UNIVERSITÄT HEIDELBERG
ZUKUNFT
SEIT 1386

Ruprecht-Karls-Universität
Heidelberg

Final project report

Advanced Machine Learning

Prof. Dr. Köthe

SIIM-FISABIO-RSNA COVID-19 Detection

Authors:	Tobias Richstein (3596554) Julian Seibel (3601340) Torben Krieger (3663391)
Field of Studies:	M. Sc. Applied Computer Science
Period:	Summer term 2021

Contents

1 Introduction and problem definition

1.1 Kaeggle Challenge

Written by Torben

1.2 Related Work

Written by Julian

Exploring the possibilities of Computer Vision methods in medical image analysis reaches back to the late 90's where first proposals of Computer Aided detection and diagnosis (CAD) systems were designed like in [13], where the authors detect aluminium dust-induced lung diseases. With the rise of deep learning, medical image analysis on X-rays is subject of recent academic research reaching from detecting diseases like pneumonia [3] [6] [9] to tuberculosis and different thoracic diseases [12] and even pulmonary [25].

Even though it is not completely clear how a COVID-19 infection impacts the human body, it is possible to detect typical patterns in lungs using for example chest X-rays of affected patients. This opens many possibilities to provide fast and solid CAD solutions while build on knowledge of previous works.

Since the pandemic started in 2020, there are many works published that deal with providing a reliable system for detecting COVID-19 infections using medical images. A CAD based process would help the public health sector and relieve medical personal in hospitals by increasing the automation of X-ray examination. There is already some work proposed in this direction, including the COVID-Net [26] by Wang, Lin, and Wong, where they use a machine-driven exploration process to design a deep convolution learning model capable of classifying X-ray images of lungs into three categories (normal, pneumonia, COVID-19). The results of their study look promising and although the authors stated explicitly the non-production-ready state, the work can be used as a basis for future projects.

Another work using a deep neural network is the proposed CovidAID [16] network by Mangal, Kalia, Rajgopal, *et al.* In their approach, they use a pre-trained CheXNet [20] Convolutional Neural Network (CNN) while substituting the output layer of the model to fit it to their needs of predicting one of the four classes (normal, bacterial pneumonia, viral pneumonia, COVID-19). The authors applied transfer learning by only actively apply the train algorithm to the final layers of their proposed model, whereas the backbone weights are frozen. In their final comparison, the authors reported to significantly improve upon the results on the previous introduced COVID-Net. Similar to this the proposed model [1] by Albahli and Albattah using transformation learning Systems that are try to detect positive cases vs.

using weighted voting ensemble [15]

[4]

However, since the challenge and the goal set in this project, we need to not only classify input images as COVID-infections, we rather need to detect and locate suspicious areas in such images. There is one work proposed going in this direction:

“Fast deep learning computer-aided diagnosis of COVID-19 based on digital chest x-ray images” [2]

1.3 Proposed Solution

Written by Tobias

just a short introduction to our solution, models will be convered in 3 on page 4

2 Data

2.1 NIH Data

Written by Tobias

2.2 RSNA Data

Written by ?

2.3 SIIM COVID-19 Data

Written by ?

3 COVID-19 Detection

3.1 Faster R-CNN

Written by Tobias Richstein

The Faster R-CNN network architecture proposed in [21] by Ren, He, Girshick, *et al.* is an evolutionary step in a line of Region Based CNNs (R-CNNs) which are CNNs that can perform object detection on images. When given an image, an R-CNN is able to predict bounding boxes for detected objects and also classify them. Each predicted bounding box is also given a confidence score that expresses how reliable the model assumes this result is. State of the art Faster R-CNNs achieve mean Average Precision (mAP) scores with a 0.5 Intersection over Union (IoU) threshold of 0.484 on a reference COCO object detection validation set making it very well suitable for all kinds of detection tasks such as the one at hand.

The original R-CNN was proposed by Girshick, Donahue, Darrell, *et al.* in [8]. This original R-CNN consists of three modeules: The first one generates category independent region proposals which are regions in the image that the network believes could have relevant objects in them. In theory R-CNN is agnostic to which network performs these region proposals but in the paper a method called *Selective Search* [24] is used to generate 2000 region proposals. In the second module the contents of these proposed bounding boxes are passed to a backbone network which in the paper was an *AlexNet* CNN [14], but could be any suitable network architecture to generate a 4096-length feature vector. Then the feature vector is passed onto the third module which is a set of Support Vector Machines (SVMs), each trained on one specific class, to predict the class of the object encompassed by the bounding box.

The approach described in [8] does work fairly well but has some big drawbacks. First it requires training of the backbone CNN and SVM classifiers and then a separate training of the region proposal network, making it very slow to train. Another issue with this architecture was, that inference was very slow with images taking multiple seconds to be processed on a GPU which is most often not sufficient for any sort of time requirements that object detection tasks may have.

To overcome the issues of the original R-CNN paper, Fast R-CNN was proposed a year later in [7]. Now, instead of passing each proposed region through the CNN separately, the entire image is processed once for which the CNN generates a feature map that is valid across all regions. Again using any CNN backbone works, but the authors used the then state of the art *VGG16* architecture [22]. While this approach does make the network faster, it still requires an external region proposal network to feed the Fast R-CNN with proposals and the image. Some speedup is achieved by being able to train the classifier and bounding box regressor at the same time.

In a third advancement the concept of the Faster R-CNN is introduced in [21]. The most noticeable change is that the region proposal network is now built in and no longer requires using Selective Search or other methods. After the backbone convolution the feature maps can now be passed onto both the region proposal network and the classifier which means that they share the results of the convolution making the network faster. In the original paper, the authors use the same *VGG16* backbone as in the Fast R-CNN paper but note that a larger *ResNet*[10] model might lead to better results at the cost of more compute intensive training and inference.

The hint that a *ResNet* architecture might be the better backbone to use with a Faster R-CNN, led us to research these kinds of networks. After Krizhevsky, Sutskever, and Hinton introduced the widely acclaimed deep CNN *AlexNet* a trend started to make these sorts of networks ever deeper, using more and more convolution layers under the assumption that more layers would lead to the detection of finer and maybe more hidden features in images. In [10] however, He, Zhang, Ren, *et al.* show that this assumption only holds to a certain degree and show that a 20 layer network can perform much better than the same network with 56 layers. There are many proposed theories why this might be but the authors focus on fixing it by introducing a so called *Residual Block* which essentially passes the input and output of a layer onto the next layer by adding a shortcut identity mapping from input to output. Also so called bottlenecks are used which perform a dimensionality reduction with 1×1 convolutions. In doing so the authors are able to train networks that are hundreds or thousands of layers deep while improving classification metrics with each layer added.

Building upon *ResNet*, Xie, Girshick, Dollár, *et al.* propose *ResNeXt* in [27]. This network architecture introduces the concept of cardinality C where a residual ResNet block is split into C identical groups of operations called paths. ResNeXt networks are described by the number of layers they have, their cardinality and the size of their bottleneck. The larger each parameter is, the more computationally intensive. As a middle ground we picked a model with 101 layers, a cardinality of 32 and a bottleneck size of 8. This is referred to as a `ResNeXt 101 32x8d`.

Training of the backbone

First we trained the ResNeXt network to have a performant backbone that the Faster R-CNN can utilize. A reference ResNeXt model architecture and implementation can be obtained directly from the makers of PyTorch [19], which we did. This reference implementation has also been pre-trained on the ImageNet dataset, meaning that we only fine-tune the weights to our use-case. We train the model on the NIH dataset described in section 2.1 on page 3 and only expect it to predict the classes of illnesses that can be seen in the X-rays. We encode the ground truths, consisting of the 14 classes of the NIH dataset, as one-hot vectors and therefore also expect output tensors of the same dimension. Like in the original ResNeXt paper, we also use a Stochastic Gradient Descent (SGD) optimizer that has Nesterov acceleration during training. Our learning rate decays over time and follows the equation given below which was originally proposed in [11] and modified slightly to provide a learning rate floor of $0.05 * \mathrm{lr}_{\mathrm{initial}}$:

$$\mathrm{lr}_t = \left(\frac{1}{2}\left(1 + \cos\left(\frac{t * \pi}{T}\right)\right) * 0.95 + 0.05\right) * \mathrm{lr}_{\mathrm{initial}}$$

where t is the current learning rate scheduler step and T is the total number of epochs. We take a step every other epoch and start with a learning rate of $\mathrm{lr}_{\mathrm{initial}} = 0.001$.

As described in the ResNeXt paper we load the images and then perform the augmentations necessary to fit the model requirements. To do so, we use a custom dataloader that provides batches of images together with the one-hot encoded ground truth vectors. The augmentation steps done during dataloading include:

- Resize the image to have 256 pixels on the shorter side
- Perform a 224×224 crop in the center of the resized image
- Normalize the RGB channels in range 0 to 1 to have a mean of $R = 0.485; G = 0.456; B = 0.406$ and a standard deviation of $R = 0.229; G = 0.224; B = 0.225$

To prevent overfitting during training we also randomly apply additional augmentations such as horizontal flips ($p = 0.5$), random blurs ($p = 0.3$), random rotations of up to 20° ($p = 1$) or random erasings of between 1 and 10 percent of the image area ($p = 0.3$).

Since we have somewhat limited hardware resources at our disposal in comparison to large scale compute clusters that are often used for such training tasks by researchers, we also apply a method called *Autocasting* to speed up training and allow us to use larger batch sizes. The basis of Autocasting is the ability to use mixed precision during network training. While most frameworks such as PyTorch usually use 32bit floating point numbers (single precision) for all calculations, it has been shown that performing some operations with 16bit representations (half precision) does not penalize accuracy but provides a large speedup since more data can fit in the most often constrained GPU memory and the also constrained data transfer bandwidth can be used more effectively [17]. The GPUs that we have at our disposal also feature special matrix multiplication hardware that works best with half precision numbers, meaning that we profit from mixed precision training in a significant way. The speedup for the ResNeXt training for example was almost twice as fast as before. The decision whether to perform operations at half precision is made automatically by PyTorch when the model is wrapped in an autocasting decorator.

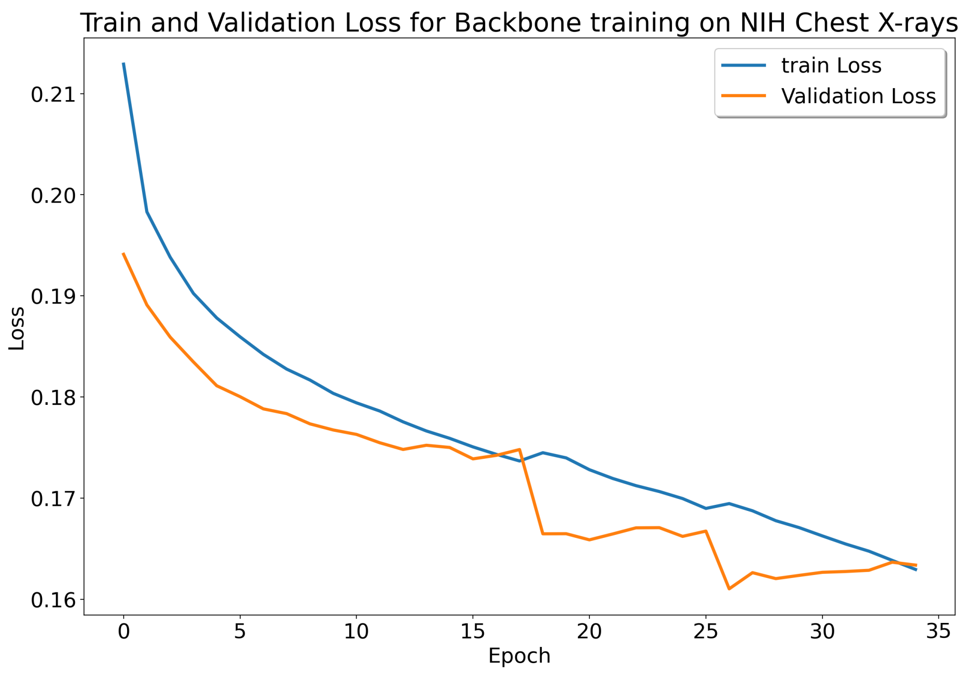

Figure 1: Loss figures of the ResNeXt training

We train the ResNeXt with a batch size of 32 (like in the original paper) and perform 35 epochs. To calculate the loss we use Binary Cross Entropy but with Logits as recommended for mixed precision training which uses the log-sum-exp trick to improve the numerical stability and avoid loss terms that cannot be represented by half precision [18]. The loss numbers for the training and validation loss can be seen in 1. It can be seen that in the end some overfit occures where the train loss keeps decreasing and the validation loss stays mostly constant or even increases very slightly. In the end we still decided to use the model after 35 epochs since the loss figures are very good and it also evaluates very well as will be shown later in chapter 4.1 on page 10.

Training of the Faster R-CNN

With the backbone network trained, we could now train the Faster R-CNN on the actual detection task of predicting where lung opacities are located in a patient's X-ray image. This training shares a lot of optimizations with the backbone network described above. We use the same SGD optimizer and learning rate schedule and train for 50 epochs which does not take too long due to the limited number of training images. We also again use autocasting since the speed improvements are too good to leave out.

Due to the limited number of samples available in the SIIM dataset, we now augment the

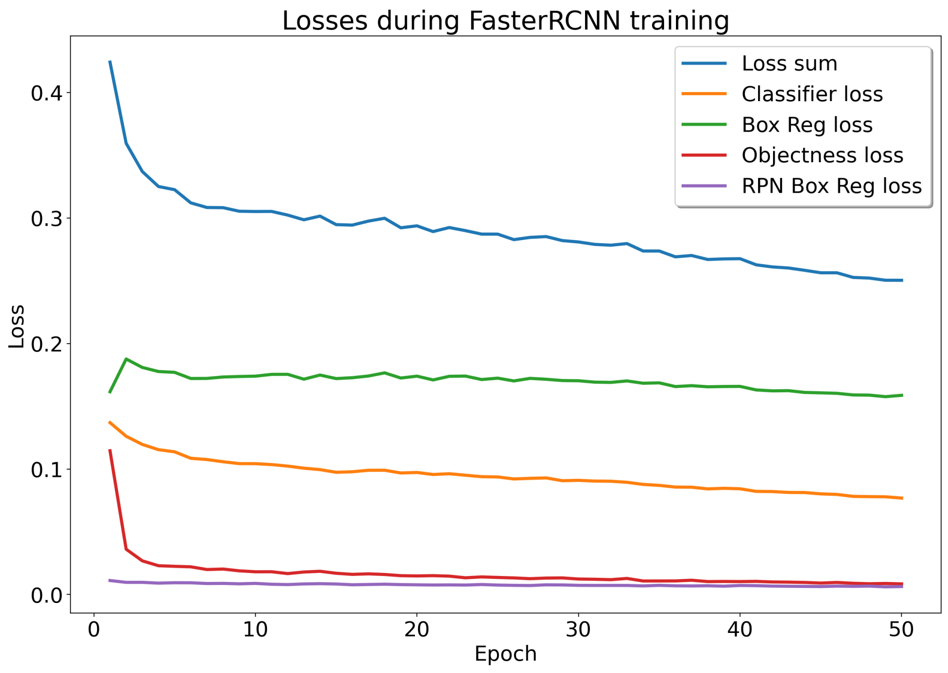

Figure 2: Loss figures of the Faster R-CNN training

images more extensively to further prevent overfitting. Because we now have bounding boxes in the aforementioned (?) COCO format we also have to apply all augmentations to those too. To also allow the network to better detect small opacities and details we now train with a much larger image size of 512×512. We also perform random horizontal flips ($p = 0.3$), random shifts with rotations of maximum 20° ($p = 0.3$), one of random sharpen ($p = 0.5$) or blur ($p = 0.25$), random brightness and contrast adjustments ($p = 0.3$) and random circular cutouts (max. 6; $p = 0.3$). During inference however we pass the inputs as 1024×1024 images to make the results even clearer. As with the backbone net, we also adjust the RGB channels to fit the required mean and standard deviation values.

Due to the much larger input images and network size we can only train the Faster R-CNN with a batch size of 10 and perform validation with a batch size of 6. During training of a Faster R-CNN multiple loss values have to be taken into account since there are the two tasks of classification and bounding box prediction. Detailed loss figures can be seen in figure 2. As will be evidenced later in chapter 4.2 on page 10 after 50 epochs there was already some overfit even though the loss numbers look promising.

Per default a Fast(er) R-CNN uses a smooth L1 loss for the box regression as described in [7] which according to the authors prevents exploding gradients unlike loss functions proposed in earlier R-CNN revisions. However, to try and improve convergence speeds and model accuracy, we also implemented a Complete-IoU (CIoU) loss, proposed in [28] and described in more detail in section 3.2, for the regressor. Unfortunately this did not work at all and the model converged a lot slower than anticipated and sometimes even became a lot worse over time. The reasons for this would need to be investigated further but due to time constraints we had to revert back to using the default smooth L1 loss function which in the end also proved quite capable as will be shown later in the evaluation in section 4.2 on the next page.

3.2 YOLO

Written by Julian

3.3 Study-Level model

Written by Torben

4 Evaluation

4.1 Evaluation of ResNeXt

Written by Tobias

accuracy, f1, usw.

4.2 Evaluation of Faster R-CNN and YOLO

Written by Julian

4.3 Evaluation of Study-Level Model

Written by Torben

5 Proposed web application

Written by Tobias !

6 Conclusion

Written by All

References

[1] S. Albahli and W. Albattah, "Detection of coronavirus disease from x-ray images using deep learning and transfer learning algorithms," *Journal of X-ray Science and Technology*, no. Preprint, pp. 1–10, 2020.

[2] M. A. Al-antari, C.-H. Hua, J. Bang, and S. Lee, "Fast deep learning computer-aided diagnosis of covid-19 based on digital chest x-ray images," *Applied Intelligence*, vol. 51, no. 5, pp. 2890–2907, 2021.

[3] S. Bharati, P. Podder, and M. R. H. Mondal, "Hybrid deep learning for detecting lung diseases from x-ray images," *Informatics in Medicine Unlocked*, vol. 20, p. 100 391, 2020, ISSN: 2352-9148. DOI: `https://doi.org/10.1016/j.imu.2020.100391`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2352914820300290`.

[4] L. Brunese, F. Mercaldo, A. Reginelli, and A. Santone, "Explainable deep learning for pulmonary disease and coronavirus covid-19 detection from x-rays," *Computer Methods and Programs in Biomedicine*, vol. 196, p. 105 608, 2020.

[5] S. Candemir and S. Antani, "A review on lung boundary detection in chest x-rays," *International journal of computer assisted radiology and surgery*, vol. 14, no. 4, pp. 563–576, 2019.

[6] J. Garstka and M. Strzelecki, "Pneumonia detection in x-ray chest images based on convolutional neural networks and data augmentation methods," in *2020 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, 2020, pp. 18–23. DOI: `10.23919/SPA50552.2020.9241305`.

[7] R. Girshick, "Fast r-CNN," *arXiv:1504.08083 [cs]*, Sep. 27, 2015. arXiv: `1504.08083`.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv:1311.2524 [cs]*, Oct. 22, 2014. arXiv: `1311.2524`.

[9] N. Gupta, D. Gupta, A. Khanna, P. P. Rebouças Filho, and V. H. C. de Albuquerque, "Evolutionary algorithms for automatic lung disease detection," *Measurement*, vol. 140, pp. 590–608, 2019.

[10] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *arXiv:1512.03385 [cs]*, Dec. 10, 2015. arXiv: `1512.03385`.

[11] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, “Bag of tricks for image classification with convolutional neural networks,” *arXiv:1812.01187 [cs]*, Dec. 5, 2018. arXiv: `1812.01187`.

[12] E. Jangam, C. S. R. Annavarapu, and M. Elloumi, “Deep learning for lung disease detection from chest x-rays images,” in *Deep Learning for Biomedical Data Analysis*, Springer, 2021, pp. 239–254.

[13] T. Kraus, K. Schaller, J. Angerer, and S. Letzel, “Aluminium dust-induced lung disease in the pyro-powder-producing industry: Detection by high-resolution computed tomography,” *International archives of occupational and environmental health*, vol. 73, no. 1, pp. 61–64, 2000.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 24, 2017, ISSN: 0001-0782, 1557-7317. DOI: `10.1145/3065386`. [Online]. Available: `https://dl.acm.org/doi/10.1145/3065386` (visited on 09/08/2021).

[15] I. E. Livieris, A. Kanavos, V. Tampakas, and P. Pintelas, “A weighted voting ensemble self-labeled algorithm for the detection of lung abnormalities from x-rays,” *Algorithms*, vol. 12, no. 3, p. 64, 2019.

[16] A. Mangal, S. Kalia, H. Rajgopal, K. Rangarajan, V. Namboodiri, S. Banerjee, and C. Arora, *Covidaid: Covid-19 detection using chest x-ray*, 2020. arXiv: `2004.09803 [eess.IV]`.

[17] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, “Mixed precision training,” *arXiv:1710.03740 [cs, stat]*, Feb. 15, 2018. arXiv: `1710.03740`.

[18] Pytorch Team, *Automatic Mixed Precision Package*. [Online]. Available: `https://pytorch.org/docs/stable/amp.html#prefer-binary-cross-entropy-with-logits-over-binary-cross-entropy`.

[19] Pytorch Team, *ResNeXt*. [Online]. Available: `https://pytorch.org/hub/pytorch_vision_resnext/`.

[20] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng, *Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning*, 2017. arXiv: `1711.05225 [cs.CV]`.

[21] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-CNN: Towards real-time object detection with region proposal networks,” *arXiv:1506.01497 [cs]*, Jan. 6, 2016. arXiv: 1506.01497.

[22] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv:1409.1556 [cs]*, Apr. 10, 2015. arXiv: 1409.1556.

[23] R. A. Solovyev and W. Wang, “Weighted boxes fusion: Ensembling boxes for object detection models,” *CoRR*, vol. abs/1910.13302, 2019. arXiv: 1910.13302. [Online]. Available: http://arxiv.org/abs/1910.13302.

[24] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013, Publisher: Springer.

[25] P. Vieira, O. Sousa, D. Magalhes, R. Rablo, and R. Silva, “Detecting pulmonary diseases using deep features in x-ray images,” *Pattern Recognition*, p. 108 081, 2021.

[26] L. Wang, Z. Q. Lin, and A. Wong, “Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images,” *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.

[27] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” *arXiv:1611.05431 [cs]*, Apr. 10, 2017. arXiv: 1611.05431.

[28] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, “Enhancing geometric factors in model learning and inference for object detection and instance segmentation,” *arXiv:2005.03572 [cs]*, Jul. 5, 2021. arXiv: 2005.03572.