# Kaggle Data Science Bowl 2017: Lung Cancer Detection

Alaa El Khatib and Olivier Nguyen

University of Waterloo, Waterloo, ON, Canada.
alaa.elkhatib@uwaterloo.ca,
olivier.nguyen@uwaterloo.ca

**Abstract.** With recent developments in machine learning, deep learning and computer vision, there is great potential for improvements in early detection of lung cancer using CT scans. This paper details the methods and techniques used in our submission to Kaggle's Data Science Bowl 2017, where the objective is to develop algorithms to determine whether a patient has or is likely to develop lung cancer using CT scan images. We explore four approaches to address the problem and compare their results. The best approach among them obtained a rank of 535/1679 in the Kaggle competition.

**Keywords:** lung cancer detection, machine learning, deep learning

## 1 Introduction

Lung cancer is one of the most common types of cancer and accounted for over 1.5 million deaths worldwide in 2012, according to the World Health Organization [6]. Using low-dose CT scans, it is possible to assess if people are at risk of potentially developing lung cancer or other pulmonary diseases using machine learning methods. However, the current technology is still limited and yields a lot of false positives that lead to unnecessary follow up scans and patient anxiety. In the past year, promising results have been obtained for similar problems such as skin cancer detection [5]. In order to improve the current state of lung cancer detection, Kaggle launched a competition with 1,000,000$ in total prizes for the Data Science Bowl 2017 where the goal is to develop algorithms to determine from CT scans whether a patient will develop cancer or not. The competition aims to convene the data science and medical communities to make great leaps in cancer detection algorithms, more specifically early detection. Early detection of lung cancer is important because it allows for timely treatment and has potential to reduce deaths.

The way lung cancer is diagnosed is by inspecting a patient's CT scan images, looking for small blobs in the lungs called *nodules*. Finding a nodule is not in itself indicative of cancer; the nodules have to have certain characteristics (shape, size, etc.) to support a cancer diagnosis.

### 1.1 Datasets

**Kaggle** [1] provided the main dataset for the competition which contained CT scan images from 1397 patients for the training set that are labeled with cancer or no cancer. In addition, there are 198 patients used for the test set. A single patient's data may contain hundreds of CT image slices, as seen in Figure 1. In total, there are over 250,000 gray-scale images of size $512 \times 512$. The main difficulty faced when using this dataset, aside from its large size, is that the labels provided are at the patient-level only; in other words, no information is provided about the location and classification of nodules.

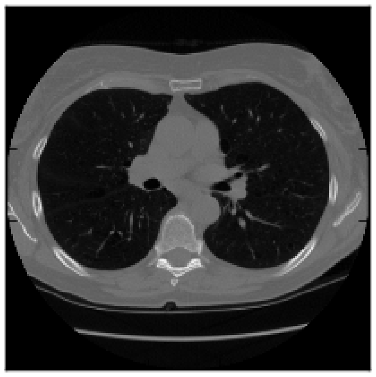

**Fig. 1.** Sample CT scan from the Kaggle dataset

**LUNA16** [2] is another publicly available dataset [13] which includes CT scans along with corresponding nodule locations annotated by 4 experienced radiologists. This dataset is used to train a neural network for the segmentation of nodules in CT scans, since the original Kaggle dataset does not contain nodule annotations. The dataset contains 888 CT scans of size $512 \times 512$ as seen in Figure 2.

## 2 Literature Review

Several approaches to address the detection and classification of lung nodules have been proposed in the literature, including both techniques that rely on traditional image processing and computer vision tools, and those that rely on representation learning and deep learning. Nodule detection generally relies on image segmentation techniques. Traditional image processing techniques make

[1] http://kaggle.com/c/data-science-bowl-2017
[2] http://luna16.grand-challenge.org

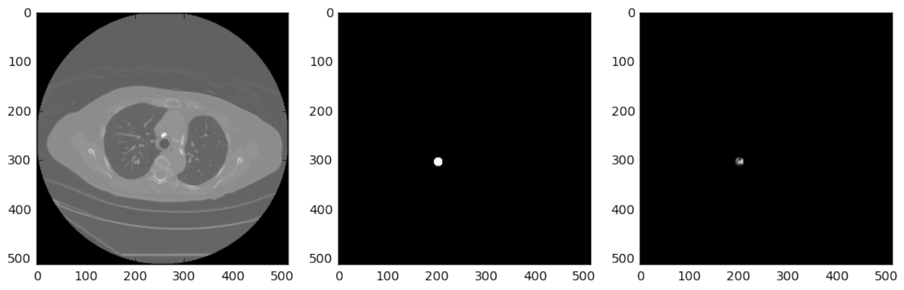

**Fig. 2.** Sample CT scan and nodule masks from LUNA16 [13]

use of intensity and geometrical shape features of connected components [2, 8], graph cuts and active contours [11], and fuzzy c-means clustering [10]. Recently, *U-Net* convolutional networks have shown to be good for segmenting neuronal structures and identifying regions with nodules [12]. A distinctive feature of the network is that it combines low-level features at the early layers with upsampled higher-level features at the final layers to make its predictions. The approach is much faster than those in which a classifier is trained to predict each pixel based on the surrounding patch [3], since it does not require the network to make pixel-level predictions. Other approaches rely on 3D information for detection, such as using a 3D CNN [7] to classify subvolumes as either containing a nodule or not, or using a voxel-level classification approach with a 3D CNN [1].

For the classification of segmented nodules, reported methods have used an autoencoder-based approach with a binary decision tree [ [9], deep features extracted from a CNN and passed to SVM and Random Forest classifiers [15], the Adaboost classifier [4], while earlier approaches relied on manually computed shape, texture, and size features of segmented nodules with shallow classifiers like LDA, quadratic discriminant and logistic regression [14].

## 3 Methods

In this section, we describe the preprocessing steps we used in our experiments, as well as four approaches to address the problem. Some of the preprocessing steps (mainly lung segmentation, extracting LUNA masks, and building the U-Net) follow the tutorial provided on the competition website [3] and use some of the provided code.

### 3.1 Preprocessing

**Lung segmentation.** In all approaches but the third, we start by detecting candidate nodule regions. In order to reduce the number of false positives generated in this process, it is helpful to isolate the lung regions (where nodules

[3] http://kaggle.com/c/data-science-bowl-2017

appear) and clear everything else in the CT scans. We do this following the Kaggle tutorial referenced earlier. The approach described in the tutorial relies on heuristics that seem to work well on both Kaggle and LUNA datasets. The lung segmentation step first isolates the lungs by thresholding the images based on the intensity of the pixels. Then, morphological filtering (dilation and erosion) is used to fill out the lung mask, patching any holes that could result from the thresholding.

**Z-score normalization** is used in all approaches. Each image in the CT scan images is centered by subtracting its mean and normalized by dividing by its standard deviation. We find that this approach works better than normalizing the entire dataset as a whole (using the mean and standard deviation of the whole dataset).

## 3.2 Nodule Detection

To detect nodules in the CT scans, we make use of the U-Net [12] neural network to segment the nodule regions. These detected nodules, or features extracted from them, are used later on as the input to a classifier to determine whether the patient will develop cancer or not. The U-Net also follows Kaggle's tutorial for the extraction of nodules, but we had to train our own version. It is trained on the normalized, segmented lung images from the LUNA16 dataset and predicts binary masks that represent nodule locations in a CT scan, as shown in Figure 2. During training, the input to the U-Net is a single slice and the output is the corresponding ground truth binary mask showing nodule locations.

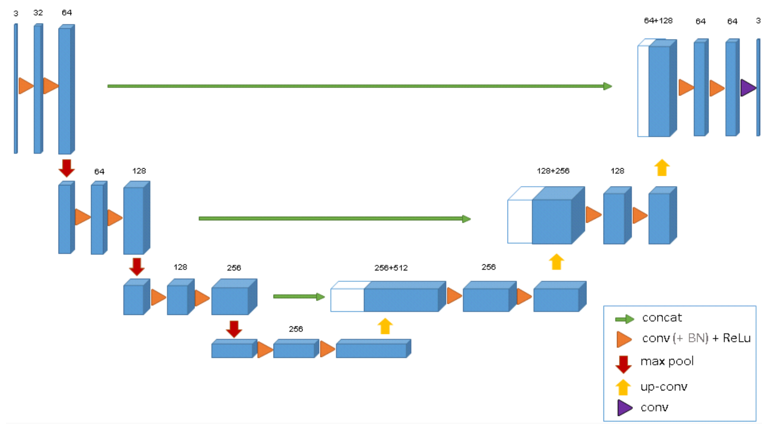

**Fig. 3.** U-Net Architecture [12]

## 3.3 First Approach

In our first approach to lung cancer detection, we start by detecting nodules in each patient's CT scans and extracting patches around them. Then we encode these patches using one of three encodings (to be described next), and use the average encoding of all the patches returned for a patient as the feature vector representing them. Finally, we train an SVM classifier on these feature vectors.

The nodule detection is done using the U-Net described in section 3.2. To reduce the number of false positives returned by the U-Net further, we follow a simple heuristic in this approach: for any nodule in any slice, if the U-Net does not detect a nodule next slice (with the set of slices ordered by position) in the vicinity of that nodule, we discard it. The assumption here is that nodules extend over multiple slices, and consequently, a nodule that appears in just the one slice is assumed to be a false positive. (In the second approach in 3.4, we describe a more systematic way of addressing false positives; we describe this approach in order to remain faithful to the methods we implemented.) We extract $48 \times 48$ patches around the remaining candidate nodules.

Next, we use one of three methods, autoencoders, local binary patterns, and flattening, to encode these patches.

**Autoencoders** are a special kind of neural networks that are trained to map the input onto itself, with the constraint that one of the hidden layers (usually called the bottleneck layer) have a lower dimensionality than the input. In this sense, they represent a class of dimensionality reduction techniques. A convoutional autoencoder is an autoencoder that uses convolutional layers (using at the beginning of the encoder and the end of the decoder). We use a convolutional autoencoder and train it on all the patches extracted from all the patients. Then, we use the autoencoder to encode each $48 \times 48$ patch as a 64-dimensional vector. The architecture of the autoencoder used here is shown in Fig. 4.

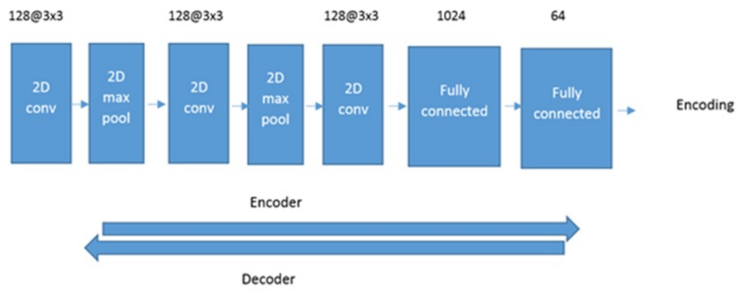

**Fig. 4.** The convolutional autoencoder used in the first approach.

**Local Binary Patterns** is a popular descriptor in Computer Vision used to capture the textural content of images. Each pixel is giving a binary code based on its value relative to its neighboring pixels. This is done by giving a value of 1 neighboring pixels that have values greater than or equal to the value of the pixel being considered and a value of zero to those with lower values. This results in an $n$-bit binary code describing each pixel, where $n$ is the number of neighbors. This process is illustrated in Fig. 5. The histogram of these binary codes in an image is used as the feature vector describing that image.

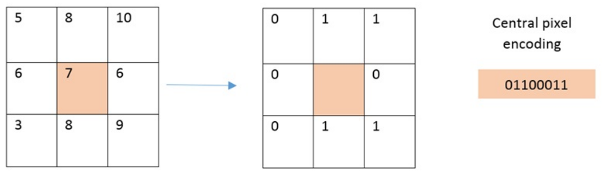

**Fig. 5.** An illustration of LBP on a $3 \times 3$ image patch. The binary code is computed for the center pixel.

**Flattening.** The last approach to encoding the candidate patches is to use the raw patch pixels as the encoding. In other words, we merely flatten the $48 \times 48$ patches into vectors.

These three methods provide encodings for the patches. Of the three encodings, using the flattened vector representation yielded the best results on a validation set, and consequently is the one used in the Kaggle submission. However, in the end we need a representation on the patient-level; i.e., we need a single feature vector describing each patient, since the given labels describe whether a patient is diagnosed with cancer or not, but do not say anything about nodules. To this end, we use the average encoding of all the patches detected for a given a patient as the final feature vector representing that patient. Finally, we train a Support Vector Machine (SVM) classifier with a Radial Basis Function (RBF) kernel on these feature vectors.

## 3.4 Second Approach

In the previous approach, we followed a simple heuristic to remove the false positives in the nodule detection phase, which involved checking subsequent slices for similar detections. However, we noticed there were still many false positives in the detected nodules, which caused problems when training a classifier. To circumvent this, we develop an approach which involves keeping nodule detections that are found in regions where multiple detections were made across all slices for a single patient. The second approach generates a heatmap to combine the detections of the U-Net from section 3.2 on all the slices for a single patient. Figure 6 shows all the nodule regions detected for a patient with the regions that are more red indicating more detections at those pixel locations. It can be seen that there are false positive detections in various places of the CT scan, but only a few regions have multiple detections i.e. the red blobs with higher intensity. The heatmap is then thresholded to keep only the pixels that have had repeated detections. The idea behind this approach is that the regions that have had multiple detections from the nodule segmentation are more likely to be actual nodules. From there, only the slices that overlap with the thresholded heatmap nodule regions are kept. Dice coefficient is the metric used to evaluate how much a segmented nodule overlaps with one of the nodules of the heatmap, which is defined as:

$$DC = \frac{2\,|X \cap Y|}{|X| + |Y|}$$

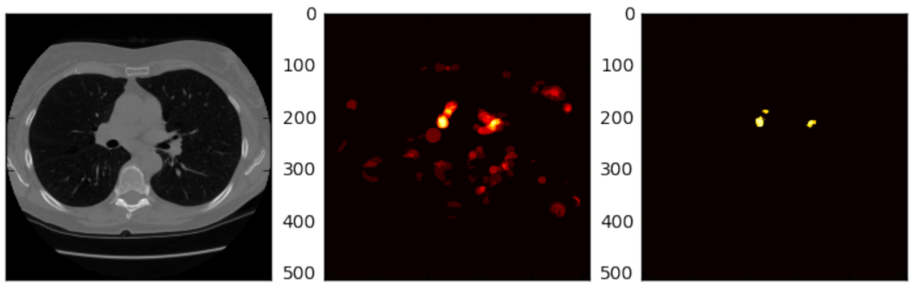

**Fig. 6.** Left: raw CT scan. Middle: heatmap generated from all slices from a patient. Right: thresholded heatmap to keep regions with multiple detections

If the detected nodule is over a certain value for the dice coefficient, it is kept and cropped to a $50 \times 50$ image to keep as a true detected nodule for that patient, and as data for the nodule classification. Moreover, unlike the first approach, which averages nodule encodings to arrive at a patient-level representation that corresponds to the provided labels, in this approach, a detected nodule is labeled cancerous if the patient has cancer and non-cancerous does not.

The data obtained from these cropped nodule detections are then used to train a classifier to predict if a patient has cancer or not. We first use a convolutional neural network that contains two convolution layers, 1 max-pooling layer, 2 drop-out and 2 fully-connected layers, which can be seen in Figure 7. Alternatively, we experiment with an autoencoder to extract features from the cropped nodule images and use the compressed representation to train a neural network classifier.

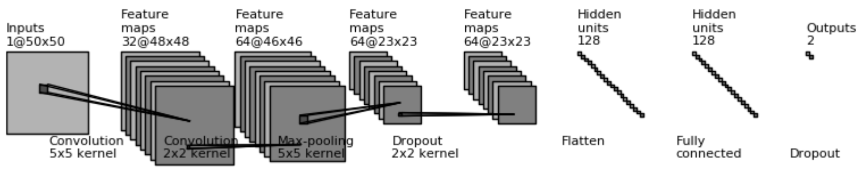

**Fig. 7.** CNN Architecture for nodule classification

### 3.5 Third Approach

The third approach we describe takes a different path to addressing the problem in that it does not attempt to detect nodules at all. Instead, we use the entire 3D volumes of the patients' CT scans (Fig. 8 shows a visualization of such a 3D volume) as input to a 3D CNN. By a 3D volume we mean all the CT scan slices of a patient concatenated in the correct order. 3D CNNs are similar to 2D CNNs, except they operate on 3D data and use 3D convolutions.

The advantage of this approach is that, again, it does not rely on detecting nodules. Hence, we do not have to handle false positives or look for a way to map nodules to the patient-level labels. The only preprocessing involved in this approach is that we normalize each slice first, and truncate the 3D volumes or pad them so that they have the same shape (since the number of slices varies from patient to patient).

On the other hand, this approach suffers from the fact that it is memory intensive. In our experiments, we use 3D volumes of shape $120 \times 512 \times 512$. For the parameters usually used, such as a batch size of 32 and, say, 128 units in the first layer, the output of the first layer has shape $32 \times 120 \times 512 \times 512 \times 128$, the output of the first layer is roughly 500GB! In order to train this model, then, we had to simplify our model considerably. The final model used is shown in Fig. 9. Notice how the first layer has only 4 units, a relatively low number of units for the first layer in a CNN. Moreover, we had to train the model using stochastic gradient descent (i.e., a batch size of 1).

### 3.6 Fourth Approach

The last approach we tried is similar to the second approach with a CNN, except that the heatmap method to reduce false positives is substituted for the heuristic method described in the first approach. As with the second approach, during training, patches are labeled cancerous if they come from a cancer patient, and non-cancerous otherwise. During testing, the model outputs a probability for each nodule detected for a test patient, and the final predicted probability is taken as the average of those probabilities.

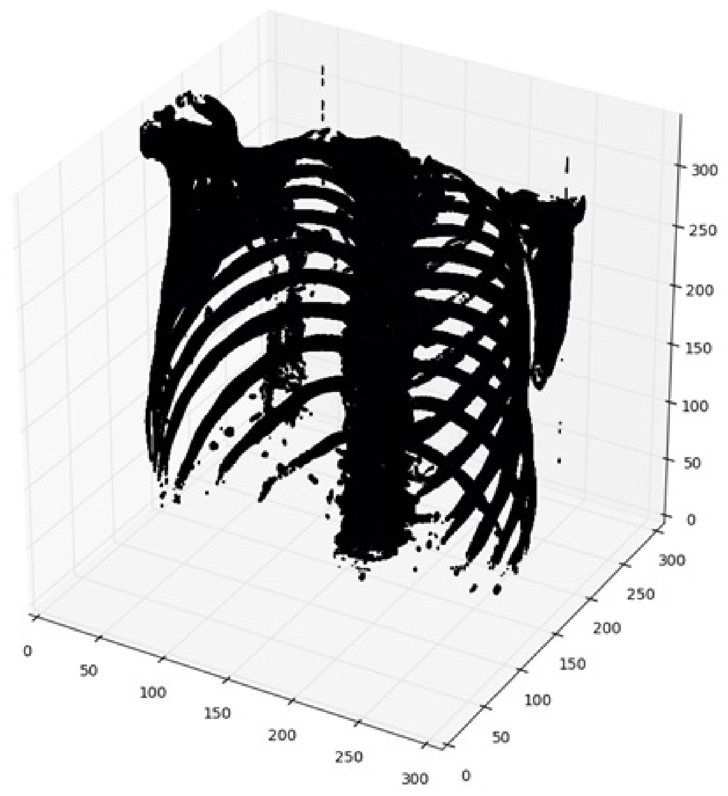

**Fig. 8.** 3D plot of full set of slices from one patient, ordered by position.

## 4 Implementation and Results

We implemented our methods using python, in addition to mainly the following libraries: numpy [4], scikit-learn [5], scikit-image [6], tensorflow [7], and keras [8]. We used two NVidia GPU machines, one with Titan X, the other with GTX 1070.

To evaluate the performance of the U-Net, we assumed that a nodule was detected correctly if the U-Net detected a nodule within 10 pixels of its correct location in the annotated mask. The U-Net described in 3.2 achieved a true positive rate of 85% and 65% on train and test sets, respectively, of the LUNA16

[4] http://www.numpy.org/
[5] http://scikit-learn.org/
[6] http://scikit-image.org/
[7] https://tensorflow.org
[8] https://keras.io

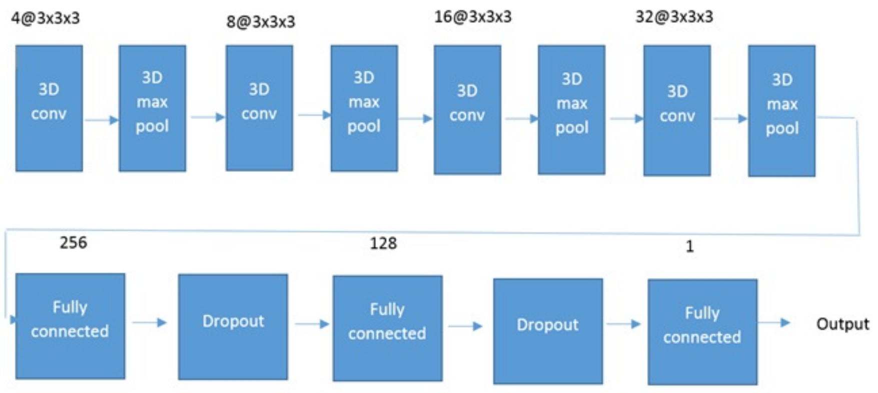

**Fig. 9.** 3D CNN used in third approach.

dataset, and an average false positive rate of 0.14 and 0.4 per slice on the same, respectively.

For the submission to the Kaggle competition, a file is uploaded with a patient id and a probability of cancer for that patient based on their CT scans. The submissions are scored based on the log loss:

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

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

The Kaggle dataset contains CT scans from 198 patients that are not labeled; these patients form the test set for the submission on which we are scored.

**Table 1.** Kaggle submission scores from all approaches

| | Log loss | Kaggle rank |
|---|---|---|
| Approach 1 | 0.57779 | 535 |
| Approach 2 + CNN | 0.58122 | 553 |
| Approach 2 + Auto-encoder | 0.60956 | 742 |
| Approach 3 | 0.60000 | 726 |
| Approach 4 | 0.59132 | 585 |

As can be seen in Table 1, approach 1 yielded the highest score on Kaggle, with a logloss of .57779 and a rank of 535.

Using approach 2 from 3.4, we obtained much fewer false positives after using a U-Net for the segmentation of nodules. Figure 10 shows predicted nodules for a single patient. From the image, we see that there are still a few false positives, but we managed to reduce the number of detections to 11 nodules detected per patient, down from close to 50 nodules detected per patient, which is now more realistic, as a nodule should only span a few CT scan slices. The classification using the convolutional neural network and the auto-encoder with the heatmap to remove false positives, however, did not bring any improvement from the results of approach 1 in section 3.3, yielding a log loss score of 0.58122 and 0.60956 respectively.

Approach 3 seemed to perform slightly worse than approach 1 and approach 2, yielding a logloss of 0.6. This is possibly due to the simplifications we were forced to make to the memory-intensive model.

Approach 4 also performed slightly worse than approach 2, yet slightly better than approach 3.

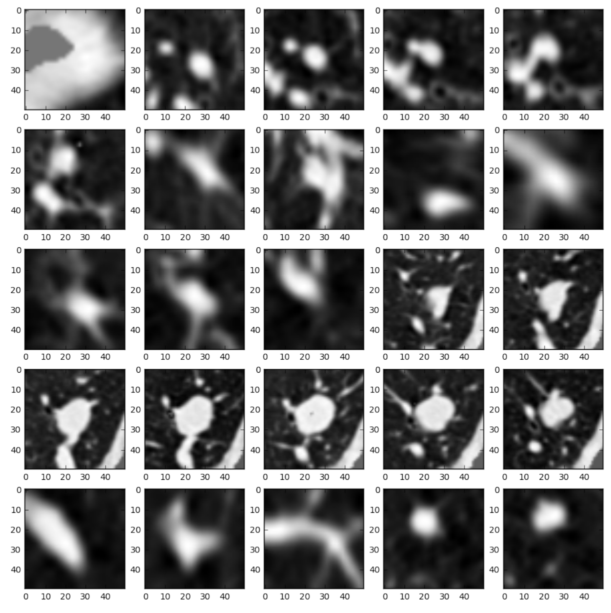

**Fig. 10.** Predicted nodules using the heatmap for a single patient

## 5 Discussion and Conclusion

In this paper, we describe the methods, implementation steps, and results of our submissions to Kaggle's Data Science Bowl 2017 on lung cancer detection. We evaluated four different approaches that used various machine learning techniques. In the first part, we made use of a U-Net for the segmentation of potentially cancerous nodules from patient CT scans. A simple heuristic is used to remove false positives by keeping nodule detections only if they are detected in the the following slice. We explored three different ways to encode the patches in this approach, and found, rather unexpectedly, that using the raw pixels yielded better results than more sophisticated methods such as autoencoders and local binary patterns. To resolve the issue of the unavailable nodule-level labels, we computed a patient-level average feature vector from the encoded patches. Using this approach, we obtained our best score and a rank of 535/1679. In the second approach, we tried to reduce the number of false positives by using a heatmap to combine all nodule detections for a patient, and thresholding it to keep the regions that have the most detections. Despite the additional step to remove false positives, the score obtained did not improve from the first approach. This could be explained by the fact that the nodule detection has a lot of false negatives; it is not segmenting nodule regions that should be identified and cropped. Additionally, it was seen that there are still many false positives using the heatmap. This could be better addressed with a better fine-tuning of parameters like the threshold value to keep the nodules across all the slices, and the dice coefficient threshold which defines how much a nodule must overlap in the heatmap regions to be kept as a nodule for classification. In the third approach, we used all slices for a single patient with a 3D convolutional neural network to exploit the volumetric data. While this approach required the least preprocessing and the least assumptions or approximations as far as extracting and labeling individual nodules are concerned, the 3D model required excessive amounts of memory that we did not have. To handle this, we had to simplify our model significantly, which is probably what lead to the weak performance. Finally, in approach 4, we used a modification of approach 2 that replaced the heatmap with the method described in the first approach to reduce the number of false positives. This lead to a slight decrease in performance compared to the second approach.

The main challenge we faced in this project was the compute and memory resources needed to handle the large size of the datasets involved. In addition, the labels provided with the main dataset were rather challenging, as they did not refer to nodule locations at all. Within the approaches described above, we explored three different methods to handle the labels: averaging nodule encodings per patient, labeling nodules with the same label given to the patient, or, finally, to not do nodule detection at all and opt instead for a 3D model on the raw images. Of these, the first method seemed to have given the best score.

## References

1. Rushil Anirudh, Jayaraman J Thiagarajan, Timo Bremer, and Hyojin Kim. Lung nodule detection using 3d convolutional neural networks trained on weakly labeled data. In *SPIE Medical Imaging*, pages 978532–978532. International Society for Optics and Photonics, 2016.
2. Margrit Betke and Jane P Ko. Detection of pulmonary nodules on ct and volumetric assessment of change over time. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 245–252. Springer, 1999.
3. Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, pages 2843–2851, 2012.
4. Martin Dolejsi, Jan Kybic, Michal Polovincak, and Stanislav Tuma. The lung time: Annotated lung nodule dataset and nodule detection framework. In *SPIE Medical Imaging*, pages 72601U–72601U. International Society for Optics and Photonics, 2009.
5. Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017.
6. Jacques Ferlay, Isabelle Soerjomataram, Rajesh Dikshit, Sultan Eser, Colin Mathers, Marise Rebelo, Donald Maxwell Parkin, David Forman, and Freddie Bray. Cancer incidence and mortality worldwide: sources, methods and major patterns in globocan 2012. *International journal of cancer*, 136(5):E359–E386, 2015.
7. Rotem Golan, Christian Jacob, and Jörg Denzinger. Lung nodule detection in ct images using deep convolutional neural networks. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 243–250. IEEE, 2016.
8. Metin N Gurcan, Berkman Sahiner, Nicholas Petrick, Heang-Ping Chan, Ella A Kazerooni, Philip N Cascade, and Lubomir Hadjiiski. Lung nodule detection on thoracic computed tomography images: Preliminary evaluation of a computer-aided diagnosis system. *Medical Physics*, 29(11):2552–2558, 2002.
9. Devinder Kumar, Alexander Wong, and David A Clausi. Lung nodule classification using deep features in ct images. In *Computer and Robot Vision (CRV), 2015 12th Conference on*, pages 133–138. IEEE, 2015.
10. Fan Liao and Chunxia Zhao. Improved fuzzy c-means clustering algorithm for automatic detection of lung nodules. In *Image and Signal Processing (CISP), 2015 8th International Congress on*, pages 464–469. IEEE, 2015.
11. Negar Mirderikvand, Marjan Naderan, and Amir Jamshidnezhad. Accurate automatic localisation of lung nodules using graph cut and snakes algorithms. In *Computer and Knowledge Engineering (ICCKE), 2016 6th International Conference on*, pages 194–199. IEEE, 2016.
12. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.
13. Arnaud Arindra Adiyoso Setio, Alberto Traverso, Thomas de Bel, Moira SN Berens, Cas van den Bogaard, Piergiorgio Cerello, Hao Chen, Qi Dou, Maria Evelina Fantacci, Bram Geurts, et al. Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: the luna16 challenge. *arXiv preprint arXiv:1612.08012*, 2016.

14. Sumit K Shah, Michael F McNitt-Gray, Sarah R Rogers, Jonathan G Goldin, Robert D Suh, James W Sayre, Iva Petkovska, Hyun J Kim, and Denise R Aberle. Computer-aided diagnosis of the solitary pulmonary nodule 1. *Academic radiology*, 12(5):570–575, 2005.
15. Wei Shen, Mu Zhou, Feng Yang, Caiyun Yang, and Jie Tian. Multi-scale convolutional neural networks for lung nodule classification. In *International Conference on Information Processing in Medical Imaging*, pages 588–599. Springer, 2015.