DataScience Bowl 2017: Lung Cancer Detection

Shyam Prabhakar[1]

[1]Department of Industrial Engineering

Abstract

Motivation: In the United States, lung cancer strikes 225,000 people every year, and accounts for $12 billion in health care costs. Early detection is critical to give patients the best chance at recovery and survival. One year ago, the office of the U.S. Vice President spearheaded a bold new initiative, the Cancer Moonshot, to make a decade's worth of progress in cancer prevention, diagnosis, and treatment in just 5 years. In 2017, the Data Science Bowl will be a critical milestone in support of the Cancer Moonshot by convening the data science and medical communities to develop lung cancer detection algorithms. Using a data set of thousands of high-resolution lung scans provided by the National Cancer Institute, participants will develop algorithms that accurately determine when lesions in the lungs are cancerous.

Data Availability: The data used is available at https://www.kaggle.com/c/data-science-bowl-2017/data

Contact: shyamprabhakar92@tamu.edu

1 Introduction

The Data Science Bowl 2017 aims at convening the data science and medical communities to develop lung cancer detection algorithms. The data set consists of thousands of high-resolution lung scans provided by the National Cancer Institute with the objective to accurately determine when lesions in the lungs are cancerous. The competition task is to create an automated method capable of determining whether or not the patient will be diagnosed with lung cancer within one year of the date the scan was taken. The ground truth labels were confirmed by pathology diagnosis.

Opening this up to the Kaggle community is a critical milestone in support of the Cancer Moonshot, to fast-track the progress in cancer prevention, diagnosis and treatment. Reducing the false positive rate that plagues the current detection technology and getting patients earlier access to life-saving interventions are the major expected outcomes from this initiative.

2 Preprocessing

Since each image has a variable number of 2D slices, there are Nx512x512 in a 3D rendering for one patient where N varies based on the machine taking the scan and the patient themselves. We definitely need to resize the images to apply machine learning/deep learning algorithms, so as not to lose too much information as well as keep within the computational restraints.
So, the preprocessing has been one of the most important steps in this project and there were a few different approaches taken to tackle this and prepare the data for Machine Learning purposes. First of all, a variable third dimension is a main problem. Since machine learning/deep learning algorithms expect a standard input size for each sample, we need to bring this to a manageable and uniform size. As a starting point, we can use an input size of 20x50x50.

So we map each of the set of CT scans for a patient from:

$$N * 512 * 512 \rightarrow 20 * 50 * 50$$

Now, apart from image size we have a few other improvements we can make to reading in the data from the dicom files. Converting the pixel values to Hounsfield Units (HU), resampling to an isomorphic resolution to remove variance in scanner resolution and things like lung segmentation, normalization, zero centering to name a few. So, considering all the different things we can do, I have two different preprocessing pipelines set up for the data.

1. A simpler process which focuses on getting all the patient's scans to a standard 3-dimensional representation,
2. More complex with most of the additional improvements mentioned above before transforming to a standard form as in Process 1

Let us refer to these as Process 1 and Process 2.

3 Modeling

The Convolutional Neural Network was a natural choice to employ in this problem because of their effectiveness in image classification problems. Properties like translation invariance and being able to preserve the spatial structure of images make them well suited for this application.

In terms of computational requirements, the ConvNet models could only be trained on GPU. The Tesla K80 GPU on the Terra cluster of TAMU HPRC was used in the training of these models. A sample model summary is shown. The number of parameters or weight optimizations is the important point to note. As you can see it is not a very deep model and even this has over 43 million weights.

Layer (type)	Output Shape	Param #	Connected to
convolution3d_1 (Convolution3D)	(None, 32, 16, 46, 46)	4032	convolution3d_input_1[0][0]
dropout_1 (Dropout)	(None, 32, 16, 46, 46)	0	convolution3d_1[0][0]
convolution3d_2 (Convolution3D)	(None, 32, 12, 42, 42)	128032	dropout_1[0][0]
maxpooling3d_1 (MaxPooling3D)	(None, 32, 6, 21, 21)	0	convolution3d_2[0][0]
flatten_1 (Flatten)	(None, 84672)	0	maxpooling3d_1[0][0]
dense_1 (Dense)	(None, 512)	43352576	flatten_1[0][0]
dropout_2 (Dropout)	(None, 512)	0	dense_1[0][0]
dense_2 (Dense)	(None, 128)	65664	dropout_2[0][0]
dense_3 (Dense)	(None, 2)	258	dense_2[0][0]
Total params: 43550562			

Residual Networks were also an interesting option that I wanted to pursue. I wrote up the code for a small 7-layer residual learning architecture but unfortunately this was just in the past few days and I have not had any successful runs of the code, no results to show. Here, the parameters of the Convolutional Network instead of trying to map a function, say H(x), are forced to map a residual function F(x), where H(x) = F(x) + x. The intuition behind this being that in the optimal case of identity mapping, it would be easier to push the residual to zero than to fit an identity mapping by a stack of non-linear layers. The skip connections seen in the residual architecture also help solve the problem of vanishing gradients.

XGBoost is a very popular machine learning model and very versatile. It can be applied to both classification and regression models, and is one of the most viral models over the past few years for Kaggle users. But this is a Computer Vision problem of considerable complexity and I did not expect it to perform well in this case. Nonetheless I wanted to test its performance against deep learning models for the problem.

The models were evaluated on the test set provided in the competition. Submissions are scored on the log loss.

4 Results

These are some model results that I have been able to produce. They have been organized in terms of the preprocessing pipeline used before feeding them into the model. The input dimensions have also been specified.

Preprocessing	Input dimension	Model	Logloss metric
Process 1	20x50x50	ConvNet	0.6017
	30x100x100	ConvNet	0.6014
	20x50x50	XGBoost	9.594
Process 2	20x50x50	ConvNet	0.65883
	20x50x50	XGBoost	9.942

It is notable that a larger representation of the scans with 30x100x100 as opposed to 20x50x50 has produced a lift in the performance of the models. The larger representation has enabled the use of deeper architecture and evidently has been able to allow the model to pick out better patterns to improve prediction performance.
However, I have honestly not been able to experiment very rigorously with the larger representation of the problem and the room for exploration it opens up. I turned to Residual Network architecture which I was not able to complete.

An estimate of computational times for the models is shown. These can vary depending on the architecture of

the models as well as the activity on the cluster at time of execution and are meant to be a rough guideline.

Computa-tion	Input dimension	Model	Time taken
GPU	20x50x50	ConvNet-4	23 min
	30x100x100	ConvNet-10	185 min
CPU	20x50x50	XGBoost	35 min

The hyphenated number next to the ConvNet model is meant to denote the number of layers in the networks prior to feeding the features to a fully connected layer.

Acknowledgements

The Kaggle website obviously deserves mention because of the extensive support and collaboration that it allows within the Data Science community. TAMU's HPRC was a great help in this project, providing not only an excellent platform to perform such large scale computation but a lot of support in helping set up the files and packages required to run the code. And thanks so much Professor for all your patience, support and motivation throughout the project. I only wish I could have done more! But I have learned a lot and will take that with me.

References

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770-778).

http://www.jiqizhixin.com/article/index/id/2655