625	0.63999999
4	3,967,268,669.921875	0.68000001
5	2,411,880,741.71875	0.56
6	1,715,221,836.21875	0.61000001
7	1,108,410,138.5	0.63999999
8	840,641,015.8515625	0.61000001
9	348,965,611.36240035	0.57999998
10	341,554,514.41552579	0.58999997

validation		0.60000002

6. Conclusion

Due to the limited memory, we only chose stage 1 of the whole dataset of the Data Science Bowl 2017. Due to the time cost (each run takes several hours), we only ran our model 10 times. For each run, log loss decreased, which is good, although we run the risk of overfitting. The accuracy wobbled around 60%: the highest was 68% and the lowest was 56%. Overall, 60% really cannot be regarded as an ideal accuracy. Now we will look back our data.

An issue we had is that the training dataset has 1035 examples that are labeled as “non-cancerous” (negative) and 362 examples labeled as “cancerous” (positive). If we are concerned with early cancer detection, then, if we replicated an improved version of this project in the future, we would require a training dataset with more “cancerous” (positive) examples. Because the number of cancerous examples is so low, an algorithm that predicted no-cancer with this model *every single time* would still be 74% accurate! This astonishing. The validation has 28 examples with cancer and 72 without cancer. We think we are probably overfitting the training data, rendering the accuracy lower, and causing a decreasing log loss after the highest accuracy. Our goal is to improve this, with the end goal in mind being improving our ability to detect cancer accurately.

Future improvements must be considered. Likely the largest issue we have is the size of our data. If we're going to be successful with a neural network, we need more data. We can either hunt for more outside datasources, or we can create new data by adding some noise to the data. Even though our data is quite large, over 200 G, there are only 1397 valid patients we could use. To better train our algorithm and to better prevent overfitting, we could further train our algorithm with more external data. One possibility is to include patients data from the LUNA 16 dataset (there are 888 more scans available to use). The reason that this patient data can be included even though it is from a foreign dataset is because it shares the same information and data of lung scans. The ability to incorporate this data was suggested by the official Kaggle Data Science Bowl.

The second issue we can improve is that it might also just simply be the case that a neural network is not the best choice of model here. Neural networks are capable of amazing solutions, but this is only the case when we have the datasets to support the problem at hand. We might just simply not have enough data so perhaps a neural network was not the best choice.
We can consider adding neural network variables, like layers, learning rate, activation functions, optimizer, etc., or we could just consider other models as alternatives.

Work Cited

Sentdex. "Introduction to Deep Learning with TensorFlow." PythonProgramming.net, n.d. Web. <https://pythonprogramming.net/tensorflow-introduction-machine-learning-tutorial/>.

Sentdex. "First Pass through Data W/ 3D ConvNet." Kaggle. Sentdex, Feb. 2017. Web. <https://www.kaggle.com/sentdex/first-pass-through-data-w-3d-convnet>.

Stack Overflow User. "Introduction to Deep Learning with TensorFlow." *Stack Overflow*. N.p., n.d. Web. <How do you split a list into evenly sized chunks?>.

Zuidhof, Guido. "Full Preprocessing Tutorial." Kaggle. Kaggle Inc., Feb. 2017. Web. <https://www.kaggle.com/gzuidhof/full-preprocessing-tutorial>.