Forecasting Lung Cancer Diagnoses with Deep Learning

Data Science Bowl 2017 Technical Report

Daniel Hammack

April 22, 2017

Abstract

We describe our contribution to the 2nd place solution for the 2017 Data Science Bowl [8], the largest Kaggle competition to date in terms of prize pool with a $1 million total pool and 2000 competing teams. The goal of the competition is to produce a system that consumes a CT scan and forecasts the probability that the patient in the scan will be diagnosed with lung cancer within a year of the scan. A less technical solution writeup is avilable at the author's github. [7]

1 Background

Lung cancer is one of the most common forms of cancer, especially in North America. It is the most common form of cancer in men and the second most common in women. Worldwide 1.6 million people die from lung cancer each year, and there are 225,000 new diagnoses of lung cancer per year in the U.S. alone. Furthermore, lung cancer is also one of the least survivable common cancers with an average 5 year survival rate of less than 20%. Early detection on average will at least double the survival rate of lung cancer. [2]

The Data Science Bowl (DSB) is an annual machine learning competition held on Kaggle. [9] The 2017 Data Science Bowl is the third DSB challenge and the largest to date in terms of number of competitors (2000) and prize pool ($1 million). The goal of the challenge is to build an automated system capable of forecasting whether a patient will be diagnosed with lung cancer in the next year given a CT scan. Only one CT scan per patient is provided and all patient-specific metadata is removed (so age, gender, weight, medical history cannot be used as indicators).

1.1 Tools Used

Our solution was done completely in Python making extensive use of the open source scientific libraries:

- keras [3]
- theano [11]
- numpy
- scipy
- scikit-learn [10]
- pandas

2 The Data

The data for this challenge are 1600 high-resolution chest CT scans with slice thickness less than 3mm. A CT scan is a 3D image with a single intensity value per location. The intensity values are integers and are measured in a standardized units given by the Hounsfield scale. The size of a typical scan is 512 x 512 x 400 volumetric pixels (voxels) which translates to approximately 100 million integers. In real world units, scans are usually around 30cm x 30cm x 40cm. Furthermore, the areas of interest (nodules) are typically on the order of 1cm^3. This means that it only takes 1/36,000th of a scan to significantly change the diagnosis. Given the large size of each scan, we decided to break the problem into smaller parts.

Each CT scan in the training set has an associated binary label. No additional patient metadata is available (e.g. age, smoking history, etc.), and only a single scan is available per patient.

2.1 External Data

We use external data from the LUNA16 grand challenge in our solution. [1] This is joined with data from the LIDC dataset (which is a superset of LUNA16) to give us detailed nodule annotations including features such as:

- diameter
- lobulation
- spiculation
- malignancy
- calcification
- sphericity

This dataset gives us an additional 900 CT scans, but more importantly detailed radiologist annotations and location information for 1200 nodules found in the new scans. The attributes that we focus on are diameter, lobulation, spiculation, and malignancy. There are other interesting nodule attributes in the LUNA16 dataset but they were discarded for varying reasons. The most important attribute by far is malignancy. Diameter is second, and lobulation and spiculation seem to add a small amount of incremental value.

3 Our Approach

There are 4 major steps in our solution:

1. Normalize CT scan
2. Find regions likely to have nodules
3. Predict nodule attributes
4. Aggregate nodule attribute predictions into a global patient-level diagnosis forecast

Ultimately our solution combines 17 3D convolutional neural network models and consists of two ensembles. The models in each ensemble were built using different architectures, training schedules, objectives, subsampled data, and activation functions. This added diversity makes combining the models more effective.

3.1 Data Normalization

Each CT scan is resized so that each voxel represents a $1mm^3$ volume. This is necessary so that the same model can be applied to scans with different 'slice thickness'. Slice thickness refers to the distance between consecutive slices (when viewing a 3D image as a collection of 2D slices) and can vary by up to 4x between scans. The scans are also clipped between -1000 and 400 Hounsfield units. Air has a value of -1000 HU and bone has 400, so values beyond these two endpoints are not informative for the diagnosis. After this, each scan is rescaled to lie between 0 and 1 with -1000 mapping to 0 and +400 mapping to 1. A crude lung segmentation is also used to crop the CT scan, eliminating regions that don't intersect the lung.

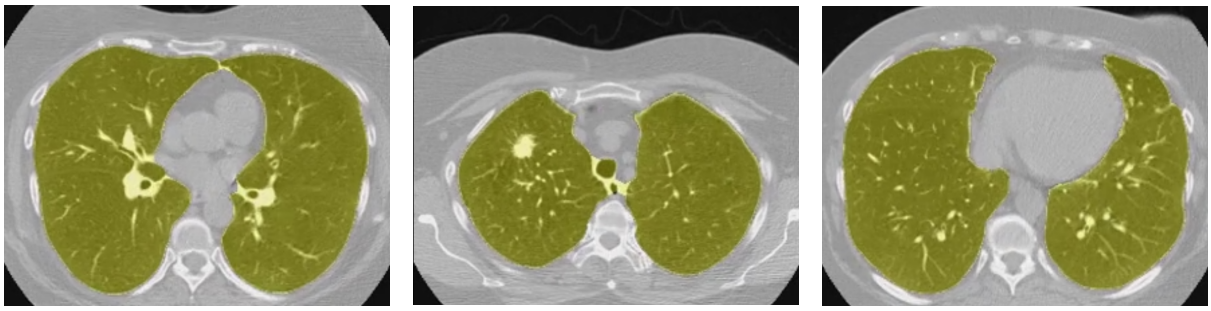

Figure 1: Lung segmentation example. Interior of lung has yellow tint. Animated gifs are available at author's GitHub.

3.2 Detecting Candidate Nodules

The second step in our pipeline is detecting regions of the scan which are likely to have useful information in diagnosis and disregarding areas that don't have any abnormalities. Taking a typical scan as an example, a raw scan would have size 300-400 mm^3 on each axis. As our models use 64 mm^3 subimages for diagnosis, all of our nodule models would need to be scored several hundred times per scan (depending on the step size used in the lattice). Furthermore even in cases of advanced lung cancer the vast majority of the lung can appear normal. Thus much computation is wasted if models are scored on regions of the lung with no abnormalities.

To speed things up, we built a model for identifying regions of the scan that appear abnormal. This model has an architecture similar to our nodule models, but it was trained on data with 90% normal samples and 10% abnormal samples. It used a regression objective with a single output representing a weighted combination of nodule attributes (with more weight for the more important attributes). Thus a larger value is assigned the more 'abnormal' the region.

Each scan is limited to have between 1 and 50 'abnormal regions'. If more than 50 are found, the 50 most abnormal are kept (and if none are found we pick the most abnormal regardless). An abnormal region is defined to be any 64 mm^3 region of the scan which received a prediction higher than a chosen threshold (our choice was 1 but it's relatively arbitrary as the units for this metric aren't meaningful).

Ultimately we don't believe this step is necessary if longer runtime is allowed. However given the limited-time nature of the competition we found this step to be useful as it allowed us to build bigger ensembles for describing nodule attributes.

3.3 Predicting Nodule Attributes

The most important part of our solution is our models that predict the 4 chosen nodule attributes (diameter, lobulation, spiculation, and malignancy). An early analysis showed that this is the most important part of the pipeline and thus we focused the majority of the computation on improving this step. We built two ensembles for nodule attribute prediction.

We found for the nodule attribute models (with our architecture), training was significantly improved by training on multiple objectives rather than just nodule malignancy (the most important objective). We suspect that training simultaneously on multiple objectives smooths out the gradients (as we were able to increase the learning rate substantially when training on multiple objectives). It was also found that 'branching' the model earlier produced better results when training with multiple objectives. If branching is done too late, the model outputs are too correlated (as they share too many parameters) and thus they provide overall less information in the next stage of the pipeline.

Test-time augmentation is used to improve the accuracy of our models at scoring time. We apply random 3D transformations of the input (which is 64 mm^3). Generally less aggressive transformations are used at test time as compared to the transformations at training time (mentioned in Section 4.1). We ultimately apply several transformations per input and average the results.

Figure 2 shows slices from a 3D saliency map of a nodule not seen during training. The image is colored based on the contribution of the pixels in each location to the malignancy prediction. Contributions are estimated by blacking out cubes of the image and recording the change in malignancy score.

3.4 Forecasting Diagnosis from Nodule Attributes

The last step in our pipeline is to forecast a cancer diagnosis given the nodule attribute predictions. The number of nodules per scan is variable and there are 4 predicted attributes for each nodule. We manually constructed features thought to be informative for diagnosis from the nodule attribute predictions. Our features are:

- max malignancy, spiculation, lobulation, and diameter across nodules (4 features)
- stdev of malignancy, spiculation, lobulation, and diameter predictions across nodules (4 features)
- location of most malignant nodule (3 features, one for each dimension in the scan)
- stdev of nodule locations in each dimension (3 features)
- nodule clustering features (4 features)

The nodule clustering features involve running a clustering algorithm on the nodule locations (in order to avoid double counting). A feature importance plot is shown in Figure 3. Clearly the maximum nodule malignancy prediction is the most important feature. The location of the most malignant nodule

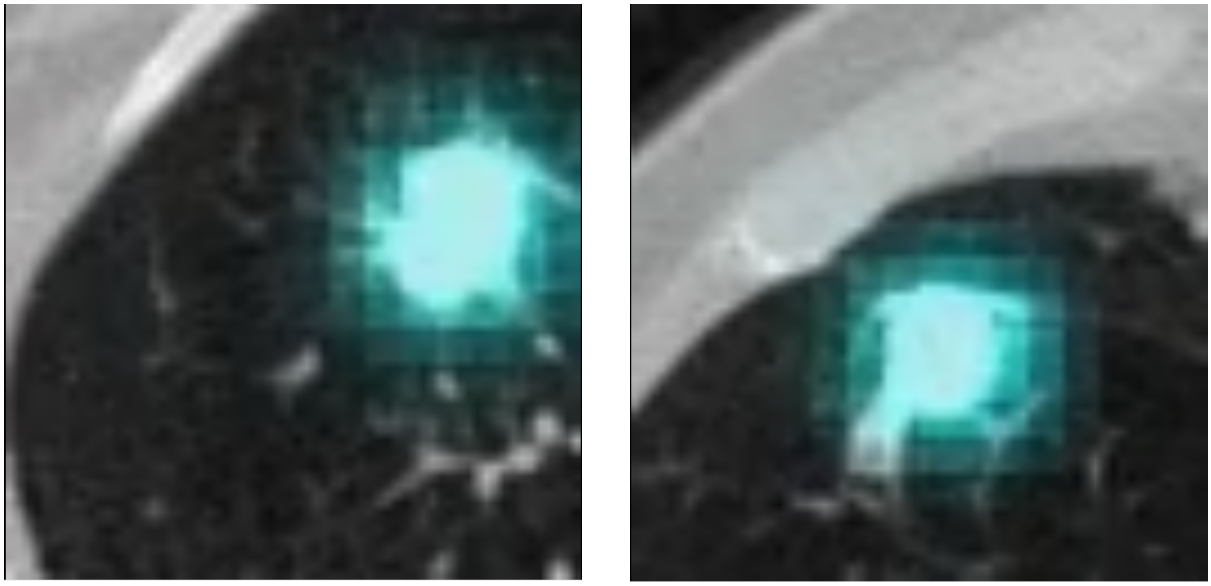

Figure 2: Slices of a 3D saliency map showing the most important parts of a 64 mm^3 section of a scan. Darker blue is more important.

in the Z dimension (closer to head vs closer to feet) is also significantly useful for diagnosis. We found that nodules in the 'superior lobe' (closest to head) were much more likely to lead to cancer diagnoses than in other locations of the lung.

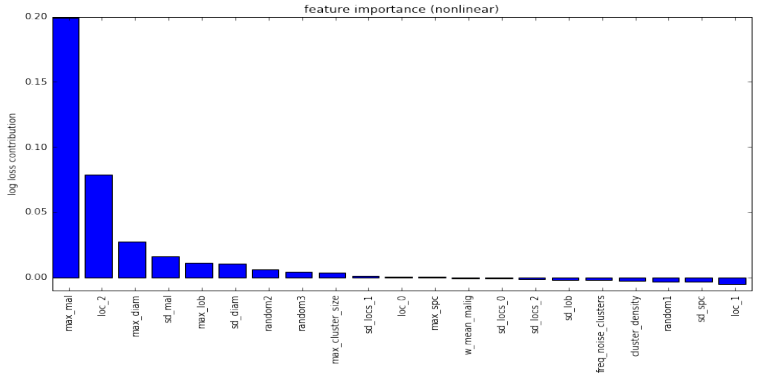

Figure 3: Feature importance plot. Y axis is increase in log loss when the feature is randomly permuted. 3 random features are added for reference.

These 18 features are fed into a 'slightly nonlinear' classifier for predicting the final diagnosis. The classifier consists of a L1-penalized logistic regression model followed by an Extremely Random Trees [6] regressor fit on the residuals from the linear model. This method is used because the relationships between predictors and the target are mostly linear but there are a few (less important) features with nonlinear relationships. Also one additional feature was added late in the competition - the output of Julian's mass detector model. This model is described in his blog post [4], and it predicts the amount of 'abnormal mass' in the lungs of a patient.

4 Neural Network Model Architecture and Training Details

Overall there are 17 3D CNN models built for our solution (with different hyperparameters) so it is not feasible to describe them all here. Thus we focus on the best performing and most prototypical model.

As mentioned previously, the input to all our neural network models are 64 mm^3 regions of the CT scan. Models are trained on a combination of LUNA16 annotated nodules and LUNA16 false positives. Models consist of 5 'conv blocks', followed by global max pooling and a nonnegative regression layer with a softplus activation (Figure 4 and Table 1). To help the model capture information at different

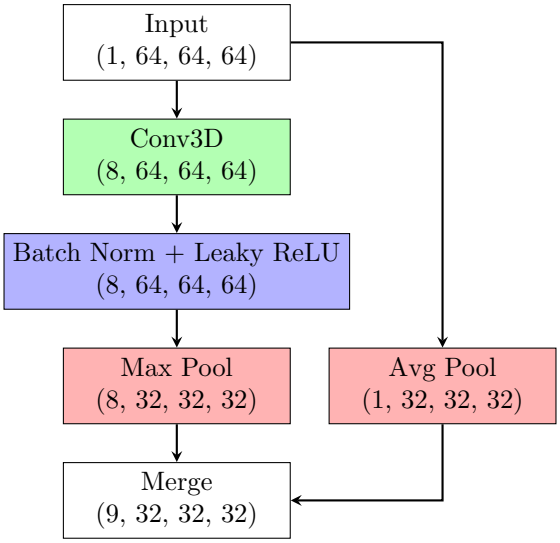

Figure 4: Structure of a conv block which forms the basis of our neural network models. Shapes are for the first conv block in the model.

scales the original input is downsampled and fed into each layer of the model, not just the first. Softplus activation is used because the targets for the model were non-negative (we also used scaled sigmoid in some models). Most models were trained with a MSE objective but some were trained with MAE and some with log loss.

We use 3D convolutions with filter size 3x3x3 everywhere, and pooling is always 2x2x2 with stride 2. Batch normalization is used after each convolution and max pooling is used for downsampling after batch norm. Models were typically built on 75% of the data and validated on the other 25%. Most of our models use the leaky rectifier activation function. The models that are used for detecting abnormalities were trained with 90% non-nodules and 10% nodules, and the models for predicting nodule attributes had the opposite distribution.

Layer Number	**Name**	**Output Shape**
0	Input	(1, 64, 64, 64)
1	conv block	(8, 32, 32, 32)
2	merge w/downsampled input	(9, 32, 32, 32)
3	conv block	(24, 16, 16, 16)
4	merge w/downsampled input	(25, 16, 16, 16)
5	conv block	(48, 8, 8, 8)
6	merge w/downsampled input	(49, 8, 8, 8)
7	conv block	(64, 4, 4, 4)
8	merge w/downsampled input	(65, 4, 4, 4)
9	4x conv block	4x (65, 2, 2, 2)
10	4x global max pooling	4x (65)
11	linear + softplus	4x (1)

Table 1: Typical architecture of a strong nodule attribute model. '4x' refers to four parallel copies of that layer - one for each output in our multi output model

4.1 Data Augmentation

Given that only 1200 positive samples are available to learn with (positive samples are the annotated nodules from LUNA16), we make extensive use of data augmentation to increase the effective size of the training set. We exploit all the symmetries of 3D space and use both lossless and lossy augmentation. We use random rotations by 90°increments, random transpositions, random zooming by small amounts,

random reordering of axes, and random arbitrary rotations by small degrees (-10 to 10). Doing this aggressive data augmentation acts as a strong regularizer while also teaching our model to be invariant to many common distortions of CT scans seen at test-time. The lossy data augmentation is quite computationally expensive so we did not apply those transformations in real time during training but had a parallel process continually rebuild different versions of the training set. The training set was reloaded with a newly augmented version after every few epochs.

4.2 Model Training

We used curriculum learning to speed up model training. Because our model uses a global max pooling layer, it can process any input of size 32 mm^3 or larger. Thus because images of size 32 mm^3 are 8x smaller than 64 mm^3, we trained our model first on inputs of this size for about 2000 parameter updates with a batch size of 128. Then we increased the input size to 64 mm^3 and trained for 6000 parameter updates with a batch size of 64. The first 25 (out of 30 total) epochs were trained with a random choice of 75% of the nodules (different for each model) and the last 5 were with the full training set. The learning rate started at 0.1 and was decreased stepwise every few epochs. The last few epochs of training use a very low learning rate of 3e-5 which we found to help. Models were built both on local hardware and AWS servers. Training time was around 8 hours on local hardware and 1.5 hours on AWS servers. Models were trained with the NAdam optimizer [5] (Adam with Nesterov momentum) from the Keras package. We did not experiment much with the choice of optimization algorithm.

5 Teamwork and Ensembling

This report describes only the author's contribution to the final submission. A team was formed with Julian de Wit late in the competition. Julian's solution [4] also involved building models on the LUNA16 data to predict nodule malignancy, though different choices were made in his pipeline. Ultimately both of the solutions produced excellent performance in cross-validation, and had a correlation around 0.88 thus there was a substantial gain in combining them. To see the details of Julian's solution, see his report. Julian's solution and our solution were combined by a weighted average at the forecast level.

References

[1] Luna 16 grand challenge. `https://luna16.grand-challenge.org/`.

[2] Cancer.org. Lung cancer survival rates. `https://www.cancer.org/cancer/small-cell-lung-cancer/detection-diagnosis-staging/survival-rates.html`.

[3] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[4] Julian de Wit. 2nd place solution for the 2017 national data science bowl. `http://juliandewit.github.io/kaggle-ndsb2017/`.

[5] Timothy Dozat. Incorporating nesterov momentum into adam. `http://cs229.stanford.edu/proj2015/054_report.pdf`.

[6] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[7] Daniel Hammack. 2nd place solution to 2017 ndsb. `https://dhammack.github.io/kaggle-ndsb2017/`.

[8] Kaggle. Data science bowl 2017. `https://www.kaggle.com/c/data-science-bowl-2017`.

[9] Kaggle. Kaggle homepage.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.