Convolutional Neural Networks for Multidrug-resistant and Drug-sensitive Tuberculosis Distinction

Daniel Braun, Michael Singhof, Martha Tatusch, and Stefan Conrad

Heinrich-Heine-Universität Düsseldorf, Institut für Informatik,
Universitätsstr. 1, 40225 Düsseldorf, Germany
{braun,singhof,conrad}@cs.uni-duesseldorf.de,
martha.tatusch@uni-duesseldorf.de

Abstract. Tuberculosis is a widespread disease and one of the top causes of death worldwide. Especially the distinction between drug-sensitive and multidrug-resistant tuberculosis is still problematic. The ImageCLEF 2017 Tuberculosis Task 1 aims at the development of a machine learning system able to decide whether a patient suffers from multidrug-resistant tuberculosis or not, based solely on a CT scan of that person's lungs.
In this paper we describe our approach to solve this problem. This consists of a preprocessing step that denoises the scans and highlights certain parts of the lungs that we assume to be meaningful for the distinction. We then utilise a shallow convolutional neural network for the following classification.

Keywords: Convolutional Neural Networks, Deep Learning, Shallow Learning, Tuberculosis, Multidrug-resistant Tuberculosis, CT Scans, ImageCLEF 2017

1 Introduction and Performed Tasks

Mycobacterium tuberculosis, the cause of the tuberculosis disease, has been discovered as early as 1882 by Robert Koch, who, in 1905 received the Nobel Prize in Physiology or Medicine for that discovery. According to the World Health Organization (WHO), tuberculosis is one of the top 10 causes of death worldwide. Adding to this, multidrug-resistant tuberculosis (MDR TB) is difficult to distinguish from drug-sensitive tuberculosis (DS TB) and more difficult to cure.

We therefore took part in ImageCLEF 2017 [8] Tuberculosis Task 1 [6], in order to help to establish a comparably cheap and fast automatic detection method for MDR TB based on lung CT scans alone. For this task, a training set of 230 volume scans with labels of the drug-resistancy were given. The objective was to develop a system that is able to predict a score for the probability of MDR TB. Since the number of training samples is relatively small, we opted for a small neural network with very few layers.

The remainder of this paper is structured as follows: In Section 2, we introduce the architecture that we developed during our work on the task. Section 3 gives a brief overview over the results of the challenge and our interpretation. In Section 4 we show some further experiments with additional network architectures and Section 5 summarises the paper and gives an outlook to future work.

2 Method

In this section we describe the main steps of our approach. First we discuss the preprocessing of the CT scans in order to enhance the visibility of importand features. Next, we describe the architecture of our convolutional network. We conclude with a brief description of our training process.

2.1 Preprocessing of CT Scans

First of all, we load the images using the nibabel python library [2].

After loading the image we assume the minimum value in the image as the code for the background of the scan, i.e. the portion of the volumetric scan that has not been scanned in reality. Since the spacing of this values to the actual black value depends on the scanner model, we then set the background value to the second lowest value -1. Then we simply transform the range of values in the scans to the interval $[0, 1]$, since the values used in the images are not consistent throughout the data set. This is likely due to different computer tomograph models used to create the scans.

According to [3], MDR TB is characterised by less frequent large nodules, cavities and bronchial dilatation with respect to DS TB. Since all of those phenomena occur in the lungs, we opted to use the provided lung segmentations that have been extracted by the method described in [5]. The lung segments use the background value b of the corresponding scan as a symbol that a voxel is not part of the lungs and then use $b + 1$ and $b + 2$ for showing the two sides of the lung. Since for the task at hand that division is not of interest, we set b to 0 and both values of $b + 1$ and $b + 2$ to a value of 1. The actual segmentation of the scan is then derived as the voxel-wise product of the CT scan and the segmentation mask.

Inspection of the resulting scans shows, that on the one hand, there is still a large variation of intensity values in different scans (see Figure 1). On the other hand, as pointed out in [3], the means to distinguish MDR TB and DS TB are mainly in the characteristics of the nodules, which are among the lighter portions of the scans. Also, there is a lot of noise in the darker parts of the scans which we try to get rid off with the following step.

We therefore create an intensity histogram consisting of 256 even spaced bins for each of the segmented scans, ignoring the background value. We chose a value of 256 because it is a good tradeoff between colour precision and meaningfulness of the colour ranges spanned by the bins. Then we search for the bin that occurs

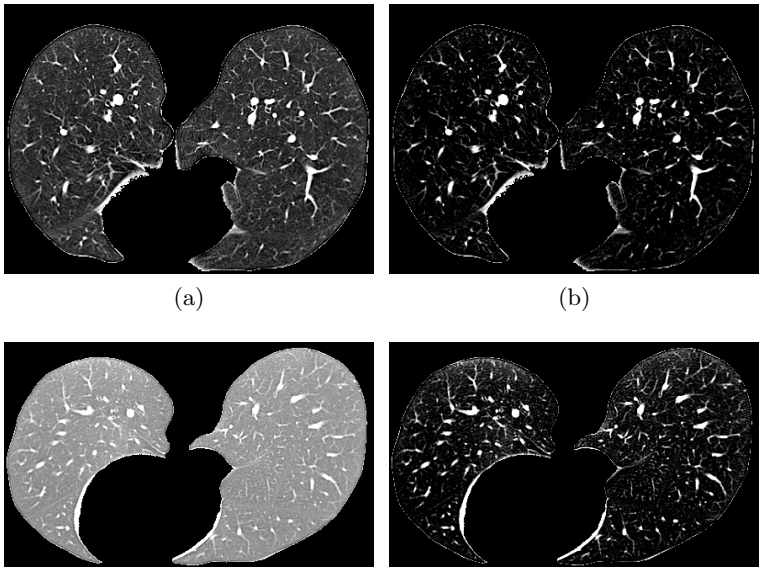


Fig. 1: An example for the different intensities in normalised scans in (a) and (c) and the effect of the denoising on that in (b) and (d).

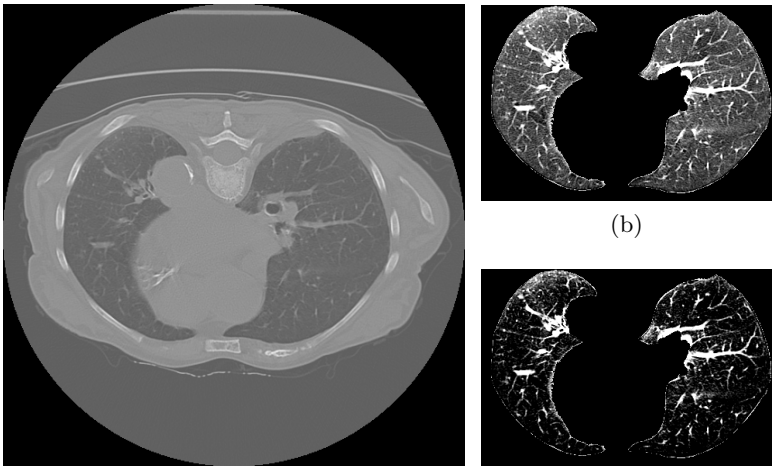
most often and determine the upper border u of that bin. Then we set all values $v \leq u$ to u in order to remove the background noise in the darker parts of the scan. Note that $b \leq u$, as well. As can be seen in Figure 1 (a) and (c), it is not advisable to use one fixed threshold for all images of the data set due to the large variety in intensity. Figure 2 shows an example of the effects of the different steps in our preprocessing pipeline.

We increase the contrast by extending the used range of $[u, 1]$ to $[0, 1]$, again, resulting in a more even intensity among the data set that highlights the nodules and thus the important parts of the scans. Subsequently, we crop the scan to the minimal bounding box of all non-zero values. This results in variable sizes for each dimension.

2.2 Classification with Convolutional Neural Networks

For the actual classification of the scans we put the preprocessed three dimensional data sets in a three dimensional convolutional network based on KERAS [4] with TensorFlow [1] as backend. The architecture of the network is shown in Figure 3.

The size of the preprocessed data is variable in every dimension except for the number of channels, which is always one. This is reflected in the input layer and indeed all of the following layers until the spatial pyramid pooling layer [7],



(a) (b) (c)

Fig. 2: An example of the effects of the preprocessing for slice 52 of MDR_TRN_031. (a) shows the slice of the original ct scan. (b) shows the normalised and segmented slice. (c) shows the same slice after the denoising.

since scaling to a fixed size induces losses in sharpness due to the necessary interpolation. This is especially the case in the z dimension because the distance between slices differs from scan to scan and, in general, that distance is larger than between the pixels in x and y dimensions.

The first layer of the network is a three dimensional max pooling layer with filter and stride size of $(4, 4, 1)$. This means, that for every 4 by 4 patch from each slice, we take the brightest pixel of that patch. We do not reduce the number of slices for the reasons mentioned above. The usage of this layer is to reduce the memory footprint of the input data since the memory on the university cluster's Tesla K20Xm GPUs was limited.

The max pooling is followed by two blocks consisting of a three dimensional convolutional layer with filter size of $(3, 3, 3)$ and ReLU as activation function and a max pooling layer with filter size $(2, 2, 2)$, each. In the first block we use 3 convolutional filters and in the second we use 6. We use a relatively low number of filters in these blocks because the number of training samples is low, too. These two blocks are followed by a dropout layer with a drop probability of 0.25.

The final fully connected layer expects an input tensor of a fixed size. Due to the variable input size, a normal flatten layer is not applicable to achieve this. We therefore use a spatial pyramid pooling layer as introduced by He et.al. in [7]. The layer introduced in this paper is intended for two dimensional images, but the extension to spatial images such as CT scans is straight forward. Basically, the idea of spatial pyramid pooling is to utilise max pooling on a fixed number of

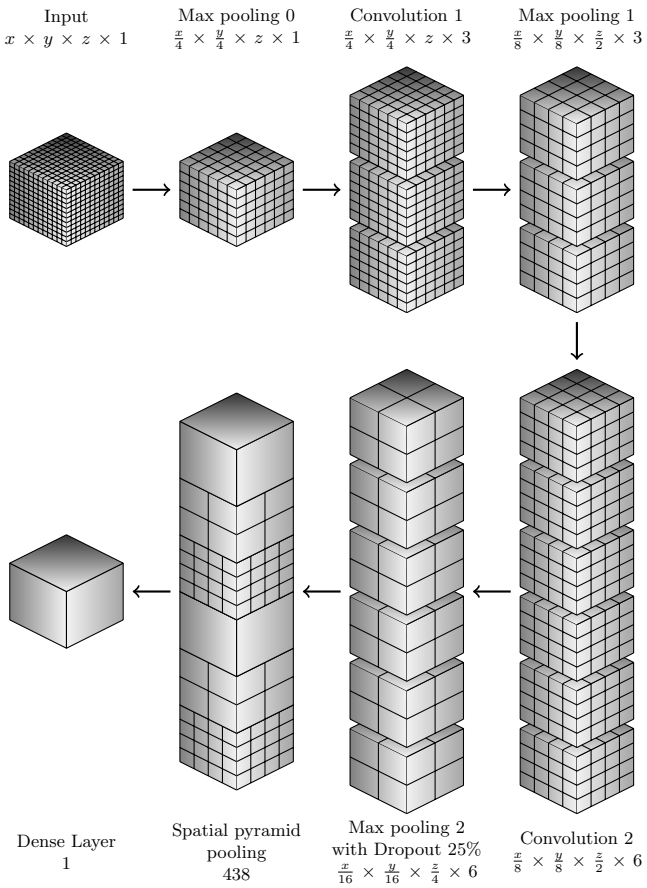


Fig. 3: Architecture of our CNN. The variable sizes of the input are denoted x, y and z.

regions for every channel of the input data for this layer. The size of the regions is determined by the number of regions.

For example, let us assume the input data for the spatial pyramid layer in one instance has a size of $100 \times 100 \times 100$ voxels and one channel and we utilise a spatial pyramid pooling layer with 1 and 4 as number of bins per dimension[1], resulting in a vector v with $1^3 + 4^3 = 65$ dimensions. Hereby, the first entry of v is the maximum value over the whole of the input data, while the second entry is the maximum value of the first $25 \times 25 \times 25$ voxel block and so on.

In our network, we use a pooling list of 1, 2, and 4 and we have 6 input channels, thus resulting in an output size of $(1^3 + 2^3 + 4^3) \cdot 6 = 438$.

[1] This is in contrast to the notation used in [7], where the parameters determine the number of bins, i.e. the parameters 1 and 4 would result in an output size of 5.

Finally, the fully connected layer has a softmax activation function and one output node. We trained the model with categorical crossentropy as loss function and Adam [9] as optimiser. The whole network has 1015 trainable parameters.

2.3 Training

For the training of our final model we used the whole of the given training set and threaded it through our preprocessing as described above. The training set consists of 230 scans, 134 of which are labelled DS TB and 96 MDR TB. We did not use any additional training data, nor did we add additional annotations to the training data.

We also opted against the usual techniques of artificial enlargement of the training set, for example by rescaling or shifting the input images or using filters on them. In respect to rescaling, all CT scanners known to us produce output images of 512×512 pixels per slide so that a scaling here would lead to unrealistic data. Shifting of the slides would not be unrealistic, however, due to our usage of the given lung masks and the subsequent cropping to these regions of interest, shifted scans would be exact duplicates after preprocessing. Finally, we did not use any filters on the test set, since we are not sure what the indicators of MDR tuberculosis are. We thus cannot ensure that an artificially changed scan does not change its class.

3 Evaluation

In this section we give a short interpretation of the preliminary evaluation results as published on the tuberculosis task homepage[2]. The test dataset was meant to consist of 214 scans, of which 101 are labelled DS TB and 113 MDR TB. However, the actual dataset passed to the participants only consisted of 213 scans, as scan MDR_TST_123 was missing. Table 1 gives the complete list of submitted runs sorted by AUC score. AUC scores range from 0.5825 to 0.4596 and accuracy is in a range from 0.4413 to 0.5681. The best values for every score are marked bold in the table.

The first thing to note in this context is, that none of these results, including our own, are exceptionally good. Given that the size of the classes was known, an ACC score of 0.5280[3] could be achieved by statically classifying all test data set points as MDR tuberculosis, i.e. 1. For AUC the result for any guess of this fashion is 0.5.

Our submitted runs are all results of the architecture described in the previous section. Only the number of training epochs differs and is given by the number at the end of the run filename. We selected a number of runs that came close in numbers to the given class distribution and that also were confident in the prediction score, i.e. they have a high number of predictions that are either

[2] `http://www.imageclef.org/2017/tuberculosis`

[3] Value is computed on the given distribution. The real achievable value is either 0.5305 or 0.5258 depending on whether there are 113 or 112 samples for MDR.

#	Group Name	Run	AUC	ACC
1	MedGIFT	MDR_Top1_correct.csv	**0.5825**	0.5164
2	MedGIFT	MDR_submitted_topBest3_correct.csv	0.5727	0.4648
3	MedGIFT	MDR_submitted_topBest5_correct.csv	0.5624	0.4836
4	SGEast	MDR_LSTM_6_probs.txt	0.5620	0.5493
5	SGEast	MDR_resnet_full.txt	0.5591	0.5493
6	SGEast	MDR_BiLSTM_25_wcrop_probs.txt	0.5501	0.5399
7	UIIP	MDR_supervoxels_run_1.txt	0.5415	0.4930
8	SGEast	MDR_LSTM_18_wcrop_probs.txt	0.5404	0.5540
9	SGEast	MDR_LSTM_21wcrop_probs.txt	0.5360	0.5070
10	MedGIFT	MDR_Top2_correct.csv	0.5337	0.4883
11	HHU DBS	MDR_basecnndo_212.csv	0.5297	**0.5681**
12	SGEast	MDR_LSTM_25_wcrop_probs.txt	0.5297	0.5211
13	BatmanLab	MDR_submitted_top5.csv	0.5241	0.5164
14	HHU DBS	MDR_basecnndo_113.csv	0.5237	0.5540
15	MEDGIFT UPB	MDR_TST_RUN_1.txt	0.5184	0.5352
16	BatmanLab	MDR_submitted_top4_0.656522.csv	0.5130	0.5024
17	MedGIFT	MDR_Top3_correct.csv	0.5112	0.4413
18	HHU DBS	MDR_basecnndo_132.csv	0.5054	0.5305
19	HHU DBS	MDR_basecnndo_182.csv	0.5042	0.5211
20	HHU DBS	MDR_basecnndo_116.csv	0.5001	0.4930
21	HHU DBS	MDR_basecnndo_142.csv	0.4995	0.5211
22	HHU DBS	MDR_basecnndo_120.csv	0.4935	0.4977
23	SGEast	MDR_resnet_partial.txt	0.4915	0.4930
24	BatmanLab	MDR-submitted_top1.csv	0.4899	0.4789
25	BatmanLab	MDR_SuperVx_Hist_FHOG_rf_0.648419.csv	0.4899	0.4789
26	Aegean Tubercoliosis	MDR_DETECTION_EXPORT2.csv	0.4833	0.4648
27	BatmanLab	MDR_SuperVx_FHOG_rf_0.637994.csv	0.4601	0.4554
28	BioinformaticsUA	MDR_run1.txt	0.4596	0.4648

Table 1: Preliminary evaluation results ranked by AUC score. Our submissions are those by "HHU DBS".

close to zero or one. From the results, we see the influence of the dropout mechanism on the learning, since the result quality does not correlate with the number of epochs in training. This is especially evident in the block of ranks 18 to 22. Also, the lowest number of submitted epochs, namely 113 has the second highest AUC score of our runs, while the highest number of epochs, 212, has the highest AUC score of our runs. This run also has the highest accuracy of all submitted runs.

4 Further Experiments

In this section, we present some other network architectures that we tested after the run submission deadline on a new Nvidia Quadro P6000 with 24GB of

Layer	Number of Filters	Filter Size	Stride
Maxpooling 0	n.a.	(2, 2, 1)	(2, 2, 1)
Convolution1	3	(3, 3, 3)	(1, 1, 1)
Maxpooling 1	n.a.	(2, 2, 2)	(1, 1, 1)
Convolution2	6	(3, 3, 3)	(1, 1, 1)
Maxpooling 2	n.a.	(2, 2, 2)	(1, 1, 1)
Dropout 0.25	n.a.	n.a.	n.a.
Spatial Pyramid	[1, 2, 4]	n.a.	n.a.
Dense	n.a.	n.a.	n.a.

Table 2: Network architecture Alt. 1 with smaller initial max pooling.

Layer	Number of Filters	Filter Size	Stride
Convolution1_1	4	(5, 5, 3)	(3, 3, 1)
Convolution1_2	4	(3, 3, 3)	(1, 1, 1)
Maxpooling 1	n.a.	(2, 2, 2)	(1, 1, 1)
Convolution2_1	8	(3, 3, 3)	(1, 1, 1)
Convolution2_2	8	(3, 3, 3)	(1, 1, 1)
Maxpooling 2	n.a.	(2, 2, 2)	(1, 1, 1)
Dropout 0.25	n.a.	n.a.	n.a.
Spatial Pyramid	[1, 2, 4, 8]	n.a.	n.a.
Dense	n.a.	n.a.	n.a.

Table 3: Deeper network architecture Alt. 2 with initial convolutional layer.

memory in order to evaluate the effects of some of our design decisions. Examples for this are the strong initial max pooling and the depth of the network. Tables 2 and 3 give an overview over these architectures.

The architecture described in Table 2 is similar to our main architecture as given in Figure 3, apart from the smaller filter size during the initial max pooling step. We used this architecture to test, whether using a strong initial max pooling has a negative impact on the classification.

In contrast to the architecture described in Section 2.2, the one given by Table 3 is a more conservative design built on two convolutional blocks with two convolutional layers and one max pooling layer, each. The larger filter size and stride setting in Convolution1_1 is due to the fact, that without those settings, the data was too large for the graphic memory to compute a whole epoch. We used this network architecture in order to give the network a greater degree of freedom in training via a larger number of trainable parameters. This is supported by the addition of an eight region per dimension portion to the spatial pyramid filtering.

We tested all network architectures we described thus far on different splits of the 230 training scans. For this, we used the preprocessed scans that we achieved by utilising the pipeline described in Section 2.1. Then, for each split, we chose

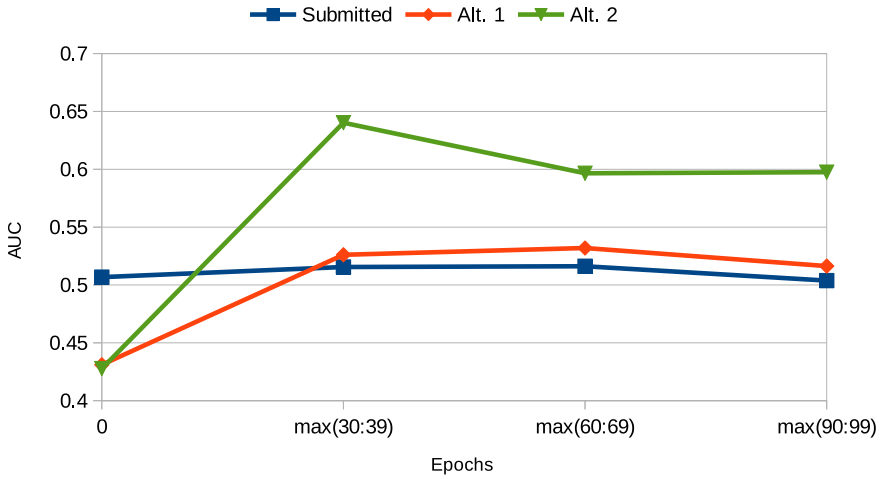


Fig. 4: Results of the preliminary AUC tests. AUC scores are computed with scikit-learn [10].

30 of the scans to act as a test set and trained each network for 100 epochs on the remaining 200 scans. We then tested on the corresponding test set. For each split, we started with previously untrained networks in order to not induce overfitting. The results of these runs are shown in Figures 4 and 5. To reduce the impact of the dropout, apart from the first epoch, for the comparison we chose the best epoch regarding the predicted AUC value of the given range. It can be seen here, that the difference between the submitted network and Alt. 1 is marginal. From this we deduce, that the stronger initial max pooling's effect is negligible.

5 Conclusion

As the results presented in section 3 show, none of the submitted runs yields outstanding results. This shows, that the distinction between DS TB and MDR TB is difficult based on CT scans only. On the other hand, [3] indicates that most likely the distinction can be made upon the nodules. Therefore, we think it would be interesting to have an annotated dataset in respect to those, i.e. with annotated bounding boxes for nodules, and cavities among other meaningful findings. The challenge itself would not have to change, i.e. it would still be possible to build a system that gets a full CT scan as input and outputs the probability of MDR TB. Also, a larger number of training data would be desirable. We believe, that with more training data, larger networks become more feasible and therefore a more fine grained distinction might be possible.

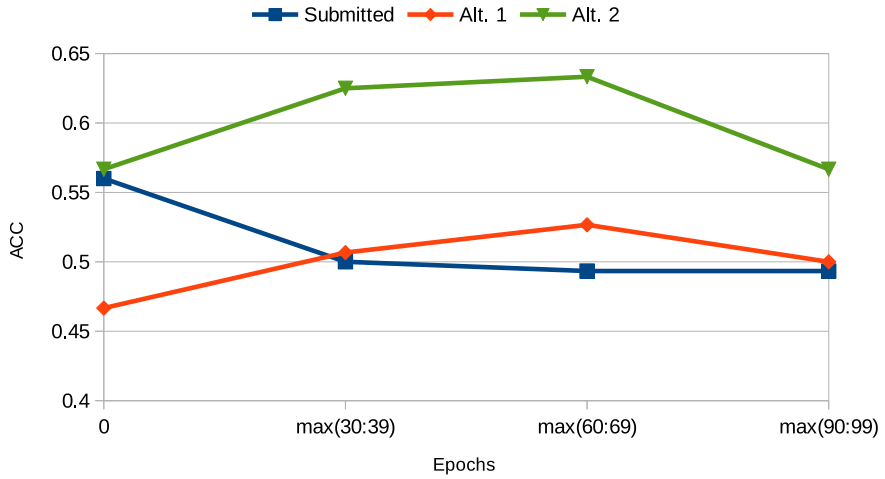


Fig. 5: Results of the preliminary ACC tests. ACC scores are computed with scikit-learn [10].

Future work in our case will include possible revisions of the preprocessing step, since we believe that a focus on the relevant features such as nodules and cavities is advisable. Also, we would like to test the architectures presented in Section 4 on the test data, if possible. The results of our second alternative approach look especially promising and could yield comparable results to the winner of the challenge on the full test set. We will also test deeper architectures in the future.

6 Acknowledgements

Computational support and infrastructure was partially provided by the "Centre for Information and Media Technology" (ZIM) at the University of Düsseldorf (Germany). We would like to especially thank Philipp Helo Rehs for his help with the cluster.

We also would like to thank Guido Königstein, our research group admin.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V.,

Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015), http://tensorflow.org/, software available from tensorflow.org
2. Brett, M., Hanke, M., Cipollini, B., Côté, M.A., Markiewicz, C., Gerhard, S., Larson, E., Lee, G.R., Halchenko, Y., Kastman, E., cindeem, Morency, F.C., moloney, Millman, J., Rokem, A., jaeilepp, Gramfort, A., van den Bosch, J.J., Subramaniam, K., Nichols, N., embaker, bpinsard, chaselgrove, Oosterhof, N.N., St-Jean, S., Amirbekian, B., Nimmo-Smith, I., Ghosh, S., Varoquaux, G., Garyfallidis, E.: nibabel: 2.1.0 (Aug 2016), https://doi.org/10.5281/zenodo.60808
3. Cha, J., Lee, H.Y., Lee, K.S., Koh, W.J., Kwon, O.J., Yi, C.A., Kim, T.S., Chung, M.J.: Radiological Findings of Extensively Drug-Resistant Pulmonary Tuberculosis in Non-AIDS Adults: Comparisons with Findings of Multidrug-Resistant and Drug-Sensitive Tuberculosis. Korean Journal of Radiology 10(3) (2009)
4. Chollet, F., et al.: Keras. https://github.com/fchollet/keras (2015)
5. Dicente Cid, Y., Jiménez del Toro, O.A., Depeursinge, A., Müller, H.: Efficient and fully automatic segmentation of the lungs in CT volumes. In: Goksel, O., Jiménez del Toro, O.A., Foncubierta-Rodríguez, A., Müller, H. (eds.) Proceedings of the VISCERAL Anatomy Grand Challenge at the 2015 IEEE ISBI. pp. 31–35. CEUR Workshop Proceedings, CEUR-WS (May 2015)
6. Dicente Cid, Y., Kalinovsky, A., Liauchuk, V., Kovalev, V., Müller, H.: Overview of ImageCLEFtuberculosis 2017 - Predicting Tuberculosis Type and Drug Resistances. CLEF working notes, CEUR (2017)
7. He, K., Zhang, X., Ren, S., Sun, J.: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: European Conference on Computer Vision (2014)
8. Ionescu, B., Müller, H., Villegas, M., Arenas, H., Boato, G., Dang-Nguyen, D.T., Dicente Cid, Y., Eickhoff, C., Garcia Seco de Herrera, A., Gurrin, C., Islam, B., Kovalev, V., Liauchuk, V., Mothe, J., Piras, L., Riegler, M., Schwall, I.: Overview of ImageCLEF 2017: Information Extraction from Images. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction 8th International Conference of the CLEF Association, CLEF 2017. Lecture Notes in Computer Science, vol. 10456. Springer, Dublin, Ireland (September 11-14 2017)
9. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. CoRR abs/1412.6980 (2014)
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011)