Data Science Bowl 2017
Model Description Team DL Munich

Preliminary thoughts We chose to rely entirely on convolutional neural network (CNN) classifiers since the given problem was a binary classification task with spatially correlated input data, for which CNNs are known to be highly performant. The most successful approach for similar classification problems in recent machine learning (ML) research is to directly feed the input data into a CNN and train it end-to-end against the label. From our point of view the main difficulty in this competition is the fact that end-to-end training is hard to realize due to the large input dimension (around 300 × 512 × 512 pixels). This causes two problems: First, it is hard to fit a model of reasonable size into the memory given current hardware limitations. Second, even if one manages to fit the model into memory, the input parameter number is two magnitudes larger than in comparable image recognition tasks, which have much larger training datasets. Therefore, training a generalizing model directly on the raw data source is most likely infeasible or at least suboptimal. So our main strategy was to design a framework which reduces the number of input parameters, with the smallest possible information loss, to a level where end-to-end training becomes possible.

Design Our model solely relies on the nodule annotations of the *LUNA16*-dataset, the *LIDC*-dataset and the *Data Science Bowl 2017* dataset. The general approach consists of four main steps:

1. Preprocessing of the raw CT-scan data.
2. Nodule segmentation in CT-scan slices.
3. Proposing nodule candidates.
4. Classifying cancer on patient level based on multiple nodule candidates.

We use a combination of segmentation and classification neural networks to solve the Data Science Bowl 2017 task. All models used in the pipeline were CNNs implemented with TensorFlow.

Preprocessing

We resample all lung CT-scans to an isotropic $1\times1\times1\,\mathrm{mm}^3/\mathrm{px}^3$ resolution. Additionally, we reduce the pixel number per CT-scans by only using the minimum bounding cube which includes the entire lung of a given CT-scan. We find the lung by segmenting each slice with a lung segmentation neural network. This network is trained with around 150 manually annotated lung wings. The process is illustrated in figure 1.

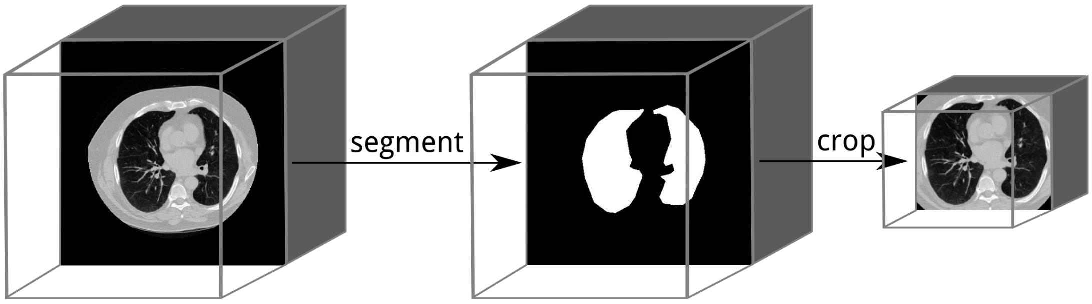

Fig 1: Lung wing segmentation pipeline.

Nodule segmentation

We reduce each Lung CT-scan to a fixed number of regions of interest. Therefore we train a multichannel segmentation network (standard UNET architecture) with the LUNA/LIDC information to segment lung nodules. The network is trained on random $128\times128\,\mathrm{px}^2$ crops of lung slices which are sampled randomly from all 3 room axis (not only axial) with the corresponding nodule annotation mask (we use elliptical annotations). Additional to the central slice we also include +2/-2 slices for input. The target is the segmentation mask of the central slice. The sampling process is illustrated in figure 2.

We do not sample slices from the entire lung, but exclusively slices which contain a nodule annotation. Additionally we sample 15 random negative slices for each room axis from each patient scan, which do not contain any nodule annotation. A training batch consists of 30 random nodule sectioning slices and 2 random negative slices. At prediction time we apply the trained segmentation network to all scans of the dsb3 dataset from all 3 room axis. This results in three 3D probability maps which will be averaged. This final 3D probability map tensor is the basis for the candidate proposal clustering.

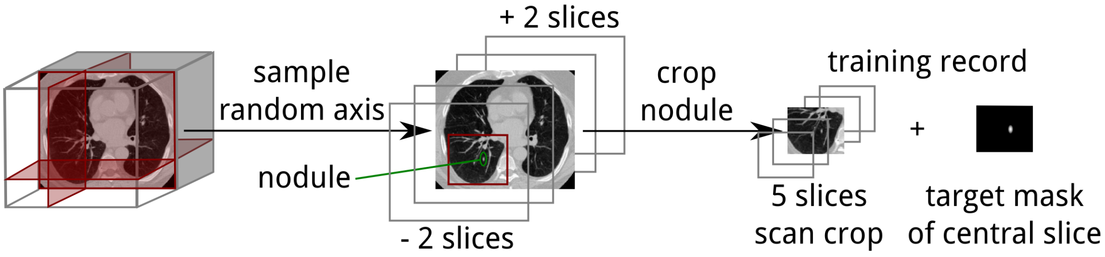

Fig 2: Illustration of slice sampling for training the nodule segmentation network.

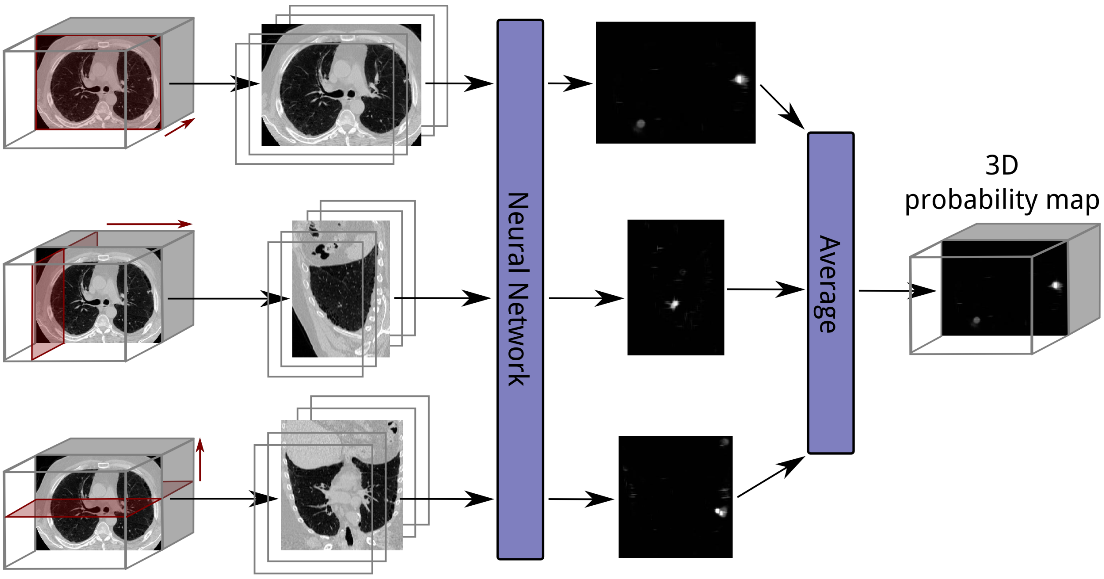

Fig 3: Prediction pipeline for the segmentation network. We apply the trained segmentation network on all slices of the three room axes. The resulting three 3D probability maps are averaged.

Candidate Proposal

Regions in the probability map tensor that have a high probability for being nodules have been identified using DBSCAN. DBSACN is a connected component analysis-type clustering algorithm. We used the implementation of scikit-learn.

We chose a relatively low threshold on the probability map to generate an input for DBSCAN and deal with the high number of proposed regions that results from this in two ways.

1. Regions that were too large to be processed in the later stage of cancer classification were again clustered with a threshold just sufficiently high for

producing small enough candidates.

2. Proposed regions were sorted according to a score compute as the sum over the m highest pixel values in the region, where m is approximately the number of pixels in the smallest annotated nodule.

The resulting cluster centers are used to crop candidates from the original data in $0.5{\times}0.5{\times}0.5\,\mathrm{mm}^3/\mathrm{px}^3$ and $0.7{\times}0.7{\times}0.7\,\mathrm{mm}^3/\mathrm{px}^3$ resolution.

Malignancy At this step it would be straight forward to apply a malignancy classification network to each of the extracted candidates, and assign an malignancy score to each nodule. This malignancy score can be used to propose only malignant nodules to the patient classifier. Unfortunately, we did not notice the malignancy information contained in the LIDC dataset. There may be room for improvement here.

Cancer classification on patient level

The "Patient Classifier" uses multiple candidates from the candidate proposal steps as input and predicts a single cancer probability for each patient. We use 5 to 20 candidates per patient. Additionally to the pixel information from the lung CT-scan we include the nodule probability map information from the segmentation network in a second input channel. Therefore, the classifier maps a tensor of shape (C,64,64,64,2), where $C = 10$ is the number of candidates, to a single output probability. The classifier is either a 3D residual convolution network of an 2D residual convolution network. In the case of 3D classifier each candidate is represented by the full (64,64,64,2) tensor. In the case of the 2D classifier each candidate is presented as the 3 center slices from all 3 room axis. The package of these three layers then is treated by the network as a multichannel input (no weight-sharing for different axis).
When applied to a single candidate the 3D residual core model reduces the candidate tensor from (64,64,64,2) pixels to an (8,8,8,128) feature map, by applying multiple stacked convolution and pooling layers (details in appendix / code). We reduce this final feature tensor with shape (8,8,8,128) with an 1x1 convolution layer with a single kernel to (8,8,8,1). A final global average pooling followed by a sigmoidal activation reduces this map to the single final output. We apply this residual core model to all candidates of a patient simultaneously and share the weight in the candidate dimension. When applying the core-module to a multi-candidate input with shape (C,64,64,64,2) we arrive at an (C, 1) output tensor. We then apply a reduction function (max) to this tensor in the second dimension

to obtain a single value. The entire graph is schematically shown for the 2D case in figure 4.

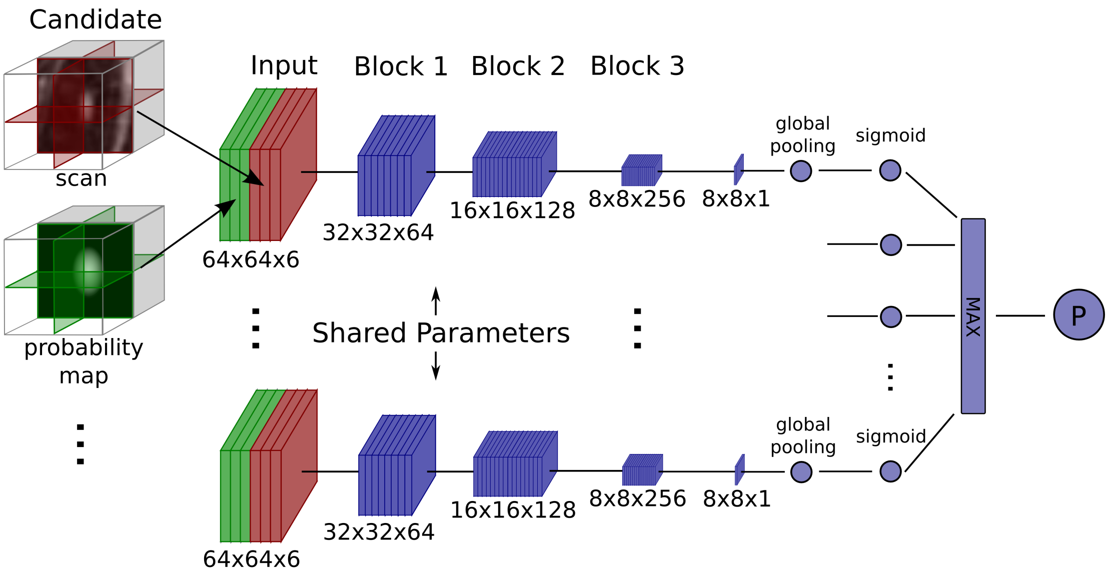

Fig 4: Scheme of complete model graph for the 2D multi-candidate residual network. Since sharing of parameters and a global reduction function, this graph can be applied to an arbitrary number of input candidates at prediction time.

In the following section we provide additional information and the reasoning behind the design choices for the proposed structure.

Probability map input Including the probability maps lowered the error on our local holdout set (280 patients) from around 0.46 to 0.44 and significantly reduced training time. Training networks with the probability map input reduces convergence time by a factor of two. This is especially helpful for the 3D networks, which train around 10 hours. We have two main explanation attempts why the inclusion of the probability maps improve performance:

1. Since we train the segmentation network on a larger patch of the lung slice, the probability map can contain more context information than the crop, which is provided as input for the classifier. Here is an example for illustration: The segmentation network can recognize a certain nodule because its characteristic distance to the lung wall and draws a more intense/larger/... probability map. When cropping the nodule with a smaller crop size, the wall vanishes as well as the crucial hypothetical distance feature. This would lead

to a worse prediction. The same logic applies for pairs of nodules. Where the existence of a second nodule raises the probability for the first one. By including the probability map of the segmentation network, the "out of crop range"-features can still be considered by the classification network.

2. The segmentation map can be used as an "attention" mechanism for the second classification network. By looking at the segmentation map the classification network has a trivial indicator which pixels in the input are the most important ones - namely the ones where the probability map channel has non-zero values. Since for this competition the amount of available training data is very limited this can counteract overfitting, since it favors the generalizing solution (looking at the nodules) for classifying cancer, instead of remembering e.g. characteristic pixels in the upper left corner of the input and simply remembering that this patient has cancer (which of course would not translate to unknown data).

Multi-Candidates-Network The proposed multi-candidate setup is directly end-to-end trainable from X candidates per patient to a single prediction per patient, while facing as little as possible inconsistency as the dsb data allows. E.g. in the alternative case when training each candidate separately against the patient label, the classifier is repeatedly forced to predict cancer from benign / non-cancerous candidates, since its from our medical knowledge its absolutely plausible that a single candidate is malignant (and therefore correlates with the patient cancer label) while most other candidates (nodule or not) are benign or simply healthy structures. This artificially introduced noise may distort the training process.

Weight-sharing between candidates Enabling weight-sharing in the candidate dimension is beneficial due to two reasons. First, the ordering of the candidates is expected to be imperfect and partially stochastic in respect to their correlation with the actual patient cancer label. This is due to the fact that our candidate proposal pipeline is optimized to find and sort all nodules - not necessarily malignant ones. So using different weights on an index-1 candidate as on an index-2 candidate makes no sense, since their index carries very little relevant information. Second, due to the fact that all candidates in principle share the same or at least very simple features learning a feature extractor on candidate 3 is most likely also beneficial for candidate 1. Additionally weight-sharing also greatly reduces the parameter number which counteracts overfitting in general. This candidate parameter sharing is the key component for making this type of architecture possible in our experiments.

Reduction function In our candidate proposal setting it is crucial to use a reduction function which is invariant under permutation of the candidates (Max, Average, Noisy-AND, ...). Here is a simple example for illustrating the difference between an permutation-invariant reduction function (MAX) and an non-permutation-invariant reduction function (MLP). Let's consider a patient with 5 candidates, where only one candidate correlates with the actual patient label: The case A (0, cancer, 0, 0, 0) should produce the same prediction as the case B (cancer, 0, 0, 0, 0), since the previous stated partial imperfect ordering of the malignant candidates. Using e.g. a MLP as reduction function each candidate-score at a different index has its own weight. When being trained on case A, a MLP would struggle making a correct prediction for case B, since the index-1-weight was never adjusted. In the case of a permutation-invariant classifier solving (0, cancer, 0, 0, 0) is identical to solving all possible permutations of it. This makes learning general features on limited data much more effective. For our final networks we use the naive MAX function as reduction function since it performed the best in our experiments. The intuition behind the MAX function is, that the most "cancerous" candidate also is the direct indicator for the cancer at patient level.

Residual architecture We did not experiment with the core module graph itself. We chose a residual network architecture since its known to have very good performance on image recognition tasks (paper) and reasonable training times. Applying small variations to the architecture (increased number of layers / kernels) did not affect the performance by a significant margin, so we decided early in this competition to focus on high level model design.

Final Ensemble

For the final solution we used four different Patient-Classifier-Networks, which we average to a single prediction. The final hyperparameters and detailed network architectures can be found in the code. This ensemble reaches a final score of 0.427 on the Stage2-Leaderboard.

Network	Isotropic scan resolution (mm^3/px^3)	architecture
05res 3DNet	0.5	3D residual network
07res 3DNet	0.7	3D residual network
05res 2DNet	0.5	2D residual network
07res 2DNet	0.7	2D residual network

Table 1: List of the four final networks and input configurations.